HONEYWELL

MULTICS ADMINISTRATION, MAINTENANCE, AND OPERATIONS COMMANDS

SOFTWARE

MULTICS

ADMINISTRATION, MAINTENANCE, AND OPERATIONS COMMANDS

SUBJECT

Description of Multics Commands Available for Use by System Administrators, System Maintainers, and System Operators

SPECIAL INSTRUCTIONS

This document contains Multics commands consolidated from the following Multics manuals: MAM System Administrator (AK50), MAM Project Administrator (AK51), MAM Registration and Accounting Administrator (AS68), MAM Resource Control (CC74), MAM Communications (CC75), Bulk I/O (CC34), Multics Operator's handbook (AM81), Multics System Metering (AN52), and Multics System Diagnostic Aids (AR97).

Throughout this manual, marginal change indicators (change bars and asterisks) only indicate technical information added (or deleted) to support Multics Release 11.0. Marginal change indicators are not used to indicate information shifted from another manual.

SOFTWARE SUPPORTED

Multics Software Release 11.0

ORDER NUMBER

GB64-00

June 1985

Honeywell

PREFACE

This document describes those Multics Commands specifically designed for use by individuals filling the roles of (1) system administrators, (2) system maintainers, or (3) system operators.

System administrators provide their sites with a particular Multics operating environment. They are responsible for such tasks as controlling and allocating resources; registering projects and users; creating load control groups; setting prices on resources; setting limits on, and billing for, resource usage; scheduling system activities such as hours of operation, shift change times, and unattended service; describing site parameters and setting site options; and assuring system security.

System maintainers configure and tune the operating system to make it comply with the special requirements of their sites. They are responsible for such tasks as backing up and recovering the system, salvaging and scavenging, analyzing crashes, balancing disks, setting up I/O daemons and the message coordinator, metering and tuning, and maintaining system databases.

Operators are responsible for everyday central site services such as system startup and shutdown, mounting and demounting of storage devices, operation of the I/O daemons, communication with remote users, and first-level maintenance.

A detailed description of the procedures associated with each of the three user groups (administrators, maintainers, and operators) is found in the following three manuals:

System Administration Procedures (Order No. AK50)
System Maintenance Procedures (Order No. AM81)
Operator's Guide to Multics (Order No. GB61)

File No.: 1L13

This manual is designed as a reference work to be used together with one of the procedural manuals specified above. Those manuals function as "how to" documents, making reference, as needed, to the specific commands required to implement the various procedures. This manual complements the procedural manuals by providing a detailed description of the commands.

Each command description in this manual provides, minimally, the long and short name, syntax line, and function of the program. Standard headings, in the order in which they appear, when present, are as follows:

SYNTAX AS A COMMAND
SYNTAX AS AN ACTIVE FUNCTION
FUNCTION
ARGUMENTS
CONTROL ARGUMENTS
ACCESS REQUIRED
NOTES
EXAMPLES

Syntax lines give the order of required and optional arguments accepted by a command or active function. Optional portions of syntax are enclosed in braces ({}). The syntax for active functions is always shown enclosed in brackets ([]), which are required for active function use. To indicate that a command accepts more than one of a specific argument, an "s" is added to the argument name (e.g., paths, {paths}, {-control_args}).

A series of arguments enclosed in nested braces (e.g., {arg1 {arg2 {arg3}}}) indicate that, while you can omit all the arguments, if you give any one of them, you must also give the ones preceding it in the list.

Significant Changes in GB64-00A

In the following lists, the number in parentheses is the section where the command is, or was, documented.

The following commands are new; they contain no change indicators.

adddev (4)	edit_project (2)
copy_disk (9)	lock_mca (2, 4, and 9)
debug_fnp (2)	restore (9)
deldev (4)	save (9)
display_log_segment (2)	set_process_audit_flags (2)
display_proc_audit_flags (2)	unlock_mca (2, 4, 9)
edit_process_audit_flags (2)	

The following commands are not privileged; they have been removed from the manual. They belong in the Commands manual (AG92).

compare_configuration_deck (2)
print_configuration_deck (2)

The following commands were accidentally left out for MR11.0:

display_disk_label (2)
read_cards (6)

The following command is obsolete:

bos (9)

Section 10 and its 36 commands are obsolete.

CONTENTS

Section 1 Introduction 1-1
  Organization of the Commands 1-1
  Moving From One Command Environment to Another 1-2
    BCE Command Environment 1-2
    Initializer Command Environment 1-2
    User Command Environment 1-3
    Registration and Accounting Command Environment 1-5
    Daemon Command Environment 1-5
  Getting Online Help with Commands 1-5

Section 2 Privileged Multics Commands 2-1
    add_mail_table_entry 2-2
    add_volume_registration (avr) 2-4
    admin_mode_exit (ame) 2-7
    admin_util 2-7
    adopt_seg 2-11
    alarm_clock_meters 2-12
    analyze_multics (azm) 2-14
    as_who 2-45
    backup_dump 2-49
    backup_load 2-53
    bind_fnp 2-58
    before_journal_meters (bjmt) 2-59
    bootload_fs 2-63
    cache_meters 2-67
    change_kst_attributes 2-69
    change_tuning_parameters (ctp) 2-70
    change_volume_registration (cvr) 2-71
    channel_comm_meters 2-75
    charge_disk 2-80
    check_cpu_speed 2-81
    check_dir 2-82
    check_mdcs 2-82
    check_mst 2-83
    check_sst_size 2-86
    check_tc_data_size 2-87
    clean_card_pool 2-88
    clear_partition 2-89
    clear_projfile 2-90
    clear_reqfile 2-90
    clear_resource (clr) 2-91
    command_usage_count (cuc) 2-91
    compact_mail_table (salvage_mail_table) 2-94
    compare_configuration_deck 2-95
    compare_mst 2-98

compute_bill 2-98
console_report 2-100
convert_configuration_deck 2-104
copy_as_meters 2-104
copy_deadproc 2-105
copy_dump 2-106
copy_mst (cpm) 2-106
copy_registry 2-106
create_daemon_queues (cdq) 2-108
create_mail_table 2-109
create_pnt 2-109
create_urf 2-110
cv_cmf 2-111
cv_dmcf 2-112
cv_pmf 2-115
cv_prt_rqti 2-117
cv_rtmf 2-118
daily_summary 2-119
deactivate_seg 2-120
debug_fnp (db_fnp) 2-121
define_work_class (dwc) 2-122.11
delete_mail_table_entry 2-122.12
delete_old_pdds 2-123
delete_proj 2-124
delete_registry 2-124
delete_volume_registration (dvr) 2-125
deregister_resource (drr) 2-126
disk_meters 2-126
disk_queue (dq) 2-131
disk_stat_print 2-132
disk_usage_stat 2-134
disklow 2-134
display_account_status (das) 2-135
display_anst 2-138
display_aste 2-139
display_aut 2-141
display_branch 2-143
display_cache_threshold 2-143
display_cdt 2-144
display_cpu_error 2-145
display_disk_label (ddl) 2-147
display_dut 2-148.1
display_fnp_idle 2-148.3
display_ioi_data 2-150
display_kst_entry 2-152
display_log_segment 2-153
display_mstb 2-154
display_process_audit_flags 2-154.1
display_prt_rqti 2-154.2
display_psp 2-154.2
display_pvte 2-155
display_rtdt 2-156
display_syserr_log_part 2-156
display_system_audit_flags 2-157

display_vtoce 2-158
dm_lock_meters 2-161
dm_lock_status 2-163
dm_send_request 2-164
dm_set_journal_stamps 2-166
dm_set_system_dir 2-167
dm_system_shutdown 2-168
dump_firmware 2-171
dump_mpc 2-172
dump_partition 2-174
ed_installation_parms 2-175
ed_mgt 2-191
edit_process_audit_flags 2-208
edit_proj 2-208.1
edit_proj$change_all 2-208.5
eis_tester (et) 2-210
enter_lss 2-240
excerpt_mst 2-240
exercise_disk 2-241
file_system_meters (fsm) 2-243
fim_meters 2-252
fix_quota_used 2-254
flush 2-255
fnp_data_summary 2-255
fnp_throughput 2-257
format_disk_pack (fdp) 2-258
generate_mst (gm) 2-262
get_flagbox 2-270
get_uid_with_lastname 2-272
hc_pf_meters 2-272
hp_delete (hpdl) 2-274
hp_delete_acl (hpda) 2-275
hp_delete_vtoce 2-277
hp_set_acl (hpsa) 2-279
hpset_dir_ring_brackets (hpdsrb) 2-280
hpset_ring_brackets (hpsrb) 2-281
idump 2-282
inhibit_pv 2-283
install 2-283
instr_speed 2-285
interrupt_meters (intm) 2-285
io_error_summary 2-288
iod_command 2-289
iod_tables_compiler 2-290
iod_val 2-291
is_he_user 2-292
is_legal_proj 2-292
link_meters 2-293
list_as_requests 2-295
list_delegated_projects 2-295
list_extra_personids 2-296
list_mst 2-297
list_partitions 2-297
list_proc_required 2-298

list_vols 2-299
list_volume_registration (lvr) 2-304
load_ctl_status 2-305
load_mpc 2-306
lock_mca 2-307
make_commands 2-307
make_volume_labels 2-309
map355 2-311
mc_trace (mct) 2-313
mcs_version 2-325
meter_fnp_idle 2-325
meter_gate (mg) 2-326
meter_rcp 2-331
meter_signal 2-333
monitor_cache 2-335
monitor_sys_log (msl) 2-336
mos_edac_summary 2-346
move_dir_quota 2-347
move_log_segments 2-348
mpc_data_summary 2-350
ms_create (mscr) 2-351
new_user 2-352
patch_firmware 2-358
pdt_copy 2-359
poll_fnp 2-360
poll_mos_memory 2-361
poll_mpc 2-362
post_purge_meters (ppm) 2-363
print_apt_entry (pae) 2-365
print_devices 2-369
print_disk 2-370
print_iod_tables 2-370
print_line_ids 2-371
print_meters 2-371
print_pdt 2-372
print_pnt 2-373
print_projfile 2-374
print_reqfile 2-375
print_sat 2-375
print_spooling_tape 2-377
print_sys_log (psl) 2-381
print_tuning_parameters (ptp) 2-389
print_urf 2-391
priv_move_quota 2-391
process_id 2-392
proj_usage_report (pur) 2-394
read_early_dump_tape (redt) 2-396
reclassify_dir 2-396
reclassify_seg 2-397
reclassify_sys_seg 2-398
reconfigure 2-399
reconstruct_registry 2-404
record_to_sector 2-405
record_to_vtocx 2-406

register_mdir 2-407
register_resource (rgr) 2-408
reload 2-416
remove_registry 2-421
remove_user 2-421
reset_cdt_meters 2-422
reset_disk_meters 2-422
reset_soos 2-423
reset_usage 2-424
reset_use_totals 2-425
response_meters 2-426
restore_pdt_access 2-431
retrieve 2-432
ring_zero_dump (rzd) 2-437
salvage_dir 2-441
salvage_mstb 2-443
save_history_registers 2-443
sc_command 2-444
scavenge_vol 2-445
sector_to_record 2-447
send_admin_command (sac) 2-447
send_daemon_command 2-449
send_ips 2-451
send_wakeup 2-451
set_dir_quota 2-452
set_flagbox 2-453
set_log_history_dir 2-454
set_mc_message_limits 2-455
set_mos_polling_time 2-456
set_proc_required (sprq) 2-456
set_process_audit_flags 2-458
set_quota (sq) 2-458.1
set_sons_volume 2-459
set_special_password 2-459
set_system_audit_flags 2-460
set_system_console 2-462
set_system_priv 2-463
set_system_search_rules 2-465
set_timax (stm) 2-466
set_tpp (stpp) 2-467
set_work_class (swc) 2-469
set_x25_packet_threshold 2-470
sort_projfile 2-471
sort_reqfile 2-472
summarize_sys_log (ssl) 2-472
sweep 2-480
sweep_pv 2-482
sys_full_report 2-484
system_comm_meters 2-485
system_daily_report 2-487
system_link_meters 2-488
system_monthly_report 2-491
system_performance_graph (spg) 2-492
system_total 2-496

test_cpu 2-497
test_dcw 2-504
test_fnp 2-519
test_io_daemon 2-520
test_tape 2-524
total_time_meters (ttm) 2-526
traffic_control_meters (tcm) 2-530
traffic_control_queue (tcq) 2-535
tty_dump 2-539
tty_lines 2-541
tune_disk 2-543
tune_work_class (twc) 2-545
unlock_mca 2-546
unwire_pages 2-546.1
up_ctr 2-547
update_mail_table_entry 2-547
usage_and_revenue 2-549
usage_total 2-551
vtoc_buffer_meters 2-552
vtoc_pathname 2-555
vtocx_to_record 2-556
wire_pages 2-556
work_class_meters (wcm) 2-557
write_acct_bill 2-561
write_billing_summary 2-561
write_notify_test 2-562
write_user_usage_report 2-564

Section 3 Accounting Commands 3-1
add_anon 3-3
bill 3-4
chaddr 3-5
chalias 3-6
change 3-6
charge 3-9
chcpass 3-9
chdf_proj 3-10
check_log 3-11
chname 3-13
chpass 3-14
chprog 3-15
credit 3-16
cu 3-16
day 3-17
delegate 3-18
disk_report (drp, disk_auto) 3-19
dmisc 3-20
dpmf 3-20
dproj 3-21
edit_proj 3-22
install 3-24
ison 3-25
new_proj 3-26
pmf 3-35

pmisc 3-36
proj_mtd 3-38
recov 3-39
register 3-39
rename_proj 3-41
rqm 3-42
setcrank 3-43
setdisk 3-43
undelegate 3-44
upmf 3-45
who_delg 3-46

Section 4 Initializer Commands 4-1
abs 4-2
accept 4-10
add_lv (alv) 4-11
add_pdir_volume 4-12
adddev (addd) 4-13
add_vol (av) 4-13
adddev (addd) 4-14
admin 4-15
attach 4-15
bce 4-16
bump 4-17
cripple 4-19
debug 4-19
define 4-20
del_lv (dlv) 4-20
del_pdir_volume 4-21
del_vol (dv) 4-21
deldev (deld) 4-22
deroute 4-22
detach 4-23
disconnect 4-24
display_volume_log 4-25
down 4-25.1
drop 4-26
dump_mpx 4-27
exec (x) 4-27
force_reset 4-29
go 4-29
help 4-30
hmu 4-30
init_vol 4-31
intercom 4-34
list_disks (ld) 4-34
load_mpx 4-36
lock_mca 4-37
log 4-37
login (logi) 4-38
logout (logo) 4-40
maxunits (maxu) 4-41
mc_list 4-41
message (motd) 4-42

multics (mult) 4-43
no_start (ns) 4-44
preload 4-44
quit 4-45
rcp 4-45
rebuild_disk 4-47
reconfigure (rcf) 4-48
recover_volume_log 4-51
redefine 4-52
release 4-53
reload 4-54
reload_system_release 4-58
reload_volume 4-62
remove 4-65
reply (r) 4-65
reregister 4-66
reroute 4-67
reset 4-68
route 4-68
salvage_dirs (salv) 4-69
salvage_vol 4-69
set_drive_usage (sdu) 4-70
set_pdir_volumes 4-71
shift 4-72
shutdown (shut) 4-74
sign_off 4-75
sign_on 4-75
standard (stan) 4-76
start_mpx 4-76
startup (star) 4-77
stop 4-77
stop_mpx 4-78
substty 4-79
sysid 4-80
terminate (term) 4-81
unbump 4-82
undefine 4-82
unload 4-83
unlock_mca 4-84
vacate_pdir_volume 4-84
warn (w) 4-85
who 4-86
word 4-88

Section 5 Initializer Exec Commands 5-1
attended (attend) 5-2
auth 5-2
auto 5-4
cat 5-5
complete_volume (vcomp) 5-6
consolidated_volume (vcons) 5-6
copy_dump 5-7
delete_dump (dd) 5-7
deny 5-8

echoplex 5-8
end_dump 5-9
inc 5-9
incremental_volume (vinc) 5-10
io 5-11
io1 5-11
meter 5-11
print_queues (pq) 5-12
punch (puna) 5-12
punch_end (end_punch) 5-12
punch_restart 5-13
read_cards (cards, rc) 5-13
repair 5-13
reprint (rep) 5-14
reset_phcs_access (rpa) 5-15
reset_tabs (tabs, rt) 5-15
scav 5-16
set_fdump_number (set_fdump, sfdn) 5-16
set_phcs_access (spa) 5-17
unattended (unattend) 5-17
wakeup_dump 5-18

Section 6 I/O Daemon Commands 6-1
auto_queue 6-2
auto_start_delay 6-2
banner_bars 6-3
banner_type 6-3
cancel 6-4
clean_pool 6-5
copy 6-5
ctl_term 6-6
defer 6-7
defer_time 6-7
go 6-8
halt 6-8
hasp_host_operators_console (hhoc) 6-9
help 6-11
help 6-11
hold 6-11
inactive_limit 6-12
kill 6-12
list 6-13
listen 6-13
logout 6-13
logout 6-14
master 6-14
new_device 6-14
next 6-15
paper_info 6-16
pause_time 6-17
print (pr) 6-17
print_devices 6-17
print_devices 6-18
print_line_ids 6-18

prt_control 6-19
pun_control 6-20
punch 6-21
read_cards 6-21
ready 6-22
receive 6-22
reinit 6-23
release 6-23.1
req_status 6-24
request_type (rqt) 6-25
restart 6-25
restart_q 6-26
restart_status 6-27
runout_spacing 6-27
sample 6-28
sample_form 6-28
sample_hs 6-29
save 6-29
sep_cards 6-30
single 6-31
slave 6-31
slave_term 6-32
start 6-33
start 6-33
station 6-33
status 6-34
step 6-35
term 6-36
wait_status 6-36
x (xxxx) 6-37

Section 7 Volume Backup Daemon Limited Service Subsystem Commands 7-1
complete_volume_dump 7-3
consolidated_volume_dump 7-5
delete_volume_log 7-8
display_pvolog 7-8
display_volume_log 7-9
dmpr_unlock_pv 7-10
end_volume_dump 7-11
incremental_volume_dump 7-11
list_retrieval_requests (lrr) 7-14
merge_volume_log 7-16
preattach_dump_volumes 7-16
purge_volume_log 7-17
rebuild_pvolog 7-18
recover_volume_log 7-19
reload_volume 7-20
retrieve_from_volume 7-22
set_volume_log 7-24
set_volume_wakeup_interval 7-25
verify_dump_volume 7-25
volume_cross_check 7-26
volume_dump_trace_off 7-27

volume_dump_trace_on 7-27
wakeup_volume_dump 7-28

Section 8 Hierarchy Backup Daemon Limited Service Subsystem Commands 8-1
catchup_dump 8-3
complete_dump 8-8
end_dump 8-13
start_dump 8-14
wakeup_dump 8-20

Section 9 BCE Commands 9-1
alert 9-2
bce 9-2
bce_state (bces) 9-2
boot 9-3
config_edit (config) 9-5
continue (go) 9-6
copy_disk 9-7
delete (dl) 9-9
die 9-9
display_disk_label (ddl) 9-10
dump 9-12
emergency_shutdown (esd) 9-15.1
exec_com (ec) 9-15.1
fwload (fw) 9-15.2
get_flagbox (gfb) 9-16
init_files 9-17
list (ls) 9-17
list_requests (lr) 9-17
lock_mca 9-17
print (pr) 9-18
probe (pb) 9-18
qedx (qx) 9-24
reinitialize (reinit) 9-25
rename (rn) 9-26
restore 9-27
save 9-30
set_flagbox (sfb) 9-33
severity 9-34
shutdown_state (sds) 9-35
test_disk (td) 9-36
unlock_mca 9-37

Appendix A Configuration Deck Description A-1

Illustrations

Figure 1-1. Moving from One Command Environment to Another . . 1-7

Table 1. RCP Effective Access 2-414

Index i-1

SECTION 1

INTRODUCTION

ORGANIZATION OF THE COMMANDS

The commands in this document are organized into eight sections (Section 2 through Section 9), as described below.

The commands in Section 2 do not require you to be in a "special" environment. They do, however, require that you have access to the segments and directories they reference. At most sites, required access are set only for those individuals on the SysAdmin, SysMaint, or Opr projects.

The commands in Section 3 are entry points (labels) in master.ec. The registration and accounting administrator issues a limited set of entry points in master.ec. The system administrator can act in the role of registration and accounting administrator.

The commands in Section 4 can be used only when communicating with the initializer process.

The commands in Section 5 can be used only when communicating with the initializer process. These commands are also contained within admin.ec and require that you type "exec" (x) prior to typing the command.

The commands in Section 6 can be used only when communicating with the I/O daemons (i.e., the I/O daemon coordinator and drivers).

The commands in Section 7 are contained within a Honeywell-provided limited-service subsystem and are used for communicating with the Volume_Dumper.Daemon, the Volume_Reloader.Daemon, or the Volume_Retriever.Daemon.

The commands in Section 8 are contained within a Honeywell-provided limited-service subsystem and are used for communicating with the Backup.SysDaemon and the Dumper.SysDaemon.

The commands in Section 9 can be executed only within the restricted environment of the Bootload Command Environment (BCE).

*
COMMAND ENVIRONMENTS

The Multics system "handles" each command by first interpreting the request and then invoking the necessary system software components to perform the required function. The commands in this manual are organized according to the system software components (the environment) required for execution.

It is the presence of the different software components that forms the "command environment." The way in which you enter a specific command environment is described at the beginning of each command-specific section. The system software components required for command execution are automatically made available when you perform the required actions.

MOVING FROM ONE COMMAND ENVIRONMENT TO ANOTHER

In some cases you can log in in one environment and move to another environment. The Multics system includes various transitional commands that permit you to enter two or more command environments while logged in at a single session. Figure 1-1 shows the commands that you can issue to move from one command environment to another; that process is described below.

BCE Command Environment

The first activity in starting or bringing up a Multics system involves the
* loading of the BCE (Bootload Command Environment) software. While BCE is running, you can enter and execute only BCE commands.

Once the Multics operating system software is bootloaded and the initializer process is active, it is not possible, without executing the shutdown command, to
* reenter the BCE command environment.

Initializer Command Environment

The initializer process is the system control process for Multics. The initializer process is created automatically as a result of a successful Multics bootload operation.

The initializer process can be active in ring 1 or 4. If the "star" argument was
* used with the BCE boot command, then the initializer process moves directly to ring 4 without pausing in ring 1. The system is also made ready for normal user sessions (users can log in from remote terminals).

If Multics is not booted with the boot star command, the initializer is first brought to command level in ring 1. The initializer remains in ring 1 until the operator moves it to ring 4 using the standard command or the multics command. The operator must then issue the startup command or the go command to make the system available for normal user sessions (users can log in from remote terminals).

You use the initializer commands and the initializer exec commands to communicate with the initializer process. Certain initializer commands can be used only when the initializer process is operating in ring 1. Others can be used only when the initializer process is operating in ring 4. Some can be used at either time. The description of the initializer commands in Section 4 indicates when the individual command can be used. The initializer exec commands can be used only when the initializer is operating in ring 4.

The initializer process remains active as long as the system is running. In its normal mode of operation the initializer responds only to initializer commands or initializer exec commands. Sometimes, however, it is desirable to use the initializer process to execute a "normal" Multics command (i.e., any command that is not an initializer command, an initializer exec command, or a BCE command). To use the * initializer to execute a "normal" command you must enter the initializer admin mode by issuing the admin command. When in admin mode the initializer process responds to "normal" commands instead of to the initializer commands or the initializer exec commands.

Once in admin mode you can shift back to the initializer command environment by issuing the admin_mode_exit command. Alternatively, if you want to remain in admin mode but want to execute a single initializer command from time to time, you can do the following:

1. If you want to enter an initializer command (excepting the exec commands), you must use the sc_command together with the desired initializer command, as in

   ```
   sc_command force_reset
   ```

2. If you want to enter an initializer exec command, you must use the ec admin command together with the desired exec command, as in

   ```
   ec admin attended
   ```

While in the initializer command environment, you can communicate with the system daemon process(es) by the initializer reply command. You use the reply command together with the daemon driver label and the daemon command, as in

```
reply prta go
```

If you want to interrupt daemon execution while in the initializer process, you must issue the initializer quit command (and not press the ATTN or INTERRUPT key).

User Command Environment

Once the bootload/initialization sequence is complete the system becomes available for normal user sessions. Users log in at remote terminals to perform assigned work. In a normal session you can use a wide variety of commands, including the privileged commands (see Section 2) as well as the commands described in the Commands manual (AG92).

While in the user command environment, you can't use any of the "special" environment commands described in this manual; you can, however, move to those special environments by using transitional commands:

1. If you want to enter an initializer command (excepting the exec commands), you must use the send_admin_command command (which enters initializer admin mode) and the sc_command command together with the desired initializer command, as in

   ```
   sac sc_command maxu auto
   ```

2. If you want to enter an initializer exec command, you must use the send_admin_command command and the ec admin command together with the desired exec command, as in

   ```
   sac ec admin attended
   ```

3. If you want to submit a "normal" command to the initializer process, you must enter initializer admin mode by using the send_admin_command command together with the selected "normal" command, as in

   ```
   sac set_system_console  -reset
   ```

4. If you want to enter a daemon command, you can use the send_daemon_command command, as in

   ```
   send_daemon_command quit bk
   ```

   Alternatively, you can use the send_admin_command command and the sc_command command together with the initializer reply command, the daemon driver label, and the desired daemon command, as in

   ```
   sac sc_command reply prta go
   ```

5. If you are a system administrator, you can enter an accounting command by changing your working directory to >udd>SysAdmin>admin (see Section 3) and using the ec master command together with the selected accounting command, as in

   ```
   ec master bill prepare
   ```

Registration and Accounting Command Environment

A registration and accounting administrator is restricted by a special process overseer to a limited set of commands and cannot move to another command environment. If you are a system administrator, you can move into, or out of, the registration and accounting command environment by changing your working directory (see Section 3).

Daemon Command Environment

The I/O daemon command environment uses a special process overseer (iod-overseer) and a special set of search rules; you can't shift from this environment to another. The commands in the volume backup daemon command environment are contained within a Honeywell-supplied limited-service subsystem; you can't shift from this environment to another. The commands for two of the hierarchy backup daemons (Backup.SysDaemon and Dumper.SysDaemon) are also contained within a Honeywell-supplied limited-service subsystem; you can't shift from this environment to another.

GETTING ONLINE HELP WITH COMMANDS

There are information segments (info segs) for the commands in this manual in the following directories:

Commands	Directories
Commands in Section 2	>doc>privileged
Commands in Section 3	>doc>ss>accounting
Commands in Section 4 (Ring 4 commands only)	>doc>ss>operator
Commands in Section 4 (Ring 1 commands only)	>doc>ss>rl_initializer
Commands in Section 5	>doc>ss>operator
Commands in Section 6	>doc>ss>io_daemon
Commands in Section 7	>doc>privileged
Commands in Section 8	>doc>privileged
Commands in Section 9	>doc>ss>bce

*

If you're in the user command environment, the volume backup limited-service subsystem, or the hierarchy backup limited-service subsystem, you can use the user help command to look at the above info segs. When using the help command, however, you must specify the absolute pathname of the info seg, as in the following example.

```
help >doc>privileged>set_system_console
```

Alternatively, you can add one or more directories to your info search list by adding a line to your start_up.ec; for example,

```
asp info >doc>privileged
```

Then, you can look up info segs in the directories without having to specify the absolute pathname; for example,

```
help set_system_console
```

If you're in the ring 4 initializer command environment, you can look at info segs for ring 4 initializer commands by using the initializer help command; for example,

```
help down
```

If the info seg you want to look at is for an exec command, you must specify the name in the format "x.command," as in

```
help x.attended
```

If you're in the ring 1 initializer command environment, the registration and
* accounting command environment, the I/O daemon command environment, or the BCE command environment, you can't look at info segs.

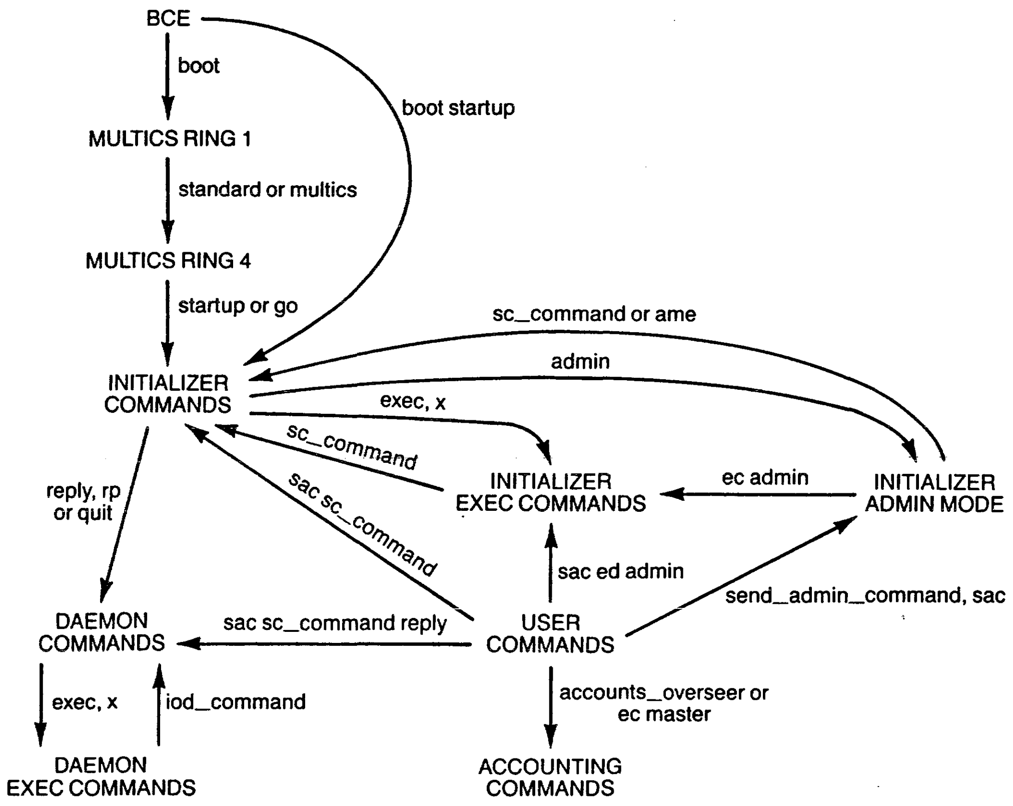

Figure 1-1. Moving From One Command Environment to Another

SECTION 2

PRIVILEGED MULTICS COMMANDS

The commands in this section do not require you to be in a "special" environment. You can use these commands while logged in at a remote terminal in a normal user session. However, these commands do perform operations on "privileged" data bases and tables and, consequently, can be used only by selected personnel (administrators, operators, and maintainers). Thus, while these commands are potentially available to all users, they can be executed only by personnel who have previously been given access to the segments and directories referenced by the commands. Before using any of these commands, you should check with your system administrator to determine if you are permitted to perform the desired operation.

Name: add_mail_table_entry

SYNTAX AS A COMMAND

```
add_mail_table_entry name {address} {-control_args}
```

FUNCTION

adds an entry to the mail table and specifies the entry's mailing address. This command is used by system administrators to specify names, usually names of users on other systems, that can be used to designate mailboxes (Person_id.Project_id), mailing lists, and Forum meetings. The name must not already exist in the mail table or the person name table (PNT). Names are not distinguished by case (e.g., Sackman and sackman are the same name).

ARGUMENTS

name
: is a string that specifies the name to be given to this mail table entry. It must be enclosed in quotation marks if it contains blank spaces.

address
: is a destination specifier, that is, a mailing address in the form used by the -mailbox or -user control arguments (see below). It can be used instead of the -mailbox or -user control arguments.

CONTROL ARGUMENTS

-acs_path path
: specifies the Access Control Segment (ACS) which controls who may update the entry's mail address. rw access to the ACS indicates that a user may update the entry. If this is not specified or is the null string, only users with access to mail_table_priv_ may update the entry.

-alias name
: specifies an alternate name for the name being defined. If this control argument is specified multiple times, all the names are defined as aliases for the entry.

-log path
: specifies the pathname of a logbox and is equivalent to:

```
    -mailbox >udd>Project_id>Person_id>Person_id.sv.mbx
```

-mailbox path, -mbx path
: specifies the pathname of a mailbox. The suffix "mbx" is added if necessary.

-mailing_list path, -mls path
: specifies the name of a mailing list. The suffix "mls" is added if necessary. The archive component pathname convention is accepted.

-meeting path, -mtg path
: specifies the pathname of a Forum meeting. The suffix "control" is added if necessary.

-save path, -sv path
: specifies the pathname of a savebox. The suffix "sv.mbx" is added if necessary.

STR -at FSystem {-via RelayN...-via Relay1}
: specifies an address on another computer system. STR identifies the user (or group of users) to receive the message and is not interpreted in any way by the local system. FSystem is the name of the foreign system where the address is located. If you give no -via, FSystem must be one of the names of a foreign system in the local system's network information table (NIT). If, however, you give -via, the foreign system name does not need to be known to the local system.

  If you specify -via, they identify an explicit route to be used to reach the foreign system. In this case Relay1 must be one of the names of a foreign system in the local system's NIT. Mail destined for this foreign address is forwarded to the system identified in Relay1. From there it is forwarded to the system identified as Relay2, and so on until it reaches the system identified as RelayN, where it is delivered to the system on which the foreign address actually resides. When the NIT is querried for either FSystem or Relay1, the query is performed in a case-insensitive manner.

-user Person_id.Project_id
: specifies the given user's default mailbox under the specified project. This control argument is equivalent to

```
-mailbox >udd>Project_id>Person_id>Person_id.mbx
```

EXAMPLES

The command line

```
add_mail_table_entry "Mary Lutyens" MLutyens.Mktg -at SystemA
```

links the name John Jones to MLutyens.Mktg on SystemA so that any time the name Mary Lutyens appears in an address used by the send_mail command, Executive Mail, or Emacs Mail, the message is sent to MLutyens.Mktg on SystemA.

The command line

```
add_mail_table_entry "Mary Lutyens" MLutyens.Mktg -at SystemA
  -alias ML
```

links the name ML, as well as Mary Lutyens, to the mailbox MLutyens.Mktg on SystemA.

To add an entry for a mailing list, you must use -mls instead of the address argument:

```
add_mail_table_entry mix -mls mixed_list
```

ACCESS REQUIRED

You must have e access to the gate mail_table_priv_.

Name: add_volume_registration, avr

SYNTAX AS A COMMAND

```
avr -control_args
```

FUNCTION

registers a new physical volume. If the specified logical volume does not exist, it is registered as well.

CONTROL ARGUMENTS

The following control arguments pertain to the registration of a physical volume.

-device_model N, -model N
: specifies the Honeywell model number for the disk device. The following values are valid for N:

VALUE	DEVICE
400	MSU0400
402	MSU0402
451	MSU0451
500	MSU0500
501	MSU0501
3380	MSU3380
3381	MSU3381

(Required)

-location STR, -loc STR
    specifies the current location of the disk pack. This is for administrative information only. STR can be any 32 characters (e.g., "offline--in cabinet 13"). (Default: "uninitialized")

-logical_volume LVNAME, -lv LV_NAME
    specifies the logical volume name. If the given logical volume is not already registered on the system, you are queried if the new logical volume should be registered. (Required)

This page intentionally left blank.

Name: admin__mode__exit, ame

SYNTAX AS A COMMAND

```
ame
```

FUNCTION

returns to initializer command level from admin mode.

Name: admin__util

SYNTAX AS A COMMAND

```
admin_util {args}
```

FUNCTION

sets and prints values in the segment sys_admin_data in the working directory. This segment is normally kept in >udd>SysAdmin>lib (with a link to it in >udd>SysAdmin>admin). It contains a number of values used by system administration procedures. One of these values is a lock, which prevents two system administrators from modifying the system administration databases simultaneously. This command also sets several values in the header of the administrator's copy of the SAT (smf.cur.sat, in the working directory). You can use the print_sat command to print information in either >sc1>sat or >udd>SysAmin>admin>smf.cur.sat.

ARGUMENTS

lock wait_time
: attempts to lock the lock in sys_admin_data. If the lock is already locked (by another process) and remains locked for more than the specified wait_time (in seconds), then an error message is printed and the program_interrupt condition is signaled. (Default: 60)

print
: prints all values in sys_admin_data.

*

set key value, default key value, dft key value
: sets the variable indicated by key to the given value. Since some of the settable variables are default values of project parameters, used when a new project is created, the words "default" and "dft" are accepted as synonyms for "set." The keys correspond to three classes of variables: installation-dependent items, project parameters, and the SAT header.

unlock
attempts to unlock the lock in sys_admin_data. If the lock was not locked by the process that is executing the command, an error message is printed. If it was locked by an existing process (other than the one executing the command), it is not unlocked.

LIST OF INSTALLATION-DEPENDENT VARIABLES

The following variables contain installation-dependent items that are printed on monthly bills and other administrative documents. Enclose the values of these items in quotes if they contain any blanks or other special characters.

user_accts XX, user_accts_office XX
where XX is the official name of the office responsible for Multics billing, for example: "Fiscal Office" or "Accounting Department." It is the first line of a return address printed on bills by the mailing_page_ subroutine. It can be up to 64 characters long.

user_accts_addr XX
where XX is the address of the above office, for example, a building and room number or a mail station. It is the second line of a return address printed on bills. It can be up to 64 characters long.

user_accts_phone XX
where XX is the phone number of the above office. It is also printed on bills. It can be up to 16 characters long.

b1 XX, b2 XX, b3 XX
where XX strings, each of which can be up to 10 characters long, are printed in large letters by the mailing_page_ subroutine, as a set, below and to the left of the address of the bill recipient. For example,

```
INTER
DEPARTMENT
MAIL
```

LIST OF PROJECT PARAMETER VARIABLES

The following variables contain default values for several project parameters. They are used by the new_proj command when the accounting administrator does not specify values for these parameters.

attributes XX
where XX, enclosed in quotes if it contains any blanks, must be acceptable to the parse_attributes_ subroutine. This string sets the default attributes for a new project. The anonymous (anon) attribute, which allows anonymous users to be registered on a project, can only be assigned by a system administrator. For a more detailed description of registration of anonymous users done by the Project administrator see the *Multics Project Administrator's* manual (AK51). The other attributes can be assigned by a project administrator if the system administrator has set them for his project. For convenience, all the valid attributes are listed below.

```
administrator       no_primary          nopreempt
admin               no_prime
                                        op_login
anonymous           no_secondary        daemon
anon                no_sec
                                        preempting
brief               no_start_up         bumping

dialok              no_warning          save_pdir
dial                nowarn
                                        save_on_disconnect
disconnect_ok       nobump              save

guaranteed_login    nolist              v_home_dir
guar                                    vhomedir
                    none
igroup              null                v_process_overseer
                                        vinitproc
multi_login
multip
```

audit XX
where XX is the default audit flags to be used when new projects are created or new users registered. For a description of the audit flags see the new_user command.

grace N
where N is the default grace time (in minutes) for a new project. The grace time specified by N is the length of time primary users retain their primary status (protected from preemption). (Default: 2880 minutes or 48 hours, which really means "never to be subject to preemption")

group XX
where XX identifies the default load control group for new projects. It can be up to eight characters long. (Default: "Other")

init_ring N
where the ring number specified by N must be a single digit from 1 to 7, inclusive. This is the default initial ring for new projects. (Default: 4)

max_ring N
  where the ring number specified by N is the default max_ring for new projects. (Default: 5)

LIST OF SAT HEADER VARIABLES

The following variables are in the header of the SAT.

uwt weight XX
  is the load control weight for the corresponding process overseer named XX, which can be up to 64 characters long. The weight must be an integer, equal to 10 times the actual weight. For example, users with the standard process overseer (process_overseer_) have a weight of 1, which is expressed as 10; users who have a process overseer that restricts them to an edit-only environment (for example, clerical personnel engaged in typing tasks) might have a weight of 0.5, which is expressed as 5. The default value of uwt depends on the process overseer, for example:

Weight	Overseer
10	process_overseer_
10	>system_library_tools>iod_overseer_
10	>system_library_tools>card_overseer_
5	>limited_service_system

  You can give the word "delete" (dl) instead of weight to delete the process overseer XX from the list. The maximum number of unit weight table entries is 24.

administrator1 User_id
administrator2 User_id
  These two items are the User_ids of two persons who are permitted to act as system administrators. They may be specific people or, more general, User_ids. For example,

```
JBSmith.*
*.SysAdmin
```

  permits anyone on the SysAdmin project, plus JBSmith when logged in on any project, to act as a system administrator.

max_units N
: The N is the load limit, which is 10 times the actual user weights. For example, if the system can handle 80 users of weight 1, the N should be 800. The default value of max_units is 750.

  This limit is usually overridden by the per-shift values from the configuration table in the installation_parms segment. However, when automatic load control is disabled (in a special session, for example), the max_units figure in the SAT header is used. *

Name: adopt_seg

SYNTAX AS A COMMAND

```
adopt_seg pvname vtocx path
```

FUNCTION

reconstructs a directory branch for a segment or directory whose branch has been lost but whose VTOCE exists. Individual segments may be adopted by the use of the adopt_seg command.

ARGUMENTS

pvname
: is the physical volume name of a mounted physical volume, on which the VTOCE for a branch that is to be constructed exists.

vtocx
: is the VTOC index, in octal, of that VTOCE.

path
: is the pathname of the entry to be constructed for the segment or directory. The directory portion of this pathname must specify an existent directory.

ACCESS REQUIRED

Use of the adopt_seg command requires access to the phcs_ and hc_backup_ gates.

NOTES

Segments whose branches are to be constructed by adopt_seg can only be returned to their original position in the hierarchy; the superior directory must exist.

The new branch that is created is lacking certain attributes, these attributes can be reconstructed either manually or by use of the rebuild_dir command. Default values are supplied for the following attributes:

ACL	null
name	entry portion of path
ring brackets	validation level of caller
safety switch	off
copy switch	off
access class	from VTOCE
bit count	36854*current length as determined from VTOCE
TPD switch	off

The fix_quota_used command and exec_com have to be used on directories that adopt segments in this way to correct quota used totals. This is always necessary if quota has been corrected since the branch was first lost.

Name: alarm_clock_meters

SYNTAX AS A COMMAND

```
alarm_clock_meters {-control_arg}
```

FUNCTION

displays information about the behavior of the simulated alarm clock used within the Multics system.

CONTROL ARGUMENTS

-report_reset, -rr
    generates a report and then performs the reset operation.

-reset, -rs
    resets the metering interval for the invoking process so that the interval begins at the last call with -reset specified. If -reset has never been given in a process, it is equivalent to having been specified at process initialization time.

ACCESS REQUIRED

This command requires access to phcs_ or metering_gate_.

NOTES

If the alarm_clock_meters command is given with no control argument, it prints a full report.

The following are brief descriptions of the variables printed out by alarm_clock_meters.

No. alarm clock sims
: is the total number of times that an alarm clock wakeup was generated.

Simulation lag
: is the average value of the simulated clock delays (i.e., the time difference between when the alarm should have gone off and when it was actually processed).

Max. lag
: is the largest of the simulated delays that has occurred since the time of bootload. This value is not affected by the -reset control argument.

Priority boosts
: is the number of times the alarm clock went off indicating a priority scheduling process on the ready list should be granted high priority, i.e., have its waiting time before rescheduling set to 0. This variable is not printed if zero.

Priority elig. lost
: is the number of times the alarm clock went off indicating a priority process that had been running lost its eligibility because it had used up its eligible time. This variable is not printed if zero.

EXAMPLES

The following is an example of the information printed when the alarm_clock_meters command is invoked with no control argument.

```
Total metering time 1:36:35

No. alarm clock sims.   2274
Simulation lag            91.676 msecs.
Max. lag                2305.883 msecs.
Priority boosts            6
```

Name: **analyze_multics, azm**

SYNTAX AS A COMMAND

```
azm {-control_args}
```

FUNCTION

invokes a subsystem that aids in system crash analysis. It can analyze dumps created by the bootload command environment (BCE) dump command and copied into >dumps in the Multics hierarchy by the copy_dump command. It can also analyze processes saved by the answering service after a fatal process error and copied by the copy_deadproc command.

CONTROL ARGUMENTS

-abbrev, -ab
: enables abbreviation expansion of request lines.

-no_abbrev, -nab
: does not enable abbreviation expansion of request lines. This is the default.

-no_prompt
: suppresses the prompt for request lines in the request loop.

-no_start_up, -nsu
: does not execute any startup exec_com. This is the default.

-profile PATH, -pf PATH
: specifies the pathname of the profile to use for abbreviation expansion. The suffix "profile" is added if necessary. This control argument implies -abbrev.

-prompt STR
: sets the request loop prompt to STR. The default is the ioa_ STR:
```
^/azm^[ (^d)^]:^2x
```

-request STR, -rq STR
: executes STR as an azm request line before entering the request loop.

-start_up, -su
: executes the exec_com "start_up.azmec" upon invocation of azm. This start_up exec_com is first searched for in your home directory, then in your project directory (>udd>Project_id), and last in >site. The first exec_com found is used.

-quit
: exits azm after execution of other arguments. Can be used with -request.

NOTES

This command uses the standard search list mechanism to locate dumps. If it does not find a "dumps" search list, it creates one, placing >dumps and >dumps>SAVE_pdrs in the search list as the default. If additional search paths are desired, the add_search_path command can be used to define them.

VIRTUAL ADDRESS CONSTRUCTS

Accessing data requires some pointer value to define an address space. The generation of the pointer value is performed by resolving a virtual address (virtual-addr). A virtual-addr consists of two parts: a segment number and a word offset.

The command resolves virtual-addrs from the following types of information:

Symbol:
: is a symbolic name for a segment number and an offset that can be resolved by data in definitions_ (i.e., sst$ptl can be resolved to the correct segment number and offset of the page table lock).

Segment name:
: a segment name can be resolved in many ways, but it can only provide one part of the virtual address: azm uses 0 as the default offset for this pointer value (i.e., tc_data is resolved to SEGNO|0).

Segment number:
: a segment number needs no resolution, but a default action needs to be taken for the offset (the default is 0, i.e., SEGNO|0).

Segment name/number and offset:
: the virtual-addr in this case can be a segment name or segment number and an octal offset (i.e., the construct of pds|20 is translated to SEGNO|20 or dseg|5 is 0|5). The notation "|" and "$" must be used without spaces (e.g., 244|0 or sst$cmp).

Temporary pointers:
: azm keeps a set of 11 temporary pointers per translation. (See the set request for a list of these pointers.) A translation is one complete entity such as a "dump". These pointers can be set with the set request (e.g., set sp 230|100). They can be referenced by other requests as another type of "symbol" in a virtual-addr expression, after they have been set. If not set, these pointers are null.

Offset operators:
: the operators "+N" and "-N" immediately preceding an octal number or virtual-addr construct can be used to alter the offset of a virtual address. N is a number interpreted in octal. No spaces are allowed between the operator and the N. For example, sst$ptl +30 are resolved to be the SEGNO for sst_seg with the offset of ptl plus 30 octal locations; sst$ptl+30 is also valid.

Indirection:

a virtual-addr can imply indirection. The indirect word can be used as an ITS pair if it is a valid ITS word pair; if not, the upper half of the word is used. The following virtual-addr construct is used to specify indirection (sst$cmp,*). The format of an indirect pointer value is:

```
segno|offset,*  segname|offset,*  symbol,*
temp_ptr,*      temp_ptr|offset,*
```

EXAMPLES OF INDIRECTION

```
17|230,*  sst|230,*  sst$cmp,*+2
sp,*      sp|230,*
```

LIST OF REQUESTS

absolute_address, absadr

SYNTAX

```
absadr virtual-addr
```

SYNTAX AS AN ACTIVE REQUEST

```
[absadr virtual-addr]
```

FUNCTION

translates a "virtual address" to an absolute memory address.

ARGUMENTS

virtual-addr
: can be a segment number, name, or symbolic address (e.g., 64, prds, prds$am_data). See "Virtual Address Constructs" above.

ACTIVE REQUEST EXAMPLE

```
!  display_absolute [absadr sst$cmp] 2
```

displays the first two words of the absolute address of sst$cmp.

add_request_table, arqt

SYNTAX

```
arqt path
```

FUNCTION

adds a user-defined request table in the list of request tables being searched by the current azm invocation.

ARGUMENTS

path
: is the pathname of the request table to be added. This request table must be consistent for use with the subsystem utility. (Refer to the *Multics Programmer's Reference Manual,* Order No. AG91, for request table structure.)

apply, ap

SYNTAX

```
ap virtual-addr {range} command_line
```

FUNCTION

extracts all or part of a segment specified by VIRTUAL-ADDR from the selected dump, and places a copy in a temporary segment. This pathname is passed as the last argument in the command_line.

ARGUMENTS

virtual-addr
: may be a segment number, name or symbolic address (e.g., 64, prds, prds$am_data). See "Virtual Address Constructs" above.

range
: specifies the number of words in octal to be copied. The default is the entire segment.

command_line
: is any command.

NOTES

The offset in the virtual address specifies where the copying of the segment begins. When only part of a segment is extracted, it goes at the beginning of the temporary segment. For example:

```
   ap pds$am_data 400 dump_segment
```

puts 256 (decimal) words at the beginning of the segment.

apte

SYNTAX

```
apte {proc_indicator} {-control_args}
```

FUNCTION

displays active page table entry (APTE) information for processes in a dump whose states match the states specified.

ARGUMENTS

proc_indicator
  used for specifying individual processes. It can take one of three forms:

  - The decimal index (starting at zero) of a process in the dump.
  - The octal APTE offset of the process.
  - The octal process_id of the process.

CONTROL ARGUMENTS

-all, -a
  displays APTE information for all processes in any state. This is the default.

-blocked, -blk
  displays APTE information for all processes in the blocked state.

-count, -ct
  specifies the total number of processes meeting the criteria specified by control_args. With -all, it gives the counts of each process state.

-current, -cur
  displays APTE information for the current process.

-page_tbl_lock, -ptl
  displays APTE information for all processes marked as page table locking.

-ready, -rdy
: displays APTE information for all processes in the ready state.

-run
: displays APTE information for all processes in the running state.

-stopped, -stop
: displays APTE information for all processes in the stopped state.

-wait
: displays APTE information for all processes in the waiting state.

EXAMPLES

```
   apte 2
```

displays information for process 2 in a dump.

```
   apte 10600
```

displays information for the process with APTE offset 10600 (octal).

```
   apte 3500555555
```

displays information for the process with octal process_id 003500555555.

associative_memory, am

SYNTAX

```
am {-control_args}
```

FUNCTION

displays SDW and/or PTW associative memories.

LOCATION CONTROL ARGUMENTS

-dump
: displays the "dump" associative memories from the bootload CPU at the time the dump was taken. This is the default.

-prds
: displays associative memories that have been stored in the prds of the processor on which the current process is running.

CONTROL ARGUMENTS

-all, -a
: displays all entries in the associative memories. The default is to display only those entries that are valid (i.e., the full bit is on).

-ptw
: displays only the PTW associative memories.

-pageno PAGENO
: displays only those entries in the PTW associative memories that have a page number that matches the value of PAGENO (which is an octal page number).

-sdw
: displays only the SDW associative memories.

-segno SEGNO
: displays only those entries in the SDW and PTW associative memories that have a segment number that matches the value of SEGNO, which is an octal segment number. (See assoc_mem.incl.pl1.)

NOTES

If no control arguments are given, both the SDW and PTW associative memories are displayed for the "dump" associative memories.

aste

SYNTAX

```
aste segno/segname {-control_args}
```

FUNCTION

displays active segment table (AST), page table, and trailer information. The default displays active segment table entry (ASTE) and page table information only.

ARGUMENTS

segno/segname
: is the segment number or segment name of interest.

CONTROL ARGUMENTS

-aste
: displays AST information for the selected entry.

-at offset, -at virtual-addr
: displays ASTE information starting at the offset or virtual address specified.

-brief, -bf
    displays everything excluding the page table information.

-long, -lg
    displays everything, that is, the ASTE, page table, and trailer information.

-page_table, -pt
    displays page table information for the selected segment.

-trailer, -tr
    displays trailer information about the selected segment.

configuration_deck, cd

SYNTAX

```
cd {card_names} {-control_args}
```

FUNCTION

displays the contents of the configuration deck in the selected dump. This request works exactly like the standard pcd command, except that it gets the configuration deck from the dump.

ARGUMENTS

card_names
    are the names of the particular configuration cards to be displayed. Up to 32 card names can be specified (separated by spaces). If no card names are given, the complete configuration deck is printed.

CONTROL ARGUMENTS

-brief, -bf
    suppresses the error message when a requested card name is not found. This is the default.

-exclude FIELD_SPECIFIERS, -ex FIELD_SPECIFIERS
    excludes particular cards or card types from being displayed. One to 14 field specifiers can be supplied with each -exclude, and up to 16 -exclude control arguments can be specified. To be eligible for exclusion, a card must contain fields that match all field specifiers supplied with any -exclude argument.

-long, -lg
    prints an error message when a requested card name is not found.

-match FIELD_SPECIFIERS
: selects particular cards or card types to be displayed. One to 14 field specifiers can be supplied with each -match, and up to 16 -match control arguments can be specified. To be eligible for selection, a card must contain fields that match all field specifiers supplied with any -match argument.

NOTES

Field specifiers can consist of a complete card field or a partial field and an asterisk (*). An asterisk matches any part of any field. Specifiers for numeric fields can be given in octal or decimal, but if decimal they must contain a decimal point. Asterisks cannot be specified in numeric field specifiers. All numeric field specifiers are converted to decimal and matched against numeric card fields, which are also converted to decimal. Hence, the field specifier "1024." matches a card containing the octal field 2000, and the field specifier "1000" matches a card containing the decimal field 512. Note that all card names must be specified before the first -match or -exclude argument. Field specifiers following a -match or -exclude argument include all arguments until the next -match or -exclude argument.

display, d

SYNTAX

```
d virtual-addr {exp} {range} {-control_args}
```

SYNTAX AS AN ACTIVE REQUEST

```
[d virtual-addr {exp} {range} {-control_args}]
```

FUNCTION

displays a selected portion of a segment in a dump or a saved process.

ARGUMENTS

virtual-addr
: specifies the initial offset of the virtual address space to be dumped. May be a segment number, name, or symbolic address (e.g., 64, prds, prds$am_data). See "Virtual Address Constructs" above.

exp
: is an expression, which is either an octal value or a virtual-addr construct yielding an octal value. This value can be positive or negative, specified by the plus or minus sign.

range
: specifies the number of words to be dumped in octal. If a range is not specified, the default action is to display one word. If the data is an ITS pair, two words are displayed.

MODE SPECIFICATION CONTROL ARGUMENTS

-character, -ch, -ascii
: displays the selected number of characters in ASCII. Characters that cannot be printed are represented as periods. As an active request, -character returns the character representation of the requested address.

-instruction, -inst
: displays the selected number of words as instructions. Read-only segments that are not part of the dump are located using the hardcore search paths. The default directory for hardcore search paths is >ldd>hard>e. However, sites that have modified hardcore can place the directory containing modified bound segments before >ldd>hard>e in their hardcore search paths. Usage as an active request is not allowed.

-octal, -oc
: displays the selected number of characters in octal. When used as an active request, it returns the octal value of the requested address. This is the default.

-ptr, -p
: displays the selected number of word pairs as pointers. When used as an active request, it returns the octal value in the form SEGNO|OFFSET.

-pptr, -pp
: displays the selected number of words as a packed pointer. When used as an active request, it returns the octal value in the form SEGNO|OFFSET.

-pptrx, -ppx
: displays the selected number of words as packed pointers and expands the SEGNO|OFFSET to a segment name. Usage as an active request is not allowed.

-ptrx, -px
: displays the selected number of word pairs as pointers and expands the SEGNO|OFFSET to a segment name. Usage as an active request is not allowed.

CONTROL ARGUMENTS

-as STRUCTURE_NAME
: displays the data as a PL/I structure defined by STRUCTURE_NAME. The STRUCTURE_NAME is an inner ring system-defined include file. See the info file structure_name.info for a list of available structures and their short names. The address given in the display request is taken as the address of the beginning of the structure. If the whole structure is being displayed, that is the address where display begins. If only certain elements are being displayed, that is the address used to compute offsets of the elements. The structure reference following -as must be a single string, containing no spaces, and must follow the syntax described below. The single string is used to specify structure elements, array indexes, and substring matching. Usage as an active request is not allowed.

-long, -lg
: displays each element of the structure on a separate line. This control argument is only implemented with -as.

STRUCTURE SYNTAX

The structure reference is made up of two parts: a structure element reference and an optional set of match strings. If no match strings are supplied, no string matching is done. The structure element reference syntax consists of one or more element names, separated by periods, and may contain subscripts following some of these element names. The first name in a structure element reference must be a level-one structure reference; partially qualified top-level references are not permitted. Intermediate levels of qualification may be omitted as long as there is no ambiguity.

All subscripts must be supplied as decimal integers. The subscripts can be cross-section references such as "(1:4)" to reference elements one through four. Asterisk bounds cannot be used: if a cross section is desired, its upper and lower bounds must be given as decimal constants. If an element has more subscripts than are supplied, the complete cross section is printed for the remaining subscripts. To eliminate the need for quoting, subscripts may be surrounded by braces instead of parentheses.

To specify that only certain elements be displayed (such as all those with names containing the string "time"), a set of match strings can be given after the structure element reference. Each match string begins with a slash and is followed by the string itself. The final match string can be followed by a slash, but this is not required. If match strings are specified, any element that matches at least one string is displayed.

EXAMPLES OF STRUCTURE REFERENCES

pvt
: the whole structure "pvt".

pvt.n_entries
: the single element "n_entries" in the structure "pvt".

sst/time/, sst/time
: any elements in the structure "sst" containing the string "time". Note that the final slash is optional.

sst/time/meter/
: any elements in the structure "sst" containing either the string "time" or the string "meter".

sst.space{3}
: element three of "sst.space".

sst.space{2:4}
    elements two, three, and four of "sst.space".

sst.space
    all elements of "sst.space".

sst.level{1}
    both elements of the "level" array for "sst.level{1}"

This page intentionally left blank.

sst.level{1}.ausedp, sst.level.ausedp{1}
the single element "ausedp" of the "level" array for "sst.level{1}"

STRUCTURE OUTPUT FORMAT

The default output format is a compressed form, which places as many values on a line as will fit within the line length. The -long control argument places one value on a line. The short form, additionally, collects all bit(1) flags and displays them, at the end of the display for each substructure or array element, in two groups: one listing all the flags that were on ("1"b) and one for all the ones that were off ("0"b).

All PL/I data types are displayed in the same representations used by probe. Additionally, the following special formats are used:

1. Bit strings are displayed in octal if the length is divisible by three, in hex if divisible by four, and as bit strings otherwise.

2. Character strings are displayed as a string concatenated with a repeated constant if the string is padded on the right with more than 16 nulls, spaces, or octal 777 characters.

3. Large-precision (> 51) fixed binary values are also displayed as clock readings if their values represent clock readings within 10 years of the present.

DISPLAY REQUEST EXAMPLES

```
d 75|560 2
```

displays the two words in seg number 75 starting at offset 560.

```
d pds|560 2
```

displays the two words in the segment named pds starting at offset 560.

```
d pds$trace
```

displays one word in the pds segment beginning at the offset specified by $trace.

```
display 244|260 +20 4
```

displays four words of segment number 244 starting at offset 300 octal.

```
d sp 20
```

displays 20 octal words starting with the segment offset defined in the azm internal temporary pointer (see set request).

```
    d sst$cmp,* +sst$cmesize sst$strsize
```

causes the word at sst$cmp to be used as an indirect word, or an indirect pointer if the resultant address has ITS modification, to develop the starting virtual address. The value derived from sst$cmesize is then added to the starting offset for the "final" starting address. The range, or number of words to be displayed, is specified by the value contained in sst$strsize.

```
    d sst|2 -as apte
```

displays the APTE entry at the given offset in the SST as it is defined by apte.incl.pl1.

display_absolute, da

SYNTAX

```
da abs-addr {range} {-control_args}
```

SYNTAX AS AN ACTIVE REQUEST

```
[da abs-addr {range} {-control_args}]
```

FUNCTION

dumps an absolute memory address space in the dump.

ARGUMENTS

abs-addr
: is the starting absolute memory address, in octal.

range
: specifies the number of words to be dumped in octal. If a range is not specified, the default is one word. If the data to be dumped is an ITS pair, two words are dumped.

MODE SPECIFICATION CONTROL ARGUMENTS

For a description of the mode specifications, see the display request.

```
    -character, -ch, -ascii
    -instruction, -inst
    -octal, -oc
    -ptr, -p
    -pptr, -pp
    -pptrx, -ppx
    -ptrx, -px
```

events, ev

SYNTAX

```
ev {-control_args}
```

FUNCTION

displays significant events, in reverse chronological order, from a dump (see "Notes").

CONTROL ARGUMENTS

-last N, -lt N
    specifies the number of events to print. The default is to print 10 events.

-long, -lg
    displays disk queue events.

-time NSECS, -tm NSECS
    specifies the time in seconds before the dump was taken when events were significant. The default is 10 seconds.

NOTES

The following events are considered as significant: machine conditions (from BCE, prds, pds, and the mc_trace_buf), traffic control state change time, system error messages, Fim frames in any stack, and connects by device and disk queues (long report only). If neither -time nor -last are specified, the default action is equivalent to "ev -time 10".

history_regs, hregs

SYNTAX

```
hregs {hregs_specifier} {-control_args}
```

FUNCTION

displays a composite analysis or octal dump of the processor history registers. This request is useful for people who are knowledgable of the hardware. The default action is to display the AU, CU, DU, and OU history registers for the pds in a threaded order and interpreted format.

ARGUMENTS

hregs_specifier
    may be chosen from the following:

    -condition virtual-addr, -cond virtual-addr
        displays history registers from a condition frame, the location of which is described by virtual-addr.

    -dump
        displays the "dump" history registers from the bootload CPU at the time the dump was taken.

    -pds
        displays the history registers that have been stored in the current processes pds. This is the default.

    virtual-addr
        displays the history registers that have been stored at the address space specified by virtual-addr.

CONTROL ARGUMENTS

-au
    displays the AU history registers only.

-cu
    displays the CU history registers only.

-du
    displays the DU history registers only.

-ou
    specifies that only the OU history registers are to be displayed.

-interpret
    displays the interpreted form of the history registers only (default), or, if -octal is given, includes the octal representation also.

-octal, -oc
    displays the octal values of history registers only, or, if -interpret is also selected, displays octal and interpreted form. If neither -octal nor -interpret is specified, the default action is to display the interpreted form only.

-thread
    displays the selected history registers in the "correct" order. This is the default.

-no_thread
    displays the selected history registers in serial order, without attempting to sort them.

list_dumps, lsd

SYNTAX

```
lsd {path} {-control_args}
```

FUNCTION

lists the dumps and/or the dead processes in the selected dump directory. If path is not given, it lists all dumps and/or dead processes in the dump directories specified in the dumps search list.

ARGUMENTS

path
    is the pathname of the dump directory to be checked. Starnames are acceptable.

CONTROL ARGUMENTS

-deadproc, -dp
    specifies that only dead processes are to be listed. If path is not given, it checks all dump directories specified in the dumps search list for dead processes.

-fdump, -fd
    specifies that only dumps are to be listed. If path is not given, it checks all dump directories specified in the dumps search list for dumps. This is the default.

NOTES

If no arguments are given, the default is to list only dumps.

Analyze_multics will use the search_list_defaults_ mechanism to locate and set up the dumps search list, which, by default, will contain the directories >dumps and >dumps>save_pdirs.

list_processes, lsp

SYNTAX

```
lsp {proc_indicator} {-control_args}
```

SYNTAX AS AN ACTIVE REQUEST

```
[lsp {proc_indicator} {-control_args}]
```

FUNCTION

lists all known processes in the selected dump. As an active request, it returns the process_ids meeting the control argument criteria, otherwise it returns a null string. If -count is specified only the total is returned.

ARGUMENTS

proc_indicator
    used for specifying individual processes. It can take one of three forms:

```
- The decimal index (starting at zero) of a process
  in the dump.
- The octal APTE offset of the process.
- The octal process_id of the process.
```

CONTROL ARGUMENTS

-all, -a
    lists all processes in the dump. This is the default.

-blocked, -blk
    lists processes marked as blocked.

-count, -ct
    counts all processes. With -all, it gives the counts of each process state.

-current, -cur
    lists the current process.

-page_tbl_lock, -ptl
    lists processes marked as page table locking.

-ready, -rdy
    lists processes marked as ready.

-run
    lists processes marked as running.

-stopped, -stop
    lists processes marked as stopped.

-wait
    lists processes marked as waiting.

EXAMPLES

```
!  do "select_process &1;sdw 0" ([list_processes])
```

displays the SDW for DSEG for all processes in the dump.

machine_conditions, mc

SYNTAX

```
mc {mc_specifier} {-control_args}
```

FUNCTION

displays all or parts of machine conditions based on the given pointer.

ARGUMENTS

mc_specifier
  may be chosen from the following:

  -dump
    specifies the dump for the bootload CPU registers at the time of the dump.

  -pds {STR1}
    where STR1 can be "all", "fim" ("fim_data"), "page_fault" ("pgf", "page_fault_data"), "signaller" ("signal", "sig", "signal_data"). It defaults to "all" if STR1 is not given.

  -prds {STR2}
    where STR2 can be "all", "fim" ("fim_data"), "interrupt" ("int", "interrupt_data"), "system_trouble" ("sys", "sys_trouble_data"). It defaults to "all" if STR2 is not given.

  virtual-addr
    is the virtual address construct used to define the address space containing machine conditions.

    The virtual address can point directly to the machine conditions or to the frame that contains the machine conditions. In the latter case, the offset is calculated for the user.

CONTROL ARGUMENTS

-eis
  displays the EIS pointers and lengths (interpreted).

-faults, -flt
  displays the fault register.

-long, -lg
  displays all elements of the MC.

-mc_err
  displays the mc_err data word.

-misc
displays the miscellaneous data (i.e., mc_err, fault reg, time).

-octal, -oc
displays the eis info, SCU data, or pointer registers, in octal. This control argument is used with -scu, -eis, or -regs.

-pointers {PR_LIST}, -prs {PR_LIST}
displays pointer registers selected by PR_LIST (from 0 to 7, separated by spaces). If PR_LIST is not specified, all the pointers are displayed.

-ppr
displays only the PSR and IC from the MC.

-registers {REG_LIST}, -regs {REG_LIST}
displays only the basic OU registers. REG_LIST can be any of the following:

```
x0 x1 x2 x3 x4 x5 x6 x7 a q all.
```

If REG_LIST is not specified, all of the basic OU registers are displayed.

-scu
displays only the SCU data of the MC.

-time, -tm
displays the MC time.

-tpr
displays only the TSR and the CA from the MC.

NOTES

If no MC specifiers are given, the temporary pointer prmc is used. The default * control arguments are -fault, -mc_err, -pointers, -scu, -time, and -tpr. The machine_conditions request sets all azm-defined temporary pointers as seen in the machine_condition frame.

EXAMPLES

```
mc -pds fim -scu
```

displays the SCU data found in the fim frame of the PDS currently being referenced in the dump.

page__trace, pgt

SYNTAX

```
pgt {-control_args}
```

FUNCTION

displays the contents of the page trace table in the current process data segment (PDS). The default is to display the last 15 trace entries. Trace entries are always displayed in reverse chronological order.

CONTROL ARGUMENTS

-all, -a
displays all trace entries.

-last N, -lt N
specifies the number of trace entries, where N is a positive decimal integer, to be displayed.

replace, rp

SYNTAX

```
rp segno/segname path
```

FUNCTION

replaces the segment designated by segno/segname in the current translation table with another segment designated by path.

ARGUMENTS

segno/segname
the segment number or segment name within the translation table to be replaced.

path
is the pathname of the segment. The equal convention can be used. For example:

```
rp bound_system_faults [e wd]>=.new
```

NOTES

Both per-process and per-system segments can be replaced. For example, if the PDS is replaced in a process, it affects only the current process; whereas if tc_data is replaced in a process, it affects the whole dump.

scus

SYNTAX

```
scus
```

FUNCTION

prints the memory address space (in octal) of each SCU from the registers saved in the dump.

sdw

SYNTAX

```
sdw {segno/name} {segno/name}
```

FUNCTION

displays the SDWs in the current process' DSEG.

ARGUMENTS

segno/name
: is the segment number or name of interest. The first is the starting segment number and the second is the ending segment number. If only one is given, only one is displayed; if none are given, all are displayed.

search, srh

SYNTAX

```
srh virtual-addr {range} search_string
```

SYNTAX AS AN ACTIVE REQUEST

```
[srh virtual-addr {range} search_string]
```

FUNCTION

searches a segment starting at virtual-addr matching on search_string. The search is performed on a 36-bit-word boundary. As an active request, the virtual addresses matching the criteria specified are returned.

ARGUMENTS

virtual-addr
: is the pointer to the address space to search.

range
specifies the number of words to be searched from the starting offset, where range is an octal value. The default is the rest of segment. The search is started from virtual-addr.

search_string
is a 12-character string representing the 12 octal digits that make up a machine word (36 bits. 3 bits per digit). This forms both the search data and search mask by using the hyphen (-) as a "don't care character" in the string. The "do care digits" are octal "from 0 to 7." Any other character is illegal.

EXAMPLES

To search for

1. all words in segment 76 that have the last two digits of 43:

```
srh 76 ----------43
```

2. all words in tc_data where the upper half = 070707:

```
srh tc_data 070707------
```

3. words that end in 1234 in sst_seg starting at 1000, but only searching for 200 octal words:

```
srh sst_seg|1000 200 --------1234
```

4. words that start with 45 and end with 77, starting at sst_seg$ptl for 100 words:

```
srh sst_seg$ptl 100 45--------77
```

segment_name, name

SYNTAX

```
name arguments
```

SYNTAX AS AN ACTIVE REQUEST

```
[name arguments]
```

FUNCTION

prints the segment name given either a virtual address or a segment number.

ARGUMENTS

virtual-addr
  is the virtual address construct used to define the segment.

number
  is the segment number of the segment to be referenced. Thus, "name 230" returns the name associated with the segment number 230, which (in MR11.0) is "stack_0".

segment_number, number

SYNTAX

```
number arguments
```

SYNTAX AS AN ACTIVE REQUEST

```
[number arguments]
```

FUNCTION

prints the segment number given either a virtual address or a segment name.

ARGUMENTS

virtual-addr
  is the virtual address construct used to define the segment.

name
  is the name of a segment, e.g., stack_0. Thus, "number sst_seg" returns the segment number associated with the segment sst_seg.

select_deadproc, sldp

SYNTAX

```
sldp {name}
```

FUNCTION

selects and translates a dead process which has been copied into the Multics hierarchy via the copy_deadproc tool. The dead process is located by means of the dumps search list. By default, copied dead processes are found in >dumps>save_pdirs.

ARGUMENTS

name
: is the pathname of the process directory of interest. This can be a relative or absolute pathname. The dead process directory name is of the form person.pdir or person.N.pdir, where N is a numeric number. N=1 for the most recently copied dead process. The suffix "pdir" is assumed if not given.

NOTES

When sldp is invoked with no arguments, it prints an identifying message.

select_dump, sld

SYNTAX

```
sld {name} {-control_args}
```

FUNCTION

selects and translates a dump of a system crash. The dump is found via the dumps search list. *

ARGUMENTS

name
: is the ERF number or the pathname of the zero component of the dump. It can also be the form path>35, where 35 is the ERF number. Several control arguments are also acceptable if name is not specified.

CONTROL ARGUMENTS

-first, -ft
: selects the first dump (by ERF number) in the dump directory found via the dumps search list.

-last, -lt
: selects the last (most current) dump in the dump directory according to ERF number.

-next, -nx
: selects the next dump in the dump directory. This is relative to the dump currently being looked at.

-previous, -prev
: selects the previous dump in the dump directory. This is relative to the dump currently being looked at.

NOTES

The sld command attempts to select the process as indicated by scs$trouble_processid. If this cannot be done, the default is the first running process found in the dump.

select_process, slp

SYNTAX

```
slp {proc_indicator} {-control_args}
```

FUNCTION

selects a process for examination. When invoked with no arguments, the current process is listed.

ARGUMENTS

proc_indicator
: used for specifying individual processes. It can take one of three forms:

```
- The decimal index (starting at zero) of a process in the dump.
- The octal APTE offset of the process.
- The octal process_id of the process.
```

CONTROL ARGUMENTS

-brief, -bf
: suppresses the message about changing processes.

-cpu TAG
: selects the DBR for the process running on the CPU identified by TAG (where TAG is one character in the range a through h).

-dbr dbr_value
: selects the process defined by the dbr_value.

-long, -lg
: prints a message announcing the new process selected. This is the default.

set

SYNTAX

```
set ptr_n virtual-addr
```

FUNCTION

sets an internal temporary pointer like a CPU pointer register (i.e., "pr6" or "sp"). These pointers can then be used as a virtual-addr by other azm requests.

ARGUMENTS

ptr_n
can be either the name or number of a "temporary pointer."

There are eight temporary pointers and two special-case pointers:

NUMBER	NAME	NUMBER	NAME
pr0	ap	pr4	lp
pr1	ab	pr5	lb
pr2	bp	pr6	sp
pr3	bb	pr7	sb

prmc	intended to be a pointer to the current MCs.
prfr	intended to be a pointer to the current stack frame.

virtual-addr
can be a segment number, name, or symbolic address (e.g., 64. prds, prds$am_data).

EXAMPLES

```
set pr6 240|100
```

sets a temporary ptr named pr6 (sp).

```
set sb 240
```

sets the temporary ptr (sb) to the base of seg 240 (240|0).

NOTES

The value of a temporary pointer can be displayed via the value request:

```
v {ptrn | -all}
```

stack, sk

SYNTAX

```
sk virtual-addr {-control_args}
```

FUNCTION

traces a given stack.

ARGUMENTS

virtual-addr
: is the virtual address construct defining the stack to be traced.

CONTROL ARGUMENTS

-arguments, -ag
: prints the arguments for the stack frames traced. Analyze_multics (azm) does not interpret the descriptors in an argument list passed to an internal procedure declared options (variable) because azm has no knowledge about arguments for internal procedures.

-for N
: traces for N stack frames. If no valid stack frames exist (stack_begin_ptr = stack_end_ptr), a -force must be used.

-force, -fc
: forces a forward stack trace. This should be used when there are no valid frames for this stack (stack_begin_ptr = stack_end_ptr).

-forward, -fwd
: traces in a forward manner.

-long, -lg
: prints the arguments and an octal dump of the stack frames traced.

NOTES

The default is to trace the stack in reverse order unless -force or -forward are specified. If the virtual-ADDR has a zero offset, then the trace starts at the offset of the first stack (stack_header.stack_begin_ptr). If it has a nonzero offset, then the trace starts from that offset in the given stack.

syserr_log, slog

SYNTAX

```
slog {-control_args}
```

FUNCTION

displays all or parts of the syserr_log and syserr_data segments from the dump. It does not examine the perm_syserr_log. The default is to print the entire log.

CONTROL ARGUMENTS

-action A
displays messages starting at action 0 to the action code specified by A, where A is a decimal integer in the range 0 to 9.

-exclude STR -ex STR
excludes any message that contains STR, where STR is a string that is matched against messages in the log.

-last N, -lt N
starts the scan N messages back from the end of the log, where N is a decimal integer.

-match STR
displays any message that contains STR, where STR is a string to be matched against messages in the log.

-expand, -exp
interprets the binary data of messages. The format is generally dependent on the text of the message.

traffic_control_queue, tcq

SYNTAX

```
tcq {-control_args}
```

FUNCTION

displays process DBR, process state, process ID, current CPU, and user ID from the traffic controller's eligible queue, as well as the "process number" in the dump. The default is to display only the eligible queue.

CONTROL ARGUMENTS

-all
: displays the eligible, real-time, interactive, and work-class queue entries, including the unthreaded entries.

-ready, -rdy
: displays the eligible, real-time, interactive, and work-class queues, excluding the unthreaded entries.

value, v

SYNTAX

```
v ptr_ni...ptr_nn
or
v -all
```

FUNCTION

displays the current value of one or all of the temporary pointers.

ARGUMENTS

ptr_n
: specifies which of the temporary pointers is to be displayed. Refer to the set request for a list of the azm-defined pointer names.

-all, -a
: specifies that all of the pointers are to be displayed. This is the default.

verify_associative_memory, vfam

SYNTAX

```
vfam {-control_args}
```

SYNTAX AS AN ACTIVE REQUEST

```
[vfam {-control_args}]
```

FUNCTION

performs a consistency check on the associative memories stored at the time of a dump by comparing them to the appropriate entries in the "dump dseg" and page tables. When used as an active request, returns "true" if any inconsistencies are found, "false" otherwise.

CONTROL ARGUMENTS

-ptw
restricts the verification to the PTW associative memories.

-sdw
restricts the verification to the SDW associative memories.

NOTES

If you give no argument, both SDW and PTW associative memories are checked.

why

SYNTAX

```
why
```

FUNCTION

tries to find the stack that has a call to syserr_real$syserr_real or call_bos$call_bos and sets the temporary pointers pr6 and prfr to the stack frame. This request searches the stacks for a frame that has a return_to_ring_0_ frame and sets the temporary pointers from this set of machine conditions that called this entry.

NOTES

If the crash is due to fim_util$check_fault finding a problem, the machine condition CU data is displayed and all temporary pointers are set from these machine conditions. If this is an execute fault, then some lock info is printed and the process selected is lock ordered: sst_seg$ptl followed by sst_seg$aslt, then scs$connect_lock followed by tty_buf$slock and tty_buf$timer_lock.

If this dump is due to a manual return to BCE, then some pertinent lock information is also printed.

Standard Subsystem Requests

.
prints a line describing the current invocation of azm.

?
prints a list of requests available in azm.

abbrev, ab
controls abbreviation processing of request lines.

answer
    provides preset answers to questions asked by another request.

do
    executes/returns a request line with argument substitution.

exec_com, ec
    executes a file of azm requests that can return a value.

execute, e
    executes a Multics command line.

execute_string, exs
    substitutes arguments into a control string.

help
    prints information about azm requests and other topics.

if
    conditionally executes/returns one of two request lines.

list_help, lh
    displays the name of all azm info segs on given topics.

list_requests, lr
    prints a brief description of selected azm requests.

quit, q
    exits azm.

ready, rdy
    prints a Multics ready message.

ready_off, rdf
    disables printing of a ready message after each request line.

ready_on, rdn
    enables printing of a ready message after each request line.

substitute_arguments, substitute_args, sbag
    substitutes arguments into a control string and prints the result on user_output.

subsystem_name
    prints/returns the name of this subsystem.

subsystem_version
    prints/returns the version number of this subsystem.

The standard escape convention for executing Multics command lines (..) is also supported.

Name: as_who

SYNTAX AS A COMMAND

```
as_who {-control_args} {User_ids}
```

SYNTAX AS AN ACTIVE FUNCTION

```
[as_who {-control_args} {User_ids}]
```

FUNCTION

lists the contents of the answering service's user tables answer_table, absentee_user_table, and daemon_user_table in the directory >sc1. As an active function, it returns Person_id.Project_id for the processes selected for output, separated by spaces.

ARGUMENTS

User_ids
can be the access control names

Person_id.Project_id
lists all users logged in with the specified Person_id and Project_id.

Person_id
lists all users logged in with the specified Person_id.

.Project_id
lists all users logged in with the specified Project_id.

You can use the star convention.

CONTROL ARGUMENTS

-absentee, -as
prints the ratio of absentee users logged in to the number of absentee slots currently available and then lists the absentee users.

-channel chn_id, -chn chn_id
lists only interactive users whose tty ID matches chn_id, daemon users whose source name (e.g., prta, vinc, etc) matches chn_id, or absentee users whose absentee name (e.g., absl) matches chn_id. The chn_id argument can be a starname (to cause several users to be listed).

-connected
prints a list of connected (interactive) processes only.

-cpu
shows CPU usage (in seconds) for the listed processes. You need access to metering_gate_ or phcs_.

-daemon, -dmn
  prints the number of currently active daemon processes and then lists them.

-disconnected, -disc
  prints a list of disconnected processes only.

-group {name}, -gp {name}
  prints a list of users that fall under the specified load control group (see "Notes" below).

-idle
  shows how long (in seconds) the listed processes have been idle. You need access to metering_gate_ or phcs_.

-interactive, -ia
  prints a list of all users having current interactive processes.

-long, -lg
  prints the long form of output including log-in time and tty ID. If you give no -lg, the command prints the User_id (Person_id.Project_id) and flag for each user (see "Notes" below).

-name, -nm
  sorts the users by name.

-no_header, -nhe
  suppresses column headings and load control heading from the printed output.

-pdir_volume {lv_name}, -pdv {lv_name}
  either includes in the output the name of the logical volume containing the user's process directory segments (if you supply no lv_name) or prints information about only those users whose process directory segments are on the volume specified by lv_name.

-process_id, -pid
  includes the process_ids for the listed processes.

-project, -pj
  sorts the users by project.

-secondary, -sc
  prints a list of all users having currently active secondary user processes (see "Notes" below).

NOTES

Access to the user table segments is usually restricted. If you lack access to these tables, use the who command (see the Commands manual, AG92) to list the whotab segment in >sc1, which is a public list of logged-in users.

The default sort is by login time.

Anonymous users' true log-in names are shown, preceded by a *.

This page intentionally left blank.

Specification of the -long control argument returns time of login, tty ID, weight, device channel, load control group, flags indicating special variables, and a User_id for each user.

Included in the output next to the User_id are flags, indicating preemption attributes (primary or secondary status), and the current status of that user's process (these attributes vary according to the login instance and the discretion of the project administrator). When this command is invoked with the -long control argument, the column listing these flags has the heading "PNDS." The flags in these four column positions indicate the user's status with respect to: Preemption, Nolist, Disconnected, and Suspended status. Additionally, if the -idle control argument is specified, R and W flags can occur. An R flag indicates the process is ready; a W flag indicates the process is waiting for another event.

Possible preemption attribute flags are:

<blank>
    indicates that the user has primary status.

S
    indicates that the user has secondary status.

+
    the user has the nobump attribute (cannot be preempted).

>
    the user is subject to bumping (preemption) by other members of his project, whose grace period has not yet run out.

X
    the user has been bumped but has not yet logged out.

D
    identifies a daemon user.

Possible nolist flags are:

<"blank>"
    the user does not have the nolist attribute.

N
    the user has the nolist attribute.

Possible disconnected flags are:

<blank>
    the user is not disconnected.

D
    the user is disconnected.

Possible suspended flags are:

<blank>
: the user is not suspended.

S
: the user is suspended.

EXAMPLES

In the following example, the as_who command is invoked with the -long control argument, and the resulting output is printed:

```
!  as_who -long

Multics mpp03221; Development Machine.
Load = 11.0 out of 70.0 units; users = 11 out of 70
Absentee users = 1 background, 1 foreground; Max background absentee users = 3
Daemon users = 7
System up since 04/23/84   1308.3
Scheduled shutdown at 04/23/84  2300.0
Last shutdown was at 04/22/84  2307.7

Login at    TTY    Load   Chan       Group     PNDS  User ID

2119.1      none   1.0     a.h018    SysProg   > D  Smith.Multics
2150.3      001    1.0     a.h019    Admin     >    Jones.SysAdmin
2200.1      Q 1    1.0    abs1       Admin          Jones.SysAdmin (crank)
2207.4      Q FG   1.0    abs2       SysProg        Doe.Multics (load_tape)
1308.4      cord   1.0    cord       IO        D    IO.SysDaemon
1308.6      bk     1.0    bk         System    D    Backup.SysDaemon
1308.6      prta   1.0    prta       IO        D    IO.SysDaemon
1308.6      puna   1.0    puna       IO        D    IO.SysDaemon
1308.7      vinc   1.0    vinc       System    D    Volume_Dumper.SysDaemon
1308.7      nw     1.0    nw         System    D    Network_Daemon.Daemon
1308.7      ns     1.0    ns                   D    Network_Server.Daemon
```

The following example invokes the as_who command to list all users on projects that begin with "Mi".

```
!  as_who .Mi*
```

The following example invokes the as_who command to list all users on the CompBar project whose names begin with "A".

```
!  as_who A*.CompBar
```

Name: backup_dump

SYNTAX AS A COMMAND

```
backup_dump path -control_args
```

FUNCTION

either dumps a single segment, directory, or subtree or uses a dump control file to dump a set of segments, directories and subtrees. It is one of the commands used to control hierarchy dumping of storage system segments and directories to magnetic tape. The other commands are:

```
catchup_dump
complete_dump
end_dump
start_dump
wakeup_dump
```

The backup_dump command is the most general of all the hierarchy dumping commands. It is called by the start_dump, catchup_dump and complete_dump commands to perform the actual dumping, after they have set certain perprocess static switches. In addition, you can use various control arguments to make the backup_dump command imitate the kind of dumping done by the start_dump, catchup_dump and complete_dump commands.

The backup_dump command allows cross dumping, a feature not allowed by the other hierarchy dumping commands. That is, it allows a specified segment, directory or subtree to be dumped to tape and recorded on the tape as coming from a different location in the hierarchy.

You should note that argument processing for all of the hierarchy backup commands is performed by a common argument processing procedure. The values of all arguments are remembered in static storage and remain in effect for the life of the process, unless changed by arguments given in subsequent invocations of backup commands. It should also be noted that the dumping commands and the reloading/retrieving commands are all part of the same hierarchy backup system, and argument values set by the dumping commands remain in effect for the reloading/retrieving commands and vice versa, unless overridden. However, dumping and reloading cannot be done in the same process; use the new_proc command between dumping and reloading. See "Notes on Default Arguments" below.

ARGUMENTS

path
    is the absolute pathname of the segment, directory or subtree to be dumped.

CONTROL ARGUMENTS

-all
: causes all directory entries (in the specified subtree) to be dumped, regardless of their dates modified or dates dumped. This argument overrides a previously given -dtd control argument or DATE argument. This is the default.

-brief_map, -bfmap
: creates a map file that lists the processed entries.

-control path
: indicates that path is the pathname of a dump control file. The suffix "dump" is assumed. For example, "-control sys_dirs" specifies a control file named sys_dirs.dump, in the working directory. See "Notes on Format of a Dump Control File" below.

-debug
: disables those phcs_ and hphcs_ calls that deactivate dumped segments and set quotas. This allows nonprivileged users to use backup_dump to save copies of their hierarchies on tape.

-destination STR, -ds STR
: specifies a destination for printing maps and error files. The default is "incremental" for maps and "error file" for error files.

-dtd
: tests and dumps each segment only if the segment or its branch has been modified since the last time it was dumped.

-error_of
: writes error messages into a file rather than online. The name of the error file is printed when the first error is encountered. This is the default.

-error_on
: writes error messages on the user's terminal.

-header STR, -he STR
: specifies a heading for dprinting maps and error files.

-hold
: leaves the current hierarchy dump tape or tapes mounted and inhibits rewinding after the current hierarchy dump cycle is completed.

-map
: writes a list of the segments and directories processed into a file. This is the default.

-nodebug
: enables hphcs_ calls to set quotas and the transparency switches. This is the default.

-nohold
rewinds and unloads the hierarchy dump tape or tapes at the end of the current dump pass. This is the default.

-nomap
inhibits listing of the names of processed segments and directories and turns the tape switch on (see -tape below).

-nooutput
inhibits writing hierarchy dump information even if the tape switch is on. This is used for a test run or debugging.

-noprimary, -npri
uses each pathname as given. The default is -primary.

-notape
inhibits writing of a hierarchy tape. This argument also causes a map to be created even if it was previously inhibited. (See -map above.)

-only
indicates that only the requested segment or directory and its branch are to be dumped. This is the opposite of -sweep.

-operator STR
indicates that STR is the user's name or initials (up to 16 characters in length).

-output
writes hierarchy dump information onto the tape if the tape switch is on. This is the default.

-primary, -pr
replaces all directory names in each pathname with the primary names. This is the default.

-pvname STR
indicates that segments and directories may only be dumped if they reside on the physical volume specified by STR.

-request_type STR, -rqt STR
specifies an output request type for printing maps and error files. Available request types can be listed by using the print_request_types command (described in the *Multics Commands and Active Functions* manual, Order No. AG92). The default is "printer." *

-sweep
indicates that the whole subtree beginning with the given directory is to be dumped, subject to the criteria of the -dtd control argument or the DATE argument if either has been invoked. This is the default.

-tape
allows writing of a tape. This is the default.

-tapes N
indicates that N is the number of output tape copies to be made where N can be either 1 or 2. The default is 1.

-1tape
sets the number of tape copies to 1 as an alternative to the -tapes argument.

-2tapes
sets the number of tape copies to 2 as an alternative to the -tapes argument.

DATE
an argument beginning with a character other than "-" or ">" is assumed to be a date, in a format acceptable to the convert_date_to_binary_ subroutine. If the argument can be converted to a date, then only segments and directories modified after that date are dumped.

NOTES ON DEFAULT ARGUMENTS

The values of arguments given to any of the hierarchy backup commands are remembered in static storage and remain in effect for the life of the process, unless explicitly changed during the invocation of a subsequent backup command.

The following defaults are in effect for the dumper before any backup commands are given; they are not, however, reset to these values at the start of each backup command, except as noted.

```
-all
-contin
-error_of
-map
-nodebug
-nohold
-output
-primary
-sweep
-tape
```

The following defaults are set automatically at the time the respective commands are executed:

```
catchup_dump
     -tape
      (default date yesterday at midnight)

complete_dump
     -all
     -tape

start_dump
     -dtd
     -hold
     -tape
     -wakeup 60
```

NOTES ON FORMAT OF A DUMP CONTROL FILE

The control file specified by "-control path" is an ASCII segment containing absolute pathnames of entries (segments, MSFs, and directory subtrees) to be dumped, each on a separate line. Cross-dumping is specified by "=new_path" following a pathname, with no intervening spaces, where new_path is the pathname of the new parent directory if the string contains >'s: otherwise, it is a new entryname to replace the entryname portion of the pathname dumped. The entry is placed on the tape as if its pathname were the resulting new pathname.

Name: backup_load

SYNTAX AS A COMMAND

```
backup_load {path} {-control_args}
```

FUNCTION

either reloads the entire contents of one or more hierarchy dump tapes into the hierarchy, leaving it looking exactly as it did when the dump tape was made, or uses a retrieval control file to reload a set of segments, directories and subtrees. It is one of the commands used for hierarchy reloading and retrieving of storage system segments and directories. The other commands are:

```
reload (initializer command)
reload (Multics command)
reload_system_release
retrieve
```

The backup_load command is the most general of all the hierarchy reloading/retrieving commands. It is called by the reload and retrieve Multics commands, and by the reload and reload_system_release initializer commands. It places its maps in the working directory and doesn't automatically dprint them.

You should note that argument processing for all of the hierarchy backup commands is performed by a common argument processing procedure. The values of all arguments are remembered in static storage and remain in effect for the life of the process, unless changed by arguments given in subsequent invocations of backup commands. It should also be noted that the dumping commands and the reloading/retrieving commands are all part of the same hierarchy backup system, and argument values set by the dumping commands remain in effect for the reloading/retrieving commands and vice versa, unless overridden. However, dumping and reloading cannot be done in the same process; use the new_proc command between dumping and reloading. See "Notes on Default Arguments" below.

ARGUMENTS

path
: is the absolute pathname of a retrieval control file (see "Notes on Format of a Retrieval Control File" below). This argument is optional. It can be given anywhere on the command line.

CONTROL ARGUMENTS

-all
: causes segments to be retrieved from the tape regardless of their date/time dumped. This control argument overrides a previously given DATE argument. This is the default.

-brief_map, -bfmap
: creates a map file that lists the processed entries.

-debug
: disables those hphcs_ calls that set quotas and transparency switches.

-destination STR, -ds STR
: specifies a destination for printing maps and error file. The default is "incremental" for maps and "error file" for error files.

-error_of
: writes error messages into a file rather than printing them. The name of the error file is printed when the first error is encountered. This is the default.

-error_on
: writes error messages on the user's terminal.

-first
: prevents searching a tape for additional copies of a requested segment or subtree after the first copy has been retrieved.

-header STR, -he STR
specifies a heading for printing maps and error files.

-last
indicates that the last copy of a given segment or subtree on a tape or set of tapes is to be retrieved. This is the default.

-map
writes a list of the segments and directories processed into a file. This is the default.

-nodebug
enables hphcs_ calls to set quotas and the transparency switches. This is the default.

-nomap
inhibits listing of the names of processed segments and directories.

-noprimary, -npri
uses each pathname as given. The default is -primary.

-noqcheck
causes the hierarchy reload to be done with quota checking suspended. Access to hphcs_ is required. This is the default.

-noquota
inhibits resetting of quotas. See -quota. This is the default.

-noreload
inhibits actual hierarchy reloading of segments into the hierarchy. This control argument can be used with -map to create a table of contents of the tape. The -noreload control argument also causes the names that would have been reloaded to be put into the map.

-nosetlvid
inhibits the setting of the logical volume identifiers for each directory to be reloaded.

-notrim
inhibits deletion of entries in a directory. Entries can only be added or modified.

-operator STR
indicates that STR is the user's name or initials (up to 16 characters in length).

-primary, -pri
replaces all directory names in each pathname with the primary names. This is the default.

-pvname STR
: indicates that segments and directories may only be reloaded onto the physical volume specified by STR.

-qcheck
: causes quota restrictions to be enforced during the reload.

-quota
: causes the quotas on directories being reloaded to be set to the values they had when the directories were dumped. Access to hphcs_ is required.

-reload
: enables actual reloading of segments into the hierarchy. This is the default.

-request_type STR, -rqt STR
: specifies an output request type for printing maps and error files. Available request types can be listed by using the print_request_types command (described in the *Multics Commands and Active Functions* manual, Order No. AG92). The default is "printer".

-setlvid
: enables setting of the logical volume identifier for reloaded entries inferior to each directory reloaded. This is the default.

-trim
: enables deletion of all entries in a directory not found in the copy of that directory being reloaded. This causes entries deleted from an earlier version of the directory to be deleted when a later version is reloaded. It has effect only in the case of a directory that is both on the tape and in the hierarchy. This is the default.

DATE
: an argument beginning with a character other than "-", or ">" is assumed to be a date in a format acceptable to the convert_date_to_binary_ subroutine. If it can be converted successfully, then the hierarchy retriever only retrieves segments and directories dumped at or after the given date/time.

NOTES ON DEFAULT ARGUMENTS

The values of arguments given to any of the hierarchy backup commands are remembered in static storage and remain in effect for the life of the process, unless explicitly changed during the invocation of a subsequent backup command.

The following defaults are in effect for the reloader and retriever before any backup commands are given: they are not, however, reset to these values at the start of each backup command, except as noted below.

```
-all
-error_of
-map
-nodebug
-nohold
-noquota
-primary
-reload
-setlvid
-trim
```

The following defaults are set automatically at the time the respective commands are executed:

```
reload (initializer command), reload (Multics command),
  reload_system_release:
    -quota
    -trim

retrieve:
    -all
    -noquota
    -notrim

All of the above commands:
    -map
```

NOTES ON FORMAT OF A RETRIEVAL CONTROL FILE

The hierarchy retrieval is controlled by an ASCII segment containing one line for each object to be retrieved. A line can contain a single pathname or two pathnames separated by an equal sign. The left-hand side specifies the segment or directory sought and the right-hand side, if present, specifies the new name under which that entity is to be retrieved. The sought pathname must begin with a > and end with either an entryname or the characters >**. If an entryname is specified, a single object of that name is retrieved.

If >** is specified, the entire directory hierarchy, beginning at the point indicated in the pathname, is retrieved. In this case, the right_hand pathname, if present, ends in the name of the directory under which these entries are to be reloaded. For example:

```
>udd>one_dir>**=>udd>two_dir
```

If a new name is specified on the right, it can be either a pathname or an entryname. If an entryname is given, the single object found is loaded with its former pathname and the new entryname.

If two pathnames are specified, both are checked against the current hierarchy and a new pathname consisting only of the primary entryname is created. This new pathname, as well as the original, is then used in searching the hierarchy. For example, >udd>sd is translated into >user_dir_dir>SysDaemon and both versions are sought. Primary names are used unless the -noprimary control argument is in effect.

A hierarchy retrieval control file can contain a maximum of 256 lines.

EXAMPLES

A hierarchy retrieval control file containing the line:

```
>udd>Multics>**
```

searches the tape for directories and segments whose first two pathname components are >user_dir_dir>Multics. These items are retrieved as found.

A hierarchy retrieval control file containing the line:

```
>ldd>a>b>c
```

searches the tape for the segment >library_dir_dir>a>b>c. This item is retrieved as found.

A hierarchy retrieval control file containing the line:

```
>ldd>a>b=c
```

searches the tape for the segment >library_dir_dir>a>b. This item is retrieved under the name >ldd>a>c.

A hierarchy retrieval control file containing the line:

```
>ldd>x>y>**=>ldd>z>y
```

searches the tape for directories and segments whose first three pathname components are >library_dir_dir>x>y. These items are retrieved in the subtree >ldd>z>y.

Name: bind_fnp

SYNTAX AS A COMMAND

```
bind_fnp pathname {-control_args}
```

FUNCTION

produces a core image segment that can be loaded into the FNP. It uses two control segments: a bindfile, which specifies the configuration that the FNP will support, the names and ordering of the object segments included in the core image, and the size of certain software tables; and an optional search rules segment, which specifies which directories are searched to find the object segments.

ARGUMENTS

pathname
    specifies the pathname of the bindfile. If pathname does not have a suffix of bind_fnp, one is assumed.

CONTROL ARGUMENTS

-cross_ref, -cref
    adds a symbol cross reference to the listing segment. If -cross_ref is specified, the listing is generated regardless of whether -list is also specified.

-list, -ls
    produces a listing segment whose name is derived from the name of the bindfile, with the suffix changed to list. The listing segment is a record of the binding. It contains a copy of the bindfile, a load map, and any error messages generated during binding.

-search, -se
    indicates that the user wishes to specify the rules used to search for Multics Communications System object segments being bound into the core image. If given, there must be a segment in the working directory containing an ASCII list of relative pathnames of directories to be searched in the order in which the search is desired. By default, the working directory is searched. This segment must have the same entryname as the bindfile, but with the suffix changed to search.

-version STR
    assigns a version of STR to the core image. The maximum length of STR is four characters. If this control argument is given, it overrides the version keyword specified in the bindfile.

NOTES

A default bindfile is supplied with the system. In general, the only fields that a site administrator would change are: hsla, lsla, version, order, and the size keyword for the trace module.

When creating a new FNP core image, object segments that are unchanged must be extracted from the object archive (see the *Multics Commands and Active Functions* manual, Order No. AG92) into a directory in the search list before executing the bind_fnp command.

The syntax of the bindfile is described in the *Multics Communications Administrator's* manual, Order No. CC75.

Name: before_journal_meters, bjmt

SYNTAX AS A COMMAND

```
bjmt {paths} {-control_args}
```

FUNCTION

displays metering information about data management before journal activity. The information provided summarizes activity on a per-system and per-journal basis. The reset capability sets the meters to zero for the process only.

ARGUMENTS

paths
: are pathnames of before journals for which metering information is to be displayed. If the .bj suffix is not present, it is added. If no pathname is specified, metering information is displayed for all journals currently active in the system.

CONTROL ARGUMENTS

-brief, -bf
: selects the brief format for reporting on before journal usage. This is the default.

-long, -lg
: selects the long format for reporting on before journal usage. This format includes a breakdown of each before journal manager operation performed during the current invocation of DMS.

-report_reset, -rr
: reports on the meters and then resets them.

-reset, -rs
: resets the meters to zero without printing a report.

ACCESS REQUIRED

Some portions of the long report require re access to dm_admin_gate_. Nonprivileged users can display the unrestricted portions of the report.

EXAMPLES

Two sample invocations of this command appear below. The first displays the brief format (default), the second, the long format.

```
bjmt

Total metering time:  3:09:51


Journals in use 1 of 64
Pages held 0 of 700 (700 per journal).
Segments active in
    4K pool 0 of 400
    16K pool 0 of 150
    64K pool 0 of 60
    256K pool 0 of 25



>site>dm>system_low>system_default.bj

journal size 4000
before images written 0
before image bytes written 0
times journal filled 0
successful recycles 0
control intervals recycled 0
transactions started 1
non null transactions 0
avg before image/transaction 0
avg bytes/before image 0
avg bytes/transaction 0
avg control intervals/recycle 0
time stamp 11/12/84 1845.7 est Mon
pages currently held 0
```

```
 bjmt -lg

 Total metering time:  3:09:19


 Journals in use 1 of 64
 Pages held 0 of 700 (700 per journal).
 Segments active in
     4K pool 0 of 400
     16K pool 0 of 150
     64K pool 0 of 60
     256K pool 0 of 25
```

```
Calls to
    begin transaction 1
    write before image 0
    write abort mark 0
    write commit mark 1
    write fm post commit 0
    write fm rollback handler 0
    write rollback mark 0
    rollback 0

Synch write
    attempts 0
    holds 0
    invalid - null DM stamp 0
    invalid - bad DM stamp 0
    invalid - bad BJ index 0
    invalid - bad time stamp 0
    tosses 0

Other ring zero calls
    unlink 0
    activate 0 (0 denied)
    deactivate 0
    set stamp 0
    allocate 3
    free 2


>site>dm>system_low>system_default.bj

journal size 4000
before images written 0
before image bytes written 0
times journal filled 0
successful recycles 0
control intervals recycled 0
transactions started 1
```

```
non null transactions 0
avg before image/transaction 0
avg bytes/before image 0
avg bytes/transaction 0
avg control intervals/recycle 0
time stamp 11/12/84 1845.7 est Mon
pages currently held 0
```

Name: bootload_fs

SYNTAX AS A COMMAND

```
bootload_fs operation {args}
```

FUNCTION

allows you to operate on a copy of the bootload command environment (BCE) file system. This includes the ability to extract the real BCE file system and to replace it with this working copy.

ARGUMENTS

operation
: is an operation listed below under "List of Operations."

args
: are arguments required by the designated operation.

LIST OF OPERATIONS

The operations are grouped into two categories. The first group determines the location of your copy of the BCE file system; operations in this group can also extract the real BCE file system and overwrite it with your copy. The second group operates on objects in your working copy of the BCE file system.

OPERATIONS ON PARTITIONS

discard_partition, dpart"

```
Syntax:  bootload_fs discard_partition {-control_arg}
```

discards the contents of the working copy of the BCE file system. Follow this operation by a read_partition, use_partition or init_partition operation.

The control arguments is

-force, -fc
: discards the contents of the working copy of the BCE file system without querying you first.

init_partition, ipart

```
Syntax:  bootload_fs init_partition {-control_arg}
```

clears out the contents of the working copy of the BCE file system. The result of init_partition is a file system containing no files; the result of discard_partition is no file system at all.

The control arguments is

-force, -fc
: clears out the contents of the working copy of the BCE file system without querying you first.

read_partition, rpart

```
Syntax:  bootload_fs read_partition pv_name part_name
```

reads the BCE file system from a specified disk partition into your working copy of the file system, overwriting the previous contents of your copy. You need access to hphcs_.

The arguments are

pv_name
: is the name of a mounted physical volume.

part_name
: is the name of a partition on the specified volume to be read.

save_partition, svpart, spart

```
Syntax:  bootload_fs save_partition path
```

saves the current contents of your working copy of the BCE file system into a segment.

The arguments is

path
: is the pathname of a segment that is overwritten with your working copy of the BCE file system.

use_partition, upart

```
Syntax:  bootload_fs use_partition path
```

copies the contents of a user-specified segment to become your working copy of the BCE file system.

The arguments is

path
: is the pathname of a segment that overwrites the current contents of your working copy of the file system.

write_partition, wpart

```
Syntax:  bootload_fs write_partition {pv_name {part_name}}
         {-control_args}
```

replaces the BCE file system found in the specified disk partition with your working copy. You need access to hphcs_.

The arguments are

pv_name
: is the name of a mounted physical volume.

part_name
: is the name of a partition on the specified volume to be read.

The control arguments is

-force, -fc
: overwrites the old partition without querying you first.

If you supply no arguments, write_partition uses the identity of the partition last specified in a read_partition operation.

OPERATIONS ON FILES

delete_files, delete, dl

```
Syntax:  bootload_fs delete_files file_name
```

deletes files from the working copy of the BCE file system.

The arguments is

file_name
    is the name of a file that is to be deleted from the BCE file system.

list_files, list, ls

```
Syntax:  bootload_fs list_files
```

lists the names and lengths (in characters) of the files in the working copy of the BCE file system.

read_file, read, r

```
Syntax:  bootload_fs read_file file_name path
```

extracts a file from the working copy of the BCE file system and places it into a Multics storage system file.

The arguments are

file_name
    is the name of a file within the working copy of the BCE file system.

path
    is the pathname of the Multics file into which the BCE file is to be copied.

rename_file, rename, rn

```
Syntax:  bootload_fs rename_file old_file_name new_file_name
```

renames a file within the working copy of the BCE file system.

The arguments are:

old_file_name
is the name of an existing file in the BCE file system.

new_file_name
is the new name to be given to the old file.

write_file, write, w

```
Syntax:  bootload_fs write_file path file_name
```

places a copy of a Multics storage system file in the working copy of the BCE file system.

The arguments are:

path
is the name of a file in the Multics hierarchy to be copied into the BCE file system.

file_name
is the name the copy is to have within the BCE file system.

Name: cache_meters

SYNTAX AS A COMMAND

```
cache_meters {-control_args}
```

FUNCTION

interprets and prints per-system metering information on central processor, hardware recoverable, cache memory errors.

CONTROL ARGUMENTS

-brief, -bf
inhibits the display of lines that are all zero. The default is to display all counters even if they are all zero.

-cpu {tag1...tagN}
displays fault counts for those processors specified by tagi. Tag may be one of the letters a through h or A through H. If tag is omitted, all processors are selected.

-report_reset, -rr
    generates a full report and then performs a reset operation.

-reset, -rs
    resets the metering interval for the invoking process so that the interval begins at the last call with -reset specified. If -reset has never been given in a process, it is equivalent to having been specified at system initialization time. The metering interval is reset for ALL processors.

-total, -tt
    displays total error counts for all processors. This is the default.

NOTES

If all of a processor's cache counters are zero, that processor is omitted from the display.

The following is a brief description of the variables printed by the cache_meters command.

CPU Tag
    is the tag or name of the processor.

Cache Type
    is the type of processor cache and cache size.

Primary Dir Parity/MultiMatch
    the processor detected a parity or multiple match in the primary cache directory. This condition causes the processor to abort the cache cycle and go to backing store for the data. Ths counter is valid for all processors and cache types.

The following counters are valid only for DPS 8 processors:

PT X Buffer OVFL/PAR/SEQ Err
    either the processor's Write-Notify buffer logic could not keep up with Write-Notify signals for the port specified by X (A, B, C or D), or the processor detected a parity or sequence error on the specified port. Detection of this condition causes the processor to flush cache.

Primary DIR/PT Buffer Overflow
    the processor's primary cache directory or port Write-Notify buffer logic could not keep up with Write-Notify signals. This condition forces the processor to flush cache.

WNO Parity ANY Port
    the processor detected a parity error on the data or address protion of the Write-Notify signal for port A, B, C or D. This condition forces the processor to flush cache.

Level N Dup Dir Parity
: the processor detected a parity error in the duplicate cache directory match logic for the level specified by N (0, 1, 2 or 3). This condition forces the processor to flush cache.

Dup Dir MultiMatch
: the processor detected a multiple match condition in the duplicate cache directory. This condition forces the processor to flush cache.

Name: change_kst_attributes

SYNTAX AS A COMMAND

```
change_kst_attributes {-control_arg} target attribute
```

FUNCTION

allows a user to change selected per-process attributes of a segment.

ARGUMENTS

target
: specifies the segment whose known segment table (KST) attributes are to be changed. Either a relative pathname or an octal segment number can be supplied.

CONTROL ARGUMENTS

-name, -nm
: is only used if the target is a relative pathname that looks like a segment number.

LIST OF ATTRIBUTES

One or more of the following must be given:

allow_deactivate
: if set, permits explicit deactivation of the segment.

allow_write
: if set, the user is not prevented from writing into the segment or directory if she or he has permission to do so. *

tms
: if set, date-time-modified is not updated on the account of the user.

tpd
: if set, pages of this object are not placed on the paging device on the account of the user.

tus
: if set, date-time-used is not updated on the account of the user.

ACCESS REQUIRED

This command requires access to the hphcs_ gate if the tms or tus attributes are to be set; otherwise, access to the phcs_ gate is required.

NOTES

Because directories are activated when their segment numbers are assigned, it is not possible to set meaningfully the tpd, tms, tus, or allow_deactivate attributes for a directory.

If an attribute is preceded by the circumflex character (^), then the attribute is reset; otherwise, the attribute is set. Attributes not mentioned are unaffected.

Name: change_tuning_parameters, ctp

SYNTAX AS A COMMAND

```
ctp name1 value1 {...  nameN valueN} {-control_arg}
```

FUNCTION

used to change the value of several tuning parameters within the system.

ARGUMENTS

namei
: is the name of a tuning parameter whose value is to be changed. It can be either the long name or the short name of the parameter.

valuei
: is the representation of the value to which the tuning parameter is to be set. It typically can be an integer, a decimal number of seconds, either "on" or "off," a decimal number, or a fullword octal value. The data type of the value depends on the individual tuning parameter being set. The data type of each valid tuning parameter is described in the *Multics System Maintenance Procedures* manual, Order No. AM81.

CONTROL ARGUMENTS

-silent
: causes the message normally printed on the operator's console to announce the change not to be printed but only to be logged. You can use -silent only in the Initializer's process.

ACCESS REQUIRED

You need access to metering_gate_ and to hphcs_. If you give -silent, ctp requires access to initializer_gate_ also.

NOTES

Before making any change, you are shown the change and asked if it is correct. The first pair of values represent the old and new values of the parameter, while the second pair of values (parenthesized) represent the octal contents of the word in the database where that parameter is kept. You must respond "yes" followed by a newline character for the change to be made. Invalid parameters are rejected. The current values of the parameters can be obtained by using the print_tuning_parameters command. (For comprehensive descriptions of the tuning parameters and their effects, see the *Multics System Maintenance Procedures Manual*, Order No. AM81.)

Name: change_volume_registration, cvr

SYNTAX AS A COMMAND

```
cvr -control_args
```

FUNCTION

changes the registration information for a physical or logical volume.

CONTROL ARGUMENTS

The -pv or the -lv control argument is required and must immediately follow the command name.

The following control arguments change the registration of a physical volume. They are recognized only when the first control argument is -pv.

-date_registered DT_STR, -date DT_STR
: specifies the date the physical volume was registered on. Normally this is generated by the software automatically. Use -date only in correcting registration data after a system failure.

-device_model N, -model N
: specifies the Honeywell model number for the disk device. Use -model only in correcting registration information before volume initialization or after a system failure. Inconsistancies between medium label information and the registered information causes difficulties in volume mounting. The following values are valid for N:

VALUE	DEVICE
400	MSU0400
402	MSU0402
451	MSU0451
500	MSU0500
501	MSU0501
3380	MSU3380
3381	MSU3381

-location STR, -loc STR
: specifies the current location of the disk pack. This is for administrative information only. STR can be any 32 characters (e.g., "offline--in cabinet 13").

-manufacturer_serial STR, -serial STR
: specifies the serial number of the physical medium. This is for administrative information only. STR can be any 32 characters (e.g., "Memorex--M234634").

-name PV_NAME, -nm PV_NAME
: specifies that the name of the physical volume is to be changed. Use -nm only in correcting registration information before volume initialization or after a system failure. Inconsistancies between medium label information and the registered information causes difficulties in volume mounting.

-physical_volume PV_NAME, -pv PV_NAME
: specifies the name of the physical volume for which the registration data is to be changed. (Required, if physical volume registration changes are desired)

-physical_volume_uid UID, -pvid UID
: specifies the new unique ID of the physical volume. A UID is a string of one to 12 octal digits. Use -pvid only in correcting registration information before volume initialization or after a system failure. Inconsistancies between medium label information and the registered information causes difficulties in volume mounting.

The following control arguments pertain to changing the registration of a logical volume. They are recognized only when the first control argument is "-logical_volume" (or "-lv").

-access_class MIN_AUTH{:MAX_AUTH}
: specifies the AIM authorization of users allowed to attach the volume. The authorization can be specified by a minimum authorization value and a maximum authorization value, in which case users at any authorization in the range are allowed to attach the volume. Alternatively, the authorization can be specified as a single value, in which case only users at the specified authorization are allowed to use the volume. The authorization values must be specified using valid site-defined authorization strings. Use the print_auth_names command for a list of valid authorization values. This control argument should only be used in correcting registration information before volume initialization or after a system failure. Inconsistancies between medium label information and the registered information will cause difficulties in volume mounting.

-acs_path PATH, -acs PATH
: specifies the pathname of the access control segment (ACS) for the volume. The ACS itself is not created, but must be created at the specified path and the desired access control list set (see "Notes" below). The name of the entry must be {lv_name}.acs. If this control argument is not given when registering a public volume, only the volume owner will have executive privileges (everyone will have read/write privileges, given proper access to the hierarchy). If this control argument is not given when registering a non-public (i.e. a private) volume, the default ACS pathname will be: >udd>{owner's project_id}>{owner's person_id}>{lv_name}.acs>

-logical_volume LVNAME, -lv LV_NAME
: specifies the name of the logical volume for which to change the registration data. This control argument is required and must immediately follow the command name (if logical volume registration changes are desired).

-logical_volume_uid UID, -lvid UID
: specifies the new unique ID of the logical volume. A UID is a string of 1 to 12 octal digits. This control argument should only be used in correcting registration information before volume initialization or after a system failure. Inconsistancies between medium label information and the registered information will cause difficulties in volume mounting.

-name LV_NAME, -nm LV_NAME
: specifies that the name of the logical volume is to be changed. This control argument should only be used in correcting registration information before volume initialization or after a system failure. Inconsistancies between medium label information and the registered information will cause difficulties in volume mounting.

-owner USER_ID, -ow USER_ID
: specifies the user who is to act as the primary volume administrator. This user has the privilege of granting executive access to other users via the ACS, granting read/write access to others via the ACS (for private volumes), and for setting up volume quota accounts. The person_id or project_id may be specified as "*".

-public STR
: specifies whether the new logial volume is to be private or public. The value of STR can be yes (public) or no (private). This control argument should only be used in correcting registration information before volume initialization or after a system failure. Inconsistancies between medium label information and the registered information will cause difficulties in volume mounting.

EXAMPLES

```
cvr -pv pub34 -serial c886 -loc dskc_13
```

This command will change the registered serial number and location of the physical volume "pub34".

```
cvr -lv ZenDisk -owner Jones.Zen_Res
```

This command will change the registered owner of the logical volume "ZenDisk".

NOTES

Physical volume names are restricted to lowercase letters, digits, and the underscore ("_").

If any of the following control arguments are given, the user is queried to determine if the changes are intentional. This is to avoid problems in volume mounting due to errors in the registration. They are:

```
-logical_volume_uid
-physical_volume_uid
-access_class
-device_model
-name
```

Please note that these control arguments should be used only to correct the volume registration data before the volume in question is actually initialized or after system failure. The medium label information is not changed. Inconsistancies between the label and registration data will cause difficulties in mounting the volume.

ACCESS REQUIRED

The user of this command is required to have "re" access to the mdc_rpiv_ gate and "s" to >lv. Also, if any name or UID is to be changed, "sm" is required to >lv.

Name: channel_comm_meters

SYNTAX AS A COMMAND

```
channel_comm_meters channel_name {-control_args}
```

FUNCTION

prints out metering information for a specified communications channel or channels.

ARGUMENTS

channel_name
: is the name of the channel for which information is to be printed. If it is the name of an FNP, totals for that FNP are reported. If channel_name is a starname, information for every channel matching the starname is printed.

CONTROL ARGUMENTS

-brief, -bf
: causes a reduced amount of information to be printed for each specified channel.

-error
: causes only those meters to be printed that reflect error conditions.

-since_bootload, -boot
: prints the meters accumulated since each channel's parent multiplexer (or, in the case of an FNP, the system) was last loaded. This control argument is incompatible with -since_dialup (below).

-since_dialup, -dial
: prints the meters accumulated since the channel last dialed up. This is the default. This control argument is incompatible with -since_bootload (above).

-summary, -sum
: causes a one-line summary to be printed for each specified channel. This control argument may not be specified if either -brief or -error is specified.

ACCESS REQUIRED

If a single channel is specified, the caller must either be the current user of the specified channel or have access to either the metering_gate_ gate or the phcs_ gate. If a starname is specified, the user must have access to one of the above-named gates.

NOTES

If -brief and -error are both specified, then only those error indications that would be printed with -brief are printed. See the example below.

EXAMPLES

In the example below, code characters appear at the beginning of some lines; these characters do not appear in the actual output of the command. The interpretation of the characters is as follows:

```
A -- this line appears for asynchronous channels only
S -- this line appears for synchronous channels only
B -- this line is among those printed if -brief is specified
E -- this line is among those printed if -error is specified
```

Only lines marked with both B and E are printed if -brief and -error are both specified.

```
   channel_comm_meters a.h000
   Total metering time 01:45:13
   a.h000

[The following meters are printed for all nonmultiplexed channels:]

                                   before conversion   after conversion   ratio
  B  Total characters input           984                 935             0.95
  B  Total characters output       10,540              11,400             1.09
  B  Average length of input            8.7                 8.3
  B  Average length of output          63.1                69.4

                                          read      write    control     total
Number of calls                           175       194       53          422
Average time per call (msec.)               2.3       5.8      1.7          4.1
Average chars. processed per call           5.6      56.1

Number of software interrupts             113
Average time per interrupt (msec.)          1.6

  B  Effective speed (bps)                  1.6      17.5

[The following meters are printed for physical FNP channels only]

                                          input     output
 SB  Messages transmitted                  240        224
 SB  Minimum message length                  5         12
 SB  Maximum message length                143        508
 SB  Average message length                 10.3       57.6

 SBE Invalid input messages               6 (2.5% of total)
 SBE Output messages retransmitted        8 (1.6% of total)
 SBE Timeout waiting for acknowledge      2 (0.4% of output messages)

Output overlaps in FNP                      127
Average length of DIA request queue    1.7 entries
```

```
A     Pre-exhaust status                        12
A   E Exhaust status                             7
A   E Software transfer timing errors            0
A   E Bell/quits                                 8
A   E Echo buffer overflows                      2
    E Parity errors                              0

Avg. number of pending status events    1.9
    E Software status queue overflows            1
    E Hardware status queue overflows            0
    E Input buffer allocation failures           1

[The following meters are printed for an entire FNP]

FNP has been up for                 04:15:12
  B  Number of channels configured        88
  B  Average number dialed up             43.7
  B  FNP idle                             74.9%
  B  Idle at peak load                     8.0%

                                     input      output
  B  Characters transmitted     71,966,400  94,934,400
  B  Characters per second           4,700       6,200

   E Abnormal DIA status events            3
   E Memory EDAC errors                    0

  B  Memory size                          64K
  B  Total available buffer pool       6,360 words
  B  Avg. amount of free space        21,876 words
  B  Average % of buffer pool available   34.7
  BE Buffer allocation failures           12
   E Output restricted by space           24


Number of interrupts from this FNP   1,964,208
Avg. time/interrupt (ms)               3.1
% of total CPU time                    1.1

Mailbox transactions:
   Input data                         220,349
   Output data                        543,210
   Input control                       14,111
   Output control                      23,456
   ------------------------------------------------------------
   Total                         801,126

Average inbound mailboxes in use       1.1
Average outbound mailboxes in use      3.1
Maximum outbound mailboxes in use     16
   E No outbound mailbox available           37
```

```
    E Input rejects                         22
    E % of input transactions rejected       0.01
```

The following example shows the format of the output of the command when the -summary control argument is specified.

```
    channel_comm_meters a.h00* -summary

  cps   cpsi   cpso  iotxXsbepQqa  err  ABE  name          user

  120   0.2     5.4    xX b  Q      12  aB   a.h000        Coren
  600   2.1   102.1   t X      a    73  s    a.h005        ABClone
   30   0.5     2.6        e         2  a E  a.h009        Parrish
```

The column headings are interpreted as follows:

cps
: is the nominal speed of the channel, in characters per second.

cpsi
: is the effective speed of input over the channel, in characters per second.

cpso
: is the effective speed of output over the channel, in characters per second.

The following flags are printed if the corresponding condition has occurred at least once on the channel.

i
: invalid input message

o
: output message retransmitted

t
: timeout waiting for acknowledge

x
: pre-exhaust status

X
: exhaust status

s
: software transfer timing error

b
: bell/quit

e
: echo buffer overflow

p2
: parity error

Q
: software status queue overflow

q
: hardware status queue overflow

a
: input buffer allocation failure

err
: is the total number of errors of all kinds that have occurred on the channel.

A
: is "a" for an asynchronous channel or "s" for a synchronous channel.

B
: is the channel is in breakall mode.

E
: is the channel is in echoplex mode.

name
: is the name of the channel.

user
: is the Personid of the current user of the channel. If the channel is not in use, or the user's name is not available, this field is left blank.

Name: charge_disk

SYNTAX AS A COMMAND

```
charge_disk {path} {-control_arg}
```

FUNCTION

is used by the daily disk accounting job (diskreport in the master.ec segment) to record the disk usage figures for each project in the project's projfile entry. The figures are obtained from the disk_stat file that is produced by the sweep command. Figures are recorded only for directories in the subtree below >udd. Within that subtree, charging may be enabled or disabled on a per-logical-volume basis.

ARGUMENTS

path
: is the pathname of the disk_stat file from which the disk usage figures are obtained. The default is the segment. disk_stat. in the working directory.

CONTROL ARGUMENTS

-logical_volume name1 name2 ... nameN

-lv name1 name2 ... nameN
: where namei is one or more logical volumes for which charging is to be enabled. Charging for all other logical volumes is disabled. If this argument is not given. it is assumed that all directories under >udd have their segments on the same logical volume. and charging for this volume is enabled. The sons logical volume identifier of the first project directory encountered is used as the logical volume identifier of the volume for which charging is enabled.

NOTES

Dollar charges are not computed by charge_disk. It records the total month-to-date time-page product (tpp) for all directories inferior to each project directory in the project's projfile entry. The corresponding dollar charge is computed by the print_disk command for printing only. The disk charge actually billed is computed during monthly billing, using the disk price in effect at that time. (Thus a change in the disk price made before billing is retroactive to the beginning of the billing period.)

If the total tpp for a project, as computed from the figures in disk_stat, is less than the previous day's month-to-date tpp recorded in projfile, an error message is printed, and the projfile figure is not changed. This situation can arise in several ways, through system or human error. If a project directory is destroyed and then re-created in a way that sets its tpp integrator back to zero instead of to the previous value, charges start accruing from the date of the re-creation and the charges for the portion of the month preceding the destruction are lost. When this happens, the set_tpp command can be used to add the charge in projfile to the tpp integrator of the project directory. Before this is done, however, the system administrator should analyze the situation carefully, to be sure that adding this charge to the project is the correct thing to do. If it is determined that the projfile figure is the one in error, the edit_projfile command can be used to correct it.

Name: check_cpu_speed

SYNTAX AS A COMMAND

```
check_cpu_speed {cpu_tags}
```

FUNCTION

performs a relative check of the speed of a currently running CPU on the system.

ARGUMENTS

cpu_tags
: are the tags of CPUs configured on the system. If more than one is supplied, the values must be separated by spaces. The default is to run on all CPUs listed by the list_proc_required command that are currently marked as ON in the configuration deck.

ACCESS REQUIRED

This command requires access to the phcs_ gate to run.

NOTES

Your process is left running with the original set of system CPUs.

The command runs on a CPU outside of your original set of CPUs if the CPU tag is given on the command line.

Name: check_dir

SYNTAX AS A COMMAND

```
check_dir dir_name {User_ids}
```

FUNCTION

scans a directory and reports the names of all segments whose bit count author is not equal to any one of the specified User_ids.

ARGUMENTS

dir_name
    is the directory to be scanned.

User_ids
    are access names of the form Person_id.Project_id.tag. Any component may be "*" and omitted components are assumed to be "*". If no User_ids are specified, the User_id of the caller is assumed.

NOTES

For each segment whose bit count author does not match any of the specified User_ids. a line is printed giving:

```
   entryname     date/time modified    author
```

Name: check_mdcs

SYNTAX AS A COMMAND

```
check_mdcs volume
```

FUNCTION

checks for valid format and invalid unique identifier (UID) pathnames in the master directory control segment (MDCS) for a given volume. These segments are found in >lv, and are sometimes damaged by system crashes. Any errors found are reported via the syserr log and, if possible, corrected.

ARGUMENTS

volume
: is the name of a storage system volume.

ACCESS REQUIRED

Access to the mdc_priv_ gate is required.

Name: check_mst

SYNTAX AS A COMMAND

```
check_mst REEL_NAME {-control_args}
```

```
check_mst -file MST_FILE_NAME {-control_args}
```

```
check_mst -tape REEL_NAME {-control_args}
```

```
check_mst OUTPUT_FILE_NAME
```

FUNCTION

scans a Multics system tape (MST). producing a report on the segments it defines and checking for certain errors and inconsistencies.

ARGUMENTS

REEL_NAME
: is the name of the reel to be checked containing an MST written with generate_mst.

MST_FILE_NAME
: is the pathname of the file created by generate_mst -file.

OUTPUT_FILE_NAME
: is the name to be given to the output file if the input file has already been attached by generate_mst -hold.

CONTROL ARGUMENTS

-density N, -den N
: specifies the recorded density of the MST.

-severity N, -sv N
: specifies the minimum severity of errors to be printed.

NOTES

After the segments have been listed a cross-reference is run over the contents of the tape, to detect links to nonexistent segments or entry points or types of links that the hardcore prelinker cannot snap. The cross-referencer produces messages for links between temporary segments (temp segs) and other segments that do not exist: for example, a collection 1 temp seg cannot legally reference a collection 2 segment because the first is guaranteed to have been deleted before the second is loaded from the tape.

Next, the cross-referencer produces four sorted lists:

1. A list of references to entrypoints that cannot be found in segments. These are usually errors.

2. A list of links to segments that cannot be found on the MST, but are present in your search rules. This is a normal occurrence, since many hardcore programs check their ring of execution and make appropriate calls. When producing a modified system tape, check this list.

3. A list of segments that cannot be found at all. Unless the code that uses them is not executed, or unless they are installed online, they produce linkage errors.

4. A list of links to special star segnames (*system, *symbol, etc.): *system links cannot be used in the hardcore, and the symbol section of hardcore programs is not retained: thus investigate any entries in this list.

NOTES ON OUTPUT FORMAT

This command produces an output file named REEL_NAME.ckrout (or OUTPUT_FILE_NAME.ckrout) that contains a list of the segments on the tape, as well as diagnostic information. For each segment there is an entry of the form:

```
PRIMARY_NAME SEGNO (W, R, E) ATTRIBUTES1
 NAME2                ATTRIBUTES2
 NAME3                ATTRIBUTES3
 NAME4                PATHNAME
 NAME5                  ...             ...
  ...                   ...             ...
ACL LIST                ...
```

where:

PRIMARY_NAME
    is the primary name of the segment.

NAME1...NAMEN
    are additional names of the segment. Names that are not printed to the left of ATTRIBUTES are printed in a third column,

W, R, E
: are the segment ring brackets, in the conventional order.

ATTRIBUTES1
: are the standard SDW access attributes for the segment: read, execute, write, privileged, encacheable, gate, and wired. The gate attribute is derived from the ring brackets.

ATTRIBUTES2
: are chosen from

 init seg
 : a segment that is deleted at the end of system initialization.

 temp seg
 : an init seg that is deleted when its collection is complete.

 per process
 : segments in the Initializer's process directory.

 firmware
 : segments that contain MPC firmware images.

ATTRIBUTES3
: indicate that the segment has one of the following types of storage allocated or that no storage is allocated:

 wired length
 : is the amount of memory used rounded up to an even 16-word boundary. This attribute is only meaningful for segments loaded before paging has been initialized.

 paged length
 : is the length in pages of a segment that is either loaded after paging is initialized or that is made paged. The latter results in entries for both wired length and paged length.

 max length
 : is a standard file system max length, and is set for segments that grow dynamically.

 PATHNAME
 : is the pathname the segment gets, if any.

 ACL LIST
 : is a standard access control list.

NOTES ON ERROR MESSAGES

The checker detects and reports on several classes of errors:

SEVERITY 4--FATAL
: the tape is malformed and cannot be read. This can be due to a logic error in generate_mst or tape I/O errors.

SEVERITY 3--SEVERE ERROR
: a segment has run out of space or the rules for the system tape have been violated. The checker keeps track of the amount of space used in the linkage segments, definitions_, and name_table. If an error indicates a wired overflow, then increase the cur_length specified in the header file to the size given in the total summary in the output file; if the error indicates a paged overflow, increase max_length.

SEVERITY 2--ERROR
: the cross-referencer finds references to entrypoints that were undefined in their containing segments or segments that cannot be found.

WARNING
: other errors detected by the cross-referencer.

Checker errors are reported by com_err_ and are written to the output file.

There is a severity indicator that is zero, or one if there were no errors detected. It is accessible with the severity active function:

```
!  [severity check_mst]
```

Name: check_sst_size

SYNTAX AS A COMMAND

```
check_sst_size {-control_args}
```

FUNCTION

displays the sizes of the system segment table (SST) header, active segment table (AST) hash table, and each of the 4 pools that are found in sst_seg along with the total size of the sst_seg.

CONTROL ARGUMENTS

-4k N
: specifies that the 4k AST pool is to have N entries.

-16k N
specifies that the 16k AST pool is to have N entries.

-64k N
specifies that the 64k AST pool is to have N entries.

-256k N
specifies that the 256K AST pool is to have N entries.

-pathname STR, -pn STR
specifies that the configuration deck with a pathname of STR is to be used.

NOTES

If you use no -4k, -16k, -64k, or -256k, a configuration deck sst card is used to supply the missing value(s).

The configuration deck, if needed, is found with your search rules if you don't use -pn on the command line.

Name: check_tc_data_size

SYNTAX AS A COMMAND

```
check_tc_data_size {-control_args}
```

FUNCTION

displays the sizes of the active process table (APT) and inter-process transmission table (ITT) that are allocated in the tc_data segment.

CONTROL ARGUMENTS

-apt N
specifies that the APT is to have N entries.

-itt N
specifies that the ITT is to have N entries.

-pathname STR, -pn STR
specifies that the configuration deck with a pathname of STR is to be used.

NOTES

The sizes of the APT and ITT databases are controlled by the values found on the tcd configuration card. You can use this command to determine if a proposed change to the tcd values will cause the tc_data segment to become larger than 255 pages. You can also use it to fill in the last page of the tc_data segment through manipulation of either of the values on the tcd card.

If you use neither -apt nor -itt, a configuration deck is used to supply the missing value(s).

The configuration deck, if needed, is found with your search rules if you don't use -pn on the command line.

Name: clean_card_pool

SYNTAX AS A COMMAND

```
clean_card_pool -age n {-control_arg}
```

FUNCTION

deletes inactive card image segments, in the >daemon_dir_dir>cards subtree, created by the system card reading process.

ARGUMENTS

-age N
: deletes all segments in person directories in the pool and all person directories whose date-time-modified (dtm) is older than N.

CONTROL ARGUMENTS

-quota N
: indicates that N pages of unused quota are to be left on each remaining access class directory. If N is 0, the quota is set to the "Number of pages used." (Default: 0)

ACCESS REQUIRED

You must be able to call the system_privilege_ gate.

NOTES

After the pool is cleaned, all empty person directories and access class directories are deleted. All links and directories contained in a person directory are deleted regardless of age. All links and segments in an access class directory are deleted regardless of age.

Name: clear_partition

SYNTAX AS A COMMAND

```
clear_partition pvname partname {control_args}
```

FUNCTION

overwrites the contents of a disk partition with zeroes or optional user-supplied pattern words.

ARGUMENTS

pvname
: is the name of the physical volume on which the partition to be cleared exists.

partname
: is the name of the partition to be cleared. It must be four characters or less long.

CONTROL ARGUMENTS

-brief, -bf
: produces brief format messages.

-long, -lg
: produces longer format messages. (Default)

-pattern word
: overwrites the partition with data consisting of the specified octal pattern word. The specified word is written into every location in the partition. If you give no -pattern, a default of all zeroes is used.

ACCESS REQUIRED

You need access to the phcs_ and hphcs_ gates.

NOTES

You are always queried whether the partition should be overwritten; by default the contents of the first eight words in the partition are displayed (in octal and as ASCII characters) as part of this question, to aid in preventing accidental overwriting of the wrong partition.

See also dump_partition and list_partitions.

Name: clear_projfile

SYNTAX AS A COMMAND

```
clear_projfile {path}
```

FUNCTION

is used after monthly billing to remove the entries for deleted projects from the project file.

ARGUMENTS

path
: is the pathname of the project file. The default is the segment, projfile, in the working directory.

Name: clear_reqfile

SYNTAX AS A COMMAND

```
clear_reqfile {path}
```

FUNCTION

is used after monthly billing to remove the entries for deleted projects from the requisition file.

ARGUMENTS

path
: is the pathname of the requisition file. The default is the segment, reqfile, in the working directory.

Name: clear_resource, clr

SYNTAX AS A COMMAND

```
clr type STR1 ... STRN
```

FUNCTION

specifies that a resource has been manually cleared (degaussed) and should be returned to the free pool.

ARGUMENTS

type
: is the resource type defined in the RTDT.

STRi
: is the unique identifying name of the particular resource being cleared. If STR is specified in control argument format (i.e., if it is preceded by a hyphen), then it must be preceded by -name or -nm.

ACCESS REQUIRED

The use of this command requires execute access to the rcp_sys_ gate.

NOTES

The Manual_clear: statement in the RTMF specifies the operation of the resource data security features of automatic acquisition and release. If "yes" is specified, volumes of the designated type are locked (when released by an accounting owner) in a way that does not allow another user to acquire them until the operator certifies that the volume has been cleared of all residual information. If "no" is specified (or if the statement if omitted), locking does not occur.

Name: command_usage_count, cuc

SYNTAX AS A COMMAND

```
cuc operation {command_names} {-control_args}
```

FUNCTION

provides a record of the number of times commands are used and the User_ids for each invocation of them. The commands to be metered in this way must be listed in a segment named command_usage_list_. Usage totals are stored in a segment named command_usage_totals_. This command actually performs three operations: it prints and clears the meters, adds commands to command_usage_list_, and deletes commands from command_usage_list_.

ARGUMENTS

operation
  can be one of the following:

  print, pr
    prints (and clears) the metered data (subject to any restrictions the specified control_args impose).

  add
    adds commands to the list (command_usage_list_) of commands to be metered. Commands added to the list in a single invocation of the "cuc add" command form a command "group", which can be manipulated as a whole.

  delete, dl
    deletes command groups (see above) from the list of commands to be metered.

command_names
  are long or short names of commands. If given with either the print or delete operation, only one command name from each group to be printed or deleted need be given, and all the commands in each group so represented are acted upon. If no names are given with the print or delete operation, all command groups are printed/deleted. Command names (long and/or short) must be given with the add operation, and all the names added in a single invocation are added as a single group to the list. Short names of commands can only be used with the print and delete operations if they were specified with the add operation.

CONTROL ARGUMENTS

-all, -a
  prints meters for all the command groups, or deletes all command groups from the list. This is the default for the print and delete operations if no command_names are given. This control argument cannot be used with the add operation.

-brief, -bf
: omits column headings from the printout (can only be used with the print operation). The default is to print column headings.

-clear, -cl
: clears the usage counters and user list when meters are printed (can only be used with the print operation). It clears the user list even if the -total control argument is also specified.

-first N, -ft N
: prints only the N greatest users of the specified commands (can only be used with the print operation). This control argument cannot be used in conjunction with the -total control argument.

-total, -tt
: prints only the total use of the commands in the specified command groups, when used with the print operation. When used with the add operation, meters only the total usage of commands specified. The default with both of these operations is to print/meter the users of the commands as well as total usage. See "Notes" below. This control argument cannot be used with the delete operation.

NOTES

In order to add and delete commands, and to clear meters, the user must have rw access to the command_usage_list_ segment. Otherwise, all users should have r access to command_usage_list_, and rw access to command_usage_totals_. Both segments are found using object search rules and most commonly are in >sss (system_library_standard directory). If they are not in >sss, a link in >sss points to them.

For each group of commands added without the -total control argument, this command creates a segment named command_name.usage in >sss (or, if a link is there, wherever the link points). The user must put the link in >sss before the first usage of cuc add, since the metering program creates the command_name.usage segment in the same directory in which it finds command_usage_list_. The command_name.usage segment contains the list of User_ids for those using the commands in the group. User_ids are printed in the order of greatest usage. When the -first control argument is given, in addition to printing the user count and name for the N greatest users, this command prints an additional line giving the user count for "all others."

At sites using the access isolation mechanism (AIM), only the usage of system_low users is recorded.

EXAMPLES

In the following example, assume that no commands are listed in the command_usage_list_ segment. The user adds two commands (in two separate command groups) by typing:

```
command_usage_count add set_search_rules ssr
cuc add enter_abs_request ear -total
```

The next time a process is created, those commands can be metered by typing:

```
cuc print
```

The following lines are then displayed:

```
USAGE   COMMAND            USER   USER
COUNT   GROUP              COUNT  NAME

3       set_search_rules    2     Baker.Multics
        ssr                 1     Green.SysMaint

1       enter_abs_request
        ear
```

Note that user count and user name are not provided for the command group added with the -total control_arg.

To delete these commands from the list, the user types:

```
cuc delete -all
```

or the equivalent:

```
cuc dl
```

Name: compact_mail_table, salvage_mail_table

SYNTAX AS A COMMAND

```
compact_mail_table {size}
```

FUNCTION

used by a system administrator to fix the mail table after it has been damaged or has become full. Because the mail table and the person name table (PNT) contain a parallel set of entries, any time the PNT needs to be compacted (using salvage_mstb) the mail table should also be compacted. When this is done, the old mail table is renamed to mail_table.MM/DD.HHMM.

ARGUMENTS

size
: specifies the size of the new mail table. If not specified, size defaults to the number of used entries (both regular and alias entries) multiplied by three, which is close to optimum for the accessing algorithm. In general, it is best to keep the mail table the same size as the PNT. If the size specified is not large enough to hold the existing entries, an error will be generated.

NOTES

This command should be run before the Answering Service is started by the multics or startup command.

salvage_mail_table is an alternate name.

*

Name: compare_mst

SYNTAX AS A COMMAND

```
compare_mst reel_id1 reel_id2 {-control_arg}
```

FUNCTION

reads two Multics system tapes (MSTs) and lists all differences between them.

ARGUMENTS

reel_id1, reel_id2
: are the reel identification numbers of the two tapes being compared. The reel identification number, which is site dependent, can be up to 32 characters long. The reel_id can also include a density specification to indicate the density of the tape being compared, as in "060341,den=1600".

CONTROL ARGUMENTS

–save
: saves the contents of corresponding segments with discrepancies in the user's working directory under the names tp1.<segment_name> and tp2.<segment_name>. An added segment is saved under the name tp2.<segment_name>.

NOTES

Differences in segment headers and the starting address of any inequalities or differing lengths of segment contents are noted. Additions, deletions, and moves of segments are handled. One can optionally save the contents of differing segments in the user's working directory for further detailed comparisons. Any number of collections can be handled, but a warning message is printed if a tape does not end in a collection mark. If the active_all_rings_data segment is found on the first tape, a message containing the system identifiers of both tapes is printed.

Name: compute_bill

SYNTAX AS A COMMAND

```
compute_bill sat_path pdt_dir
```

FUNCTION

is used by biller.ec to generate billing information as part of the monthly billing process. It totals the usage figures in the PDTs, the miscfile, and the projfile segments, and places the totals in the reqfile entry for each project. It is also used by crank.ec as part of the daily accounting job to update these same segments.

ARGUMENTS

sat_path
: is the pathname of the SAT: >udd>SysAdmin>admin>safe_pdts>sat.

pdt_dir
: is the pathname of the directory containing the PDTs: >safe_pdts. Generally this directory contains the copies of the PDTs that were copied from the live PDTs during an earlier part of the daily accounting job.

Entry: compute_bill$update_pdts_from_reqfile sat_path pdt_dir

FUNCTION

This entry is used by the daily accounting job to write the latest per-project information from the reqfile, projfile and SAT into the live PDTs.

ARGUMENTS

sat_path
    is the pathname of the SAT: >udd>SysAdmin>admin>safe_pdts>sat. |

pdt_dir
    is the pathname of the directory containing the PDTs: >safe_pdts. |

NOTES

The reqfile, projfile, and miscfile segments are assumed to be present in the working directory, and are implicit inputs to compute_bill.

Name: console_report

SYNTAX AS A COMMAND

```
console_report {as_log_paths} {-control_args}
```

FUNCTION

creates and displays metering of terminal usage on the system based on the information obtained from the answering service logs.

ARGUMENTS

as_log_paths
    are pathnames of answering service logs. The pathnames can be absolute or relative.

CONTROL ARGUMENTS

-print
    causes a display of the metering on user_output.

-report_reset, -rr
    causes a display of the metering on user_output and resets the metering in the data base.

-reset, -rs
    resets the metering in the data base.

-sort
    sorts the data base alphanumerically by terminal answerback identifiers.

NOTES

Control arguments and pathnames can appear anywhere in the command line. They are processed from left to right, one at a time.

The header in the display produced when the -print and -report_reset control arguments are used is obtained from the system titles. This header may be too wide for the user's terminal and the user may wish to use the file_output command.

The command creates and stores data into two segments, termseg and termuseg, in the current working directory when pathnames of answering-service logs are specified. These segments are the data bases that the command expects to find in the current working directory when any of the control arguments are used.

EXAMPLES

```
Multics terminal usage

Period from 08/26/84 1407.9 to 09/25/84 1600.2


Type        Count     Logins   Nologins        CPU   Connect

2741          247         28         17       0:05     25.22
TN300         395        632         48      18:47    604:18
Daemon        101        873          4      38:08   4456:06
ASCII        1077       9563       1423     531:45   7954:30
TELNET         13          0        324       0:00      0:00
LA36           93         60         11       0:34     15:40
VT100          41        419         45       6:10    367:53
LA120           5        112          9       2:24     89:51
```

This page intentionally left blank.

This page intentionally left blank.

```
ID  Type    Location    Logins    Nologins       CPU    Connect
    User

002  ASCII                 1           3        0:04       3:03
 Derek.Multics             1                    0:04       3:03

004  ASCII                 2           4        0:01       1:16
 Aulin.Multics             2                    0:01       1:16

405  LA120                73          26        0:46      75:54
 MKane.Network            38                    0:31      53:07
 Marcus.Network            8                    0:04       6:09
 Inada.Network            11                    0:07       5:53
 SBWeber.Network           9                    0:05       6:37
 Ducot.FlexMan             5                    0:01       3:43
 Yip.Network               1                    0:01       0:02
 Feinstein.Network         1                    0:01       0:27

40>  ASCII                 2           0        0:01       0:26
 Stanzel.ARCS              1                    0:01       0:03
 AWhite.ARCS               1                    0:01       0:24

 40>  VT100                2           0        0:01       0:38
 HSPP-BASIC.Student        1                    0:01       0:16
 Soley.ARCS                1                    0:01       0:23
etc.
```

Type
: is the terminal type.

Count
: is the number of different terminals of that type.

Logins
: are the number of completed logins.

Nologins
: the number of logins attempted and not completed.

CPU
: is the amount of CPU time used.

Connect
: is the amount of time actually logged in.

ID
: is an identification number assigned to a specific terminal.

Type User
: the terminal type and ID is shown on a heading, followed by a line for each user of the terminal giving usage statistics.

Location
: is not used.

Name: convert_configuration_deck

SYNTAX AS A COMMAND

```
convert_configuration_deck ascii_path binary_path
```

FUNCTION

converts an ASCII source form of a configuration deck, as produced by the print_configuration_deck command, into a binary (system) form.

ARGUMENTS

ascii_path
: is the pathname of an ASCII source form of a config deck. Both labeled and unlabeled fields may appear on the config cards. The archive convention is allowed.

binary_path
: is the pathname of the resultant binary form of the config deck. This form is compatible with the system config deck.

NOTES

This command is intended to be used to perform a level of validation on a proposed new ASCII config deck. It may also be used to convert an ASCII config deck into the form required by the compare_configuration_deck command.

Name: copy_as_meters

SYNTAX AS A COMMAND

```
copy_as_meters path
```

FUNCTION

copies the system metering information maintained by as_meter_ from the segment >sc1>stat_seg into a specified data segment, and resets the index in stat_seg.

ARGUMENTS

path
    is the pathname of the copy to be created.

NOTES

The statistics are placed in the segment path.

This command is executed by the crank (in master.ec), which is usually run once each day, to enable statistical tools to analyze the system's performance. The system_total command uses the segment created by the copy_as_meters command.

Name: copy_deadproc

SYNTAX

```
copy_deadproc {deadproc_name} {-control_args}
```

FUNCTION

sets up a dead process directory in preparation for use by the analyze_multics command. It copies a dead process directory specified by deadproc_name into the directory under the >dumps>save_pdirs directory. Several hardcore segments needed by analyze_multics are also copied into the directory. Two segments are created by the copy_deadproc tool, pdir_info and uid_hash_table. These are used by analyze_multics when examining a dead process. The dead process directory is renamed to person.pdir, where "pdir" is the standard suffix. If person.pdir already exists, it is renamed, before copying the new directory, to person.N.pdir, where N=1. If person.1.pdir already exists, it is renamed to person.N+1.pdir, and so on. Access to the new pdir is determined by the initial dir_acl of the save_pdirs directory.

ARGUMENTS

deadproc_name
: is the name of the dead process directory to be copied. If deadproc_name is not an absolute pathname, the default path is >process_dir_dir>deadproc_name. The names of dead process directories in the >process_dir_dir are of the form person.project.f.tty_name. A name of this form and the full name must be specified.

CONTROL ARGUMENTS

-delete, -dl
: deletes the original dead process. Status and modify access to the containing directory is needed. If access is lacking, the user is queried about whether to continue copying.

-name deadproc_name, -nm deadproc_name
: specifies the name of the process to be copied.

-no_delete, -ndl
: specifies that the dead process directory is not to be deleted after copying is complete. This is the default.

-owner, -ow
: specifies that access be set appropriately for the owner of the fatal process. This access is status on the dead process directory.

ACCESS REQUIRED

The use of this command requires access to phcs_. A user can copy his own process if he has "sma" on the save_pdirs directory and access to phcs_. If the dead process doesn't belong to the process doing the copying, access to the hphcs_ gate is required. When copying terminated processes of a different authorization level than the process doing the copying, access to the system_privilege_ gate is required.

Name: copy_dump

SYNTAX AS A COMMAND

```
copy_dump
```

FUNCTION

copy a dump image taken by BCE out of the DUMP partition into the Multics hierarchy. It creates as many segments (up to ten) in >dumps as necessary to hold the dump image.

NOTES

The name of each segment has the form

```
   mmddyy.tttt.s.eee
```

where

mmddyy
    is the date the dump was taken.

tttt
    is the time the dump was taken.

s
    is a sequence number (0, 1, 2,...9).

nnnn
    is the dump number assigned when recording this dump.

Entries: copy_dump$set_fdump_num, copy_dump$sfdn

```
Usage:  copy_dump$sfdn dmpno
```

FUNCTION

This entry point sets the value of the next dump to be taken by changing the value associated with the dump number in the DUMP partition.

ARGUMENTS

dmpno
    is the dump number for the next dump to be taken.

ACCESS REQUIRED

This command interfaces to hphcs_$copy_fdump and to hphcs_$set_fdump_num and requires access to hphcs_.

NOTES

The DUMP partition is modified only after the last dump taken has been copied. If you attempt to change the dump number before a dump has been copied, an error message is returned. |

This command does not allow a particular dump to be copied twice; therefore it returns an error code if you attempt to recopy a dump.

Name: copy_mst, cpm

SYNTAX AS A COMMAND

```
cpm reel_id1 reel_id2
```

FUNCTION

* copies a system tape (BCE/Multics bootload tape) onto another reel of tape.

ARGUMENTS

reel_id1
: is the reel identification number of the tape from which information is to be copied. The reel identification number, which is site dependent, can be up to 32 characters long. The reel_id can also include a density specification to indicate the density of the tape being copied, as in "060341,den=1600".

reel_id2
: is the reel identifier number of the tape onto which the copy is to be made.

*

Name: copy_registry

SYNTAX AS A COMMAND

```
copy_registry path1 {path2} {-control_arg}
```

FUNCTION

makes checkpoint copies of RCP Resource Management registries. You can use these copies as a basis for the reconstruction of registries destroyed by catastrophic system failure.

ARGUMENTS

path1
: is the pathname of the registry to be copied. If you give no rcpr suffix, it is assumed. You can use the star convention.

path2
: is the pathname of the copy to be created. The equals convention is accepted. If the suffix rcpr is not given, it is assumed. If to_path is not supplied, the copy will be placed in the working directory and will have the same name as the original. (See "Notes.")

CONTROL ARGUMENTS

–reset
: specifies that the contents of the registry journal are to be discarded after the copy operation has been successfully completed. (See "Notes" below.)

NOTES

The RCP Administrator must not copy registries into >sc1>rcp (for reconstruction purposes or otherwise) except under special session.

The registry journal contains a record of all operations performed against all registries since the time its contents were last reset via the use of the –reset control argument described above. Since a successful reconstruction operation depends on the journal containing a record of all operations performed since the copies of the registries were created, it is important that the –reset control argument only be specified for invocations which result in the copying of all registries. The copying of any number of registries and the resetting of the journal within one invocation of the copy_registry command is performed as an indivisible operation, which guarantees that no operations can be performed against any of the registries involved until the copying operation is complete and the journal has been reset. Since this cannot be guaranteed between multiple invocations of the copy_registry command, the –reset control argument should never be used without copying all active registries.

When –reset is specified, the journal is reset only if the copy operations are completed successfully.

Copies of system registries are automatically made each night by the system accounting facility (crank) using this command.

ACCESS REQUIRED

This command requires access to the rcp_admin_ gate.

EXAMPLES

To make checkpoint copies in the current working directory of all RCP Resource Management entries and discard journal entries made since the last checkpoint, type:

```
copy_registry ** -reset
```

Name: create_daemon_queues, cdq

SYNTAX AS A COMMAND

```
cdq {path} {-control_args}
```

FUNCTION

creates the I/O daemon queues. It determines which queues to create by examining the iod_tables segment.

ARGUMENTS

path
is the pathname of an iod_tables segment created by the iod_tables_compiler. The queues are created in the containing directory of path, using the request types specified by the iod_tables segment. This argument is optional.

CONTROL ARGUMENTS

-directory path, -dr path
queues are created in the directory whose pathname is path. This control argument is provided for testing purposes only; normally, it should be omitted. When not specified, the queues are created in the >daemon_dir_dir>io_daemon_dir directory. This argument may not be given with a path specification.

-reset_access
resets the ACLs on each queue to the default value, if the queue already exists.

NOTES

The I/O daemon tables segment, called iod_tables, is expected to be found in the same directory in which the queues are to be created. For each request type defined in iod_tables, one to four queues are created, depending on the maximum number of queues for that request type (as defined in iod_tables). The name of each queue is of the form XXX_N.ms where XXX is the associated request type name and N is the priority number of the queue. The ms suffix indicates that each queue is a message segment.

Name: create_mail_table

SYNTAX AS A COMMAND

```
create_mail_table
```

FUNCTION

used by system administrators to create the initial mail table. The mail table is initially populated with information taken from the person name table (PNT).

ACCESS REQUIRED

The user must have e access to the gate mail_table_priv_, r access to the PNT, and sma access to >site>mail_system in ring-2. In addition, the access class of the mail table will be the current authorization of the creator, so this user should be logged in at system_low (this may be ignored if the site does not use multiple AIM levels): the command will attempt to set dir and seg privileges, in which case the mail table will be created at system low, regardless of the authorization of the creator.

NOTES

Generally, this command should only be needed once, the first time the system is booted when it is delivered, or when the site first upgrades to MR10.2 (or a later release. The mail table was first implemented in MR10.2). It must be used after the PNT has been created, but before the system is brought up for regular service; the safest way is to run it before the Answering Service is started.

Name: create_pnt

SYNTAX AS A COMMAND

```
create_pnt pathname {-size N}
```

FUNCTION

create a pnt segment.

ARGUMENTS

pathname
: specifies the pathname of the pnt segment. If the ".pnt" suffix is not included in the segment name, it is automatically added.

CONTROL ARGUMENTS

-size N
: specifies that the pnt is to be created with sufficient space for N entries, where N is a decimal number. (A user registration uses one entry and an alias uses one entry.)

ACCESS REQUIRED

This command requires access to the pnt_fs_gate_.

NOTES

This command is automatically executed by the accounting startup exec_com (acct_start_up.ec) when a new Multics site is initialized. This command is useful for administrators who want to create an alternate pnt for testing purposes.

Name: create_urf

SYNTAX AS A COMMAND

```
create_urf path {-size N}
```

FUNCTION

creates a user registration file.

ARGUMENTS

path
: specifies the pathname of the urf segment.

CONTROL ARGUMENTS

-size N
: specifies that the urf is to be created with sufficient space for N entries, where N is a decimal number. A user registration uses one entry.

NOTES

This command is normally executed by the accounting startup exec_com (acct_start_up.ec) when the site is initialized.

Name: cv_cmf

SYNTAX AS A COMMAND

```
cv_cmf cmf_path {-control_args}
```

FUNCTION

converts an ASCII channel master file (CMF) into a binary channel definition table (CDT). The binary table can be installed using the install command.

ARGUMENTS

cmf_path
: is the pathname of the channel master file. If path does not have a suffix of cmf, one is assumed. However, the suffix cmf must be the last component of the name of the source segment. The channel master file is normally located at >udd>SysAdmin>admin>CMF.cmf.

CONTROL ARGUMENTS

-brief, -bf
: uses short form of error messages.

-long, -lg
: uses long form of error messages.

-severity N, -sv N
: causes error messages whose severity is less than N (where N is 0, 1, 2, 3, or 4) not to be written to the user_output switch. If this control argument is not specified, a severity level of 0 is assumed (i.e., all error messages are written to the user_output switch).

NOTES

If no control arguments are given, each error message is printed in long form the first time it occurs and in short form thereafter.

The converted channel master file is given a name corresponding to the entryname of the source segment, with the cmf suffix replaced by cdt. It is placed in the working directory.

LIST OF SEVERITY VALUES

The cv_cmf command associates the following severity values to be used by the severity active function:

Value	Meaning
0	No compilation yet or no error.
1	Warning.
2	Correctable error.
3	Fatal error.
4	Unrecoverable error.
5	Could not find source.

Name: cv_dmcf

SYNTAX AS A COMMAND

```
cv_dmcf path
```

FUNCTION

converts a configuration source file into a DMS configuration table, used to shape the run-time environment at DMS initialization. The source file is an ASCII segment composed of source statements as described below. The table must be installed by the administrator in the per-AIM directory to enable the DM daemon to boot DMS.

ARGUMENTS

path
: is the pathname of the configuration source file. The file must have the suffix .dmcf as part of its entryname. The suffix is added if not present.

NOTES ON SOURCE STATEMENTS

Source file statements take the form <keyword:> <value;>. Any number of statements can be specified in any order, but a keyword can only appear once. The last statement must be <end;>.

LIST OF KEYWORDS

system_before_journal_size N
: sets the size of the system default before journal to N control intervals. The number of transactions processing concurrently, their average time, and rate of modification are factors to be considered in establishing optimal journal size. Analyzing metering information and other performance criteria will enable you to fine-tune this value over time. The default is 4000.

max_processes N
: sets the maximum number of processes that can use Data Management concurrently to N. Unless there is reason to restrict the number of users, this value should be keyed to the number of processes that can log in to the AIM classification being configured for DMS. The default is whatever the load control for a given Multics configuration can support, up to 256.

max_transactions N
: sets the maximum number of transactions that can be in progress concurrently to N. Ideally, this value should equal the number of processes that can use DMS. The default is 128.

default_before_journal path
: specifies the location of the system default before journal. Path is composed of one or both of dir=DIR and entry=ENTRY. If both values are specified, they are separated by a comma. DIR is either an absolute pathname of the directory in which the default before journal is to reside, aim_dir (the per-AIM directory of the DMS being initialized), or bootload_dir (the DMS per-bootload directory). ENTRY is either the entryname of the default before journal or system_default.bj. The default is dir=aim_dir,entry=system_default.bj and is the recommended location of the system default before journal.

idle_timeout N
: sets the frequency of wakeup calls to the DM daemon to every N minutes. Upon receiving a wakeup call, the daemon performs its caretaker function of adjusting unfinished transactions. Independent of this mechanism, the daemon is on call (through process termination notifications, for example) to perform this function as needed. The idle timeout is provided to ensure that the daemon is periodically stirred to action in the absence of any other notification. The default is 15.

begin_shutdown_delay N
: sets the delay between the user warning and begin shutdown stages of DMS shutdown to N minutes. The default is 5.

user_shutdown_delay N
: sets the delay between the begin shutdown and user shutdown stages of DMS shutdown to N minutes. The default is 5.

user_bump_delay N
: sets the delay between the user shutdown and user bump stages of DMS shutdown to N minutes. The default is 5.

daemon_logout_delay N
: sets the delay between the user bump and daemon logout stages of DMS shutdown to N minutes. The default is 5.

prev_bootload_status list
: determines the disposition of the previous DMS per-bootload directory when initializing DMS in the same AIM directory. Options in the list are separated by commas; they are as follows:

    hold
    : specifies that the previous bootload directory is to be renamed and retained (usually for debugging or test purposes). The default is ^hold.

    adopt
    : specifies that if there is already a DMS bootload directory for the current Multics system bootload, the DM daemon should take it over. The implication is that the deamon process for that DMS invocation died. The default is adopt.

    recover
    : specifies that recovery of DM files associated with the previous DMS invocation should be attempted. The default is recover.

    recovery_check_mode
    : specifies that before journals are to be searched completely for uncommitted transactions. The implication is that there is a discrepancy between the number of indicated and actual transactions requiring recovery. This is a debugging tool, to be turned on only at the direction of Multics System Support . The default is ^recovery_check_mode.

    The default values in effect for these options are the recommended operational settings.

current_bootload_enable STR
: specifies whether to force enabling of the current invocation of DMS, despite being unable to recover from a crash of the previous invocation. STR is force to enable DMS whether or not there are errors in recovery, or ^force to enable DMS only if recovery was successfully completed. A value of force should be specified only at the direction of Multics System Support. The default is ^force.

log_proc_terms STR
: specifies whether the DM daemon is to log each occurrence of adjusting a transaction because the owner process has died. Some sites may consider this type of notification benign and wish to turn it off to conserve log space. STR can be on or off. The default is on.

NOTES

The output from this command is a segment table that takes the same entryname with the suffix .dmct. To install this table in the per-AIM directory, use the copy command, giving the new segment the name dm_configuration. See the section entitled "Data Management Administration" in the *Multics System Administration Procedures* manual, Order No. AK50, for details on installing and bringing up Data Management on Multics.

You should not attempt to install dm_configuration in a per-AIM directory when the DM daemon is initializing the system. A lockword is set in dm_configuration to prevent multiple daemons from attempting initialization with the same segment at the same time. If the copy overlaps initialization, this protection may be subverted, with unpredictable results. As a safeguard, perform the copy only when the daemon is logged out.

EXAMPLES

The sample source file is presented below as a guide to proper formatting. It is not intended as a suggested DMS configuration.

```
max_processes: 100;
max_transactions: 100;
system_before_journal_size: 2000;
default_before_journal: dir=bootload_dir;
begin_shutdown_delay: 10;
user_shutdown_delay: 10;
prev_bootload_status: hold,^adopt,recovery_check_mode;
log_proc_terms: off;
current_bootload_enable: force;
end;
```

Name: cv_pmf

SYNTAX AS A COMMAND

```
cv_pmf pmf_path {-control_args}
```

FUNCTION

converts an ASCII project master file (PMF) into a binary project definition table (PDT).

ARGUMENTS

pmf_path
: is the pathname of the PMF. If path does not have a suffix of pmf, one is assumed; however, the suffix pmf must be the last component of the name of the source. The PMF is normally stored in the hierarchy of the project administrator.

CONTROL ARGUMENTS

-brief, -bf
: prints error messages in the short format.

-long, -lg
: prints error messages in the long format.

-severity N, -sv N
: causes error messages whose severity is less than N (where N is 0, 1, 2, 3, or 4) not to be written to the user_output switch. If not specified, a severity level of 0 is assumed; i.e., all error messages are written to the user_output switch.

NOTES

The newly converted PDT is placed in the current working directory. The entryname of the new PDT is the same as the entryname of its source PMF, with the pmf suffix replaced by pdt. This command associates the following severity values to be used by the severity active function:

Value	Meaning
0	No compilation yet or no error.
1	Warning.
2	Correctable error.
3	Fatal error.
4	Unrecoverable error.
5	Could not find source.

EXAMPLES

```
cv_pmf >udd>Project_id>projfile
```

converts the PMF >udd>Project_id>projfile.pmt and creates a PDT named projfile.pdt in the project administrator's working directory.

Name: cv_prt_rqti

SYNTAX AS A COMMAND

```
cv_prt_rqti path {-control_arg}
```

FUNCTION

converts an ASCII printer request type info source segment into a printer request type info segment (rqti segment) for use by the I/O daemon. The newly converted rqti segment is placed in the current working directory. The entryname of the new rqti segment is the same as the entryname of its source segment without the rqti suffix. The ASCII source segment is created by the administrator. See the *Multics System Maintenance Procedures* manual, Order No. AM81 for additional information on the creation of rqti source segment.

ARGUMENTS

path
: is the pathname of the request type info source segment. The source segment must have a rqti suffix, although the suffix may be omitted in the command invocation.

CONTROL ARGUMENTS

-brief, -bf
: prints error messages in the short format.

-long, -lg
: prints error messages in the long format. This is the default.

EXAMPLES

The command line:

```
   cv_prt_rqti printer_info.rqti
```

creates a request type info segment named printer_info in the working directory.

Name: cv_rtmf

SYNTAX AS A COMMAND

```
cv_rtmf rtmf_path {-control_args}
```

FUNCTION

converts an ASCII resource type master file (RTMF) into a binary resource type description table (RTDT). The binary table is installed using the install command. If the user has made any errors in the RTMF, this command prints error messages while performing the conversion.

ARGUMENTS

rtmf_path
: is the pathname of the resource type master file. If path does not have a suffix of rtmf, one is assumed. However, the suffix rtmf must be the last component of the name of the source segment. The ASCII RTMF is normally located at >udd>SysAdmin>admin>RTMF.

CONTROL ARGUMENTS

-brief, -bf
: prints short form of error messages

-long, -lg
: prints long form of error messages

-severity N, -sv N
: causes error messages whose severity is less than N (where N is 0, 1, 2, 3, or 4) not to be written to the user_output switch. If this control argument is not specified, a severity level of 0 is assumed (i.e., all error messages are written to the user_output switch).

NOTES

If no control arguments are given, an error message is printed in long form the first time it occurs, and in short form thereafter.

The converted resource type master file is given a name corresponding to the entryname of the source segment, with the rtmf suffix replaced by rtdt. It is placed in the working directory.

The syntax of an RTMF is described in the *System Administration Procedures* manual, Order No. AK50.

EXAMPLES

The cv_rtmf command associates the following severity values to be used by the severity active function:

Value	Meaning
0	No compilation yet or no error.
1	Warning.
2	Correctable error.
3	Fatal error.
4	Unrecoverable error.
5	Could not find source.

Name: daily_summary

SYNTAX AS A COMMAND

```
daily_summary {-control_args}
```

FUNCTION

prints a summary of system usage for the current billing period and a list of projects with overspent accounts. It also places a flag in the SAT for each project that has an overspent account or has passed its termination date.

CONTROL ARGUMENTS

-nocutr
  do not print a list of projects to be cutoff.

-nosat
  do not flag SAT entries.

-nosum
  do not print a system usage summary.

-warn
  for all projects to be cutoff, use a warning flag that permits users to login.

NOTES

The segments reqfile and smf.cur.sat are assumed to be present in the working directory and are implicit inputs to the daily_summary command.

This command writes the system usage of each project in the reqfile on the sumry I/O switch. The usage figures are cumulative from the time that billing was last run. It also writes a list of projects that have run out of funds or are past the termination date and flags their SAT entries on the cutrpt I/O switch.

The information about the closeness of a project to its limits, that is used to decide whether to print a cutoff warning message when users on the project log in, is placed in the SAT by this command.

The cutrpt and sumry I/O switches must be attached by the caller.

EXAMPLES

The command line:

```
daily_summary -nosum -nocutr
```

should be used before each installation of the SAT, to update cutoff information. The entries in master.ec that install the SAT do this.

Name: deactivate_seg

SYNTAX AS A COMMAND

```
deactivate_seg path {-control_arg}
```

FUNCTION

allows you to deactivate a segment or directory.

ARGUMENTS

path
: is the pathname of the segment or directory to be deactivated or a segment number.

CONTROL ARGUMENTS

-force, -fc
: deactivates the segment or directory, if possible, by using the highly privileged demand_deactivate entry.

ACCESS REQUIRED

This command requires access to the phcs_ gate; if you use -force, it requires access to the hphcs_ gate.

NOTES

If you don't supply -force, the segment or directory is only deactivated if all processes connected to it have the allow_deactivate attribute set (see the change_kst_attributes command). If you give -force, the segment is deactivated unless its entry hold switch is set and the directory is deactivated unless it has active inferiors.

Name: debug_fnp, db_fnp

SYNTAX AS A COMMAND

```
db_fnp {request}
```

FUNCTION

is a debugging aid intended to be used by FNP software developers and in FNP dump analysis. You can use it to patch or dump memory in a running FNP, to examine a dump from a crashed FNP or a core image segment before it is loaded, to set breakpoints in a running FNP, and to display symbolically FNP control blocks, buffers, etc.

ARGUMENTS

request
: is the first request(s) to be executed. Once the initial request(s) is completed, debug_fnp reads request lines from user_input. Each line can contain multiple requests, separated by semicolons. If an error occurs in any request, the remaining requests on that line are not be executed. You can abort any request by typing "quit" followed by a "pi" (program_interrupt).

NOTES ON debug_fnp MODE SELECTION

This command can operate on a running FNP, a dump segment, or a core image segment; with few exceptions, most requests work the same regardless of the selection. When first invoked, the command is set up to examine the first configured FNP. You can switch between dumps, core images, and running FNPs at any time.

LIST OF MODE SELECTION REQUESTS

fnp
: Usage: fnp tag

  selects a running FNP, where tag can be "a", "b", "c", "d", "e", "f", "g", or "h".

image
Usage: image path

selects a core image, where path is the Multics pathname of a segment containing the core image. Core image segments and dump segments have a different format, so the image and dump requests are not interchangeable. The pathname on this request can also be a starname, provided it matches only one entry in the directory specified.

dump
Usage: dump path

selects a dump, where path is the Multics pathname of a segment containing the dump. Core image segments and dump segments have a different format, so the image and dump requests are not interchangeable. The pathname on this request can also be a starname, provided it matches only one entry in the directory specified.

In most cases, it is not necessary to know the pathname of the dump to be examined, as special requests are provided for selecting dumps.

dumps
Usage: dumps

lists all the dumps currently in the dump directory. The default dump directory is >dumps.

dump_dir
Usage: dump_dir {path}

changes the default dump directory, where path is the pathname of the new dump directory. If you omit "path," the name of the current dump directory is printed.

last_dump
Usage: last_dump

selects the latest dump.

prev_dump
Usage: prev_dump

selects the next earliest dump. You can use this request to peruse any or all the dumps in the dump directory, going in either direction. You can use it repeatedly as long as there are more dumps.

next_dump
    Usage: next_dump

selects the next latest dump. You can use this request to peruse any or all the dumps in the dump directory, going in either direction.

select_fnp
    Usage: select_fnp tag

selects which FNP in the dump is examined, when dealing with a dump that contains multiple FNPs, such a BOS fdump; tag can be "a", "b", "c", "d", "e", "f", "g", or "h".

what
    Usage: what

prints the fnp tag or the pathname, to find out what FNP, dump, or core image is selected.

NOTES ON EXPRESSIONS

Many of the following requests take numeric arguments such as addresses, lengths, etc. You can express any of these arguments as a generalized FNP expression. Expressions can be arbitrarily complex, containing "(", ")", "+", "-", "*", and "/" with their normal meanings and precedence. The symbol "|" is synonymous with "+", as in module|offset. You can specify indirection by ",*", following the address to indirect through. Numeric constants are interpreted as octal unless they are followed by a ".", in which case they are decimal. You can use "*" for the current location counter, which is generally the last address used in a display or patch request. You can also use many common FNP symbols, including all fields in the system comm region, the hardware comm region, the software comm region, and the TIB. (Before you can use TIB, hwcm, and sfcm addresses, establish the addresses of these control blocks. See the line and set requests.) A symbol can also be any opblock mnemmonic, the name of any FNP object module, or a machine instruction (enclosed within apostrophes). In addition, you can define user symbols. For example,

```
hsla|500
t.icp,*
*+30
tib|14,*+10
goto
'lda 0,2,b.0'
cax3   (apostrophe not needed if there is no operand)
```

LIST OF FNP MEMORY DISPLAY REQUESTS

display, d
    Usage: display address {length} {mode}
           d address {length} {mode}

displays the contents of FNP words, where address is the starting address, length is the number of words, and mode is the display mode. The symbol "*" is set to the address specified. You can use the following display modes:

```
octal, oct
character, ch
address, addr  (in form module|offset)
clock, ck  (4 FNP words as a Multics clock)
instruction, inst  (355 instruction format)
opblock, op  (pseudo-opblock format)
decimal, dec
bit
ebcdic, ebc
```

If you omit length, it defaults to 1 unless address is a predefined FNP symbol, in which case the appropriate length for that symbol is used. Similarly, if you omit mode, octal is used unless address is a predefined FNP symbol, in which case the mode appropriate for that symbol is used.

buffer, buf
    Usage: buffer {address} {mode} {-brief, -bf}
           buf {address} {mode} {-brief, -bf}

displays a buffer, where address is the address of the buffer, mode is the mode to display it in (see the display request), and -brief means display only the first two words of the buffer. If you omit address, the next buffer pointer from the previous buffer displayed is used. If you omit mode, character mode is assumed. If you omit -brief, the entire buffer is displayed. The length is determined automatically by reading the buffer header.

buffer_chain, bufc
    Usage: buffer_chain {address} {mode} {-brief, -bf}
           bufc {address} {mode} {-brief, -bf}

displays a buffer chain.

block, blk

Usage: block {address} {-offset, -o offset} {-length, -l length}
blk {address} {-offset, -o offset} {-length, -l length}

displays a control block at the address specified. Use this request if the data being displayed is in the form of threaded control blocks. The default offset to the forward pointer in the block is 0. The default block length is eight words. If you specify no address, the next block in the chain is displayed (using the forward pointer from the previous block).

block_chain, blkc

Usage: block_chain {address} {-offset, -o offset} {-length, -l length}
blkc {address} {-offset, -o offset} {-length, -l length}

displays the entire chain of control blocks until one with a zero forward pointer is encountered. Use this request if the data being displayed is in the form of threaded control blocks. The request follows the threads in the buffer chain, displaying each buffer. The default offset to the forward pointer in the block is 0. The default block length is eight words. If you specify no address, the next block in the chain is displayed (using the forward pointer from the previous block).

flags

Usage: flags address {type}

shows the setting of individual bits. Use this request if the data being displayed is a word of flags. The argument address is the the address of the word containing flags; type can be:

```
t.stat  tib status word
t.flg  first tib flag word
t.flg2  second tib flag word
t.flg3  third tib flag word
sf.flg  hsla sfcm flags
istat  interpreter status word
hs.1  first word of hsla hardware status
hs.2  second word of hsla hardware status
```

If you omit type, it is assumed to be the same as "address," which then must be one of the items in the above list. The flags are listed by name, as they appear in the macros.map355 source file. You can use the explain request for help on unfamiliar names. Occasionally, the value of a flag word is known (from a trace, for example), without knowing an address of it. In this case, you can use the following form:

```
flags =expression type
```

where expression is any valid expression and type is one of the types shown above.

LIST OF FNP MEMORY-PATCHING REQUESTS

patch

Usage: patch address arg1...{argn}

patches the contents of FNP memory, where address is the starting address to patch and argi represent patch data. Each argi can be an expression representing the value to be stored in one FNP word or a character string in quotes (which can contain more that one word of data). The total number of words patched cannot exceed 32. Before the patch is applied, the effects of the patch are displayed (old and new contents of every word) and you are asked to verify that the patch is correct. The symbol "*" is set to the address specified. Examples of patch requests are:

```
patch 43102 203456 -1 2
patch .crver "3.1x"
patch ctrl|1400 goto ctrl|1600
patch hsla|1541 'tze 13' cax3 'lda 0,2,b.1'
```

A shorthand form of this request is

```
=arg1...{argn}
```

which is equivalent to

```
patch * arg1...{argn}
```

set_flag

Usage: set_flag flag_symbol

sets the bit associated with the flag_symbol specified in the appropriate word. Use this request to manipulate individual flag bits in words of flags.

clear_flag
Usage: clear_flag flag_symbol

clears an individual bit. Use this request to manipulate individual flag bits in words of flags. This request is not an indivisible operation; this means if other flag bits in the word are dynamically changing, this request may change their value if they happen to have been changed between the time the word was read and when it was rewritten.

LIST OF DUMP ANALYSIS REQUESTS

The following requests are valid only when using debug_fnp on a dump.

why
    Usage: why

finds out the cause of a dump. It prints the type of fault that caused the crash and, if the crash was caused by a "die" opcode in the FNP, interprets the reason for the crash.

regs
    Usage: regs

prints the contents of all machine registers at the time of the fault.

call_trace
    Usage: call_trace address {-long, -lg}

starts at the address specified and perform a backward trace of all subroutine calls. The subroutine must be as defined by the map355 "subr" macro. If you give -lg, the registers saved at each subroutine level are also printed. You can also use this request on a running FNP, but the information is probably changing too fast for the request to be useful.

LIST OF FNP TRACE TABLE REQUESTS

A running FNP or a dump contains a trace table of the most recent events occurring in the FNP.

print_trace
    Usage: print_trace {start} {count}

displays the trace table, where start indicates the starting trace message and count is the number of messages to display. If you supply no arguments, the entire trace table is printed. If you give no count, the trace table is displayed from the starting point specified to the end. If the start number is positive, it is counted from the oldest message; if negative, it is counted from the most recent. For example,

```
print_trace 200.
```

skips the 199 oldest entries and print the rest.

```
print_trace -50.
```

prints the 50 most recent messages.

Note that the start and count, like all other numeric values, are octal unless followed by a decimal point. Printing the trace table of a running FNP is only meaningful if tracing has been suspended; otherwise the table is changing too fast to be interpreted.

stop_trace
    Usage: stop_trace

    suspends tracing in a running FNP.

start_trace
    Usage: start_trace

    restarts tracing in a running FNP.

Tracing can also be stopped and started with some of the breakpoint requests explained below.

Which modules in the FNP are traced is determined by the trace mask, kept in FNP memory.

trace_mask
    Usage: trace_mask {modules}

    examines or updates the trace mask. If you give no modules, trace_mask displays and interprets the current trace mask. If you give modules, they represent modules to be added to, or deleted from, the current mask. Specify the module as "name" or "+name" to set the tracing bit for the module and as "-name" or "^name" to turn off the corresponding bit. In addition, you can specify all and none. For example,

```
trace_mask hsla ^dia -util
```

    turns on tracing for hsla_man and turn off tracing for dia_man and utilities.

    Turning tracing on for a module for which tracing was not enabled when the core image was bound has no efect.

NOTES ON FNP BREAKPOINT FACILITY

The control table interpreter in the FNP allows breakpoints to be set in the interpreted control tables. A breakpoint causes the line encountering it to stop execution in the interpreter until you give a request to restart it.

You can use breakpoints only if the module breakpoint_man is included in the core image.

Breakpoints are often a useful tool, but be careful in their use. The following points are important:

1. Breakpoints can only be set in interpreted opblocks. They cannot be set at machine instructions.

2. While at a break, the line is executing an opblock equivalent to

   ```
   wait 0,0,0
   ```

   followed by no status blocks. This means that timers can run out unnoticed, status can be ignored, hangups can be missed, etc. For this reason it may be difficult to restart a channel after a breakpoint.

3. Breakpoints cannot be set at subroutine levels where waits would be illegal.

4. Breakpoints cannot be set when a restart may execute a wait opblock.

5. Breakpoints cannot be set at a status opblock.

6. If a breakpoint is set at a wait opblock, it must be reset before the line is restarted. In addition, a breakpoint cannot be set at a wait if any channels are currently waiting at that block.

7. Control tables that use local internal variables (as opposed to variables in the TIB extension) cannot depend on these variables being preserved during the break unless other channels that may use the same control tables are not running.

8. No notice is given when a channel encounters a breakpoint. The list_break request lists all breakpoints and shows what channels are stopped at each one.

LIST OF FNP BREAKPOINT FACILITY REQUESTS

set_break, sb
    Usage: set_break address -line- {-stop_trace}
           sb address -line- {-stop_trace}

sets a breakpoint at the address specified. If you give a tty channel, the breakpoint applies to that line only, and any other channel encountering the breakpoint continues execution; otherwise, the breakpoint applies to any channel that encounters it. If you supply -stop_trace, the FNP automatically suspends tracing if any channel stops at that breakpoint.

reset_break, rb
    Usage: reset_break address, rb address
           reset_break -all, rb -all

resets a break at the address specified. Any lines stopped at the break are not automatically restarted. If you give -all, all breaks are reset.

start, sr

```
Usage:  start line {address} {-reset} {-start_trace}
        sr line {address} {-reset} {-start_trace}
        start -all, sr -all
```

restarts the line specified. If you supply an address, the line restarts at the address given, instead of where it was stopped. If you supply -reset, the break is reset before the line is started. If you supply -start_trace, tracing resumes as the line is restarted. If you supply -all, all lines at breakpoints at the time you issue the request are restarted.

list_break, lb

```
Usage:  list_break
        lb
```

lists all FNP breakpoints and the channels stopped at each.

LIST OF PERFORMANCE ANALYSIS REQUESTS

The FNP software periodically samples the instruction counter to determine whether the FNP is running or idling.

idle_time

```
Usage:  idle_time {-reset, -rs}
```

prints the percentage of time the FNP has been idling since bootload or the last time you invoked the request with -rs.

sample_time

```
Usage:  sample_time {new_time}
```

prints or sets the sampling interval used by the FNP for metering data, where new_time is the new sampling interval in milliseconds (between 1 and 1000). If you supply no argument, the current sampling interval is printed. (Default when the FNP is booted: 50 milliseconds)

You can collect more detailed information on FNP usage by configuring the module "ic_sampler" in the FNP core image. This module periodically samples the instruction counter (at the rate set by sample_time) and adds 1 to a bucket, which represents a small range (typically 16) of FNP addresses. With this data you can determine with some precision where the FNP is spending its time when it is running.

ic_sample

This request is only accepted if the ic_sampler module is configured in the FNP. It has the following options:

Usage: ic_sample start

starts the ic sampling feature. Sampling is normally disabled when the FNP is booted.

Usage: ic_sample stop

stops ic_sampling.

Usage: ic_sample reset

zeroes all the sampling buckets.

Usage: ic_sample module

prints a table showing each module in the core image and what percentage of samples collected occur in that module.

Usage: ic_sample histogram, hist {fraction}

prints a histogram showing each bucket address that has data and the percentage of nonidle time that bucket represents. The fraction must be a floating point number between 0.0 and 1.0. With this option, the histogram only contains the most frequently used buckets. Enough buckets are printed so that the fraction specified of the total data collected is printed. For example, if the fraction is .9, 10% of the data collected is not displayed by discarding infrequently referenced buckets. This option is useful in deleting "noise" from the histogram.

LIST OF MISCELLANEOUS REQUESTS

line

Usage: line {line}

locates the TIB, software comm region, and hardware comm region of the line specified. Once these addresses are set, you can reference fields in these control blocks by name in any expression in other requests. You can specify the line either in Multics form (a.h012) or as an FNP line number (1014). If you give no line, the name of the current line is printed. If the line selected is not on the current FNP, the proper FNP is selected automatically. You can specify the line as -login_channel, in which case your login channel is selected.

buffer_status, bstat

Usage: buffer_status {-brief, -bf}
       bstat {-brief, -bf}

prints a table showing each line and how much buffer space in the FNP it is using. If you use -bf, only summary information is printed.

set

Usage: set symbol value

sets a symbol, where symbol is "*", "tib", "hwcm", "sfcm", or any user-defined symbol. Setting control block addresses (tib, hwcm, sfcm) is more easily done with the line request, but can be manually done with this request in case internal FNP tables have been damaged. Note that setting any of these control block addresses has no effect on the current value of other control blocks. Setting "*" is also done by any dump or patch request. Once set, a symbol can be used in any expression in any other request.

map

Usage: map

displays a list of modules, their addresses in the core image, and their dates of compilation.

convert_address, cva

Usage: convert_address {address1} ... {addressn}
       cva {address1} ... {addressn}

displays the address in any other meaningful forms that can be derived. For example, octal values are converted to module|offset, and vice versa.

explain

Usage: explain sym1 {sym2} ... {symn}

finds the explanation of any FNP symbol (usally the output of a flags or convert_address request), where symi are symbols to be explained. It prints the comment from the line in macros.map355 that defined the symbol.

e

Usage: e command_line

executes any Multics command by passing "command_line" to the command processor.

quit, q

Usage: q

exits from debug_fnp.

Name: define_work_class, dwc

SYNTAX AS A COMMAND

```
dwc pct_wc1 {pct_wc2...pct_wcn}
```

FUNCTION

assigns percentages of CPU time to various work classes independently of what is specified in the master group table (MGT).

ARGUMENTS

pct_wci
: is the percentage of CPU time to be assigned to the ith work class. If pct_wci is zero, the ith work class is not defined. If you don't specify pct_wci (there is no ith argument), the ith work class is not defined.

NOTES

The effect of dwc is temporary, since the answering service defines the work classes specified in the MGT at the next shift change or the next time the operator issues the "maxu auto" command. The percentages are ignored if the scheduler is in deadline mode.

The percentages specified are normalized to a sum of 100% by the hardcore scheduler.

This command does not allow work classes defined in the MGT to be undefined; it can create, however, undefined work classes. No changes made by dwc are recorded in the MGT. No work class containing active processes can be undefined. You can use dwc to change all work classes out of real-time mode, although you can also use the tune_work_class command after dwc to change some work classes into real-time work classes.

You can use this command in the following cases:

1. While testing or tuning the priority scheduler, to rapidly change the percentages assigned to different work classes.

2. If the percentages specified in the MGT are temporarily out of balance, to handle the transient condition gracefully without installing a new MGT.

3. To establish special or ad hoc work classes, at the administrator's discretion, for the duration of a shift. You can then use the set_work_class command to move certain processes into the new work class.

EXAMPLES

The following three command lines are equivalent and define work classes 1, 2, 3, 4, and 6, each with 20% of available CPU time.

```
    dwc 20 20 20 20 0 20

    dwc 20 20 20 20 0 20 0

    dwc 1 1 1 1 0 1
```

Assume work classes 1, 2, and 3 already exist with 50%, 30%, and 20% of CPU time, respectively. To temporarily create a special work class and give it approximately 10% of available CPU time, while holding the relative amounts of CPU time received by the other work classes constant, type:

```
   dwc 50 30 20 10
```

The actual effect is to give the four work classes 46%, 27%, 18%, and 9%, respectively.

Name: delete_mail_table_entry

SYNTAX AS A COMMAND

```
delete_mail_table_entry name
```

FUNCTION

removes entries from the mail table that are not associated with a registered user.

ARGUMENTS

name
: is a string that specifies a mail table entry that is to be deleted. Enclose it in quotation marks if it contains blank spaces.

NOTES

In order to maintain consistency with the person name table (PNT), this command will not permit entries which are associated with registered users to be deleted. If the mail table entry has aliases then the alias entries are also deleted.

ACCESS REQUIRED

The user must have e access to the gate mail_table_priv_.

Name: delete_old_pdds

SYNTAX AS A COMMAND

```
delete_old_pdds {-control_args}
```

FUNCTION

deletes old copies of >process_dir_dir and system_library_1 created during bootload. This command is intended mainly for use in the Utility.SysDaemon's start_up.ec. If the system is unable to boot for lack of quota, boot to standard and run from the initializer process in admin mode.

CONTROL ARGUMENTS

-exclude_first N
: specifies that the first N old copies of `>process_dir_dir` and `>system_library_1` (that is, the N oldest ones) are not to be deleted.

-exclude_last N
: specifies that the last N old copies of `>process_dir_dir` and `>system_library_1` (that is, the N most recent ones) are not to be deleted.

ACCESS REQUIRED

This command requires access to the hphcs_ gate.

NOTES

The old copies of `>process_dir_dir` are named `pdd.[unique]`, and branch directly off the root. The old copies of `>system_library_1` are named `sl1.[unique]`, and also branch directly off the root.

The control arguments for specifying that some old `>process_dir_dir` copies are not to be deleted are useful when it is necessary to have the process directory contents of processes at the time of a crash, in order to debug system problems.

If the process running delete_old_pdds has access to the soos privilege, it will be enabled to allow salvaging of soos directories. It is recommended that the
* delete_old_pdds command be run in a process with access to the soos privilege.

Name: delete_proj

SYNTAX AS A COMMAND

```
delete_proj Project_id
```

FUNCTION

is used by the dproj command in the master.ec segment. It does a portion of the work of deleting a project. (Other commands called by master.ec do the remainder.)

ARGUMENTS

Project_id
    is the name of the project to be deleted.

NOTES

The entries for the specified project are marked as deleted in the administrator's copy of the SAT (smf.cur.sat) and in the projfile and reqfile segments. Those entries are not physically removed however. In addition, the name "delete.Project_id.pdt" is added to the Project_id.pdt segment in >sc1>pdt as a reminder that this PDT should be deleted (by the system administrator) after the next monthly bills are run.

Name: delete_registry

SYNTAX AS A COMMAND

```
delete_registry paths
```

FUNCTION

deletes old unused RCP Resource Management registries. These registries must have been previously removed from service via the remove_registry command.

ARGUMENTS

path
    is the pathname of an old registry to be deleted. The star convention is accepted. If the suffix old is not given, it is assumed.

ACCESS REQUIRED

You need access to the rcp_admin_ gate.

Name: delete_volume_registration, dvr

SYNTAX AS A COMMAND

```
dvr -control_args
```

FUNCTION

deletes the registration information for a physical or logical volume.

CONTROL ARGUMENTS

You can supply only one of these two control arguments:

-logical_volume LV_NAME, -lv LV_NAME
    specifies that the registration information for the given logical volume is to be deleted.

-physical_volume PV_NAME, -pv PV_NAME
    specifies that the registration information for the given physical volume is to be deleted.

NOTES

You can't delete the last physical volume registered to a logical volume. You must delete the whole logical volume registration.

When deleting a logical volume, all physical volumes in that logical volume are deleted.

There is no reason to delete and reregister volumes. If there is an error in registration, use change_volume_registration to correct it.

ACCESS REQUIRED

You need re access to the mdc_priv_ gate and sm to >lv.

Name: deregister_resource, drr

SYNTAX AS A COMMAND

```
drr type STR1...STRn
```

FUNCTION

makes a particular resource unknown to the system. The deregistration process informs the system that the resource is no longer available for use.

ARGUMENTS

type
: is a resource type defined in the resource type description table (RTDT).

STRs
: is the unique identifying name of the particular resource being deregistered. If you specify STR with a leading hyphen, precede it by -name (-nm).

ACCESS REQUIRED

You require execute access to the rcp_admin_ gate.

NOTES

To be deregistered, the resource must be in the free state. A resource owned by a user (or belonging to the system pool) must be released (see the release_resource command in the Commands manual, AG92) before you deregister it.

If you give multiple resource names to drr and an error occurs in the deregistration of any of these resources, none of the resources are deregistered.

Name: disk_meters

SYNTAX AS A COMMAND

```
disk_meters {subsystem_name} {-control_args}
```

FUNCTION

prints statistics for each disk subsystem. A disk subsystem is a set of channels and devices where every channel can reach every device.

ARGUMENTS

subsystem_name
: is the name of a disk subsystem (dska, dskb, dskc, etc.) for which information is to be printed. If not specified, information for all subsystems is printed.

CONTROL ARGUMENTS

-channels, -chn
: requests subsystem channel information.

-detail, -dtl
: requests detailed drive information.

-long, -lg
: requests all information obtained by the -channels, -detail, -queues, -system control arguments.

-queues, -q
: requests drive queue information.

-report_reset, -rr
: generates a full report and then resets the metering interval to begin at the current point in time.

-reset, -rs
: resets the metering interval to being at the current point in time. No report is generated. A reset is accomplished by making a copy of the statistics as of the reset time. Future invocation of the command will display the difference between current statistics and the copy.

-system, -sys
: requests system statistics and optimizing information.

-unreset, -urs
: resets the metering interval to begin at system initialization time.

ACCESS REQUIRED

This command requires access to phcs_ or metering_gate_.

NOTES

The following are brief descriptions of each of the variables printed by disk_meters.

call locks
: are lockings of a disk subsystem in order to queue read or write operations. Information displayed includes the number of lockings (Count), the number of waits for the lock (Waits), the percentage of lockings that needed to wait (%Waits), and the average wait time (Avg. Wait (ms.)).

run locks
: are lockings of a disk subsystem in order to check the status of all devices and channels. Run is called by page control on all subsystems when too many write transfers are pending. Information displayed is the same as for call locks. If the number of run lockings is greater than a tenth of the number of call lockings, a transfer bottleneck probably exists at the subsystem, channel, or device level.

interrupt locks
: are lockings of a disk subsystem at interrupt time. Information displayed is the same as for call locks.

allocations
: are metered whenever a transfer request is queued within a disk subsystem. Information displayed includes the number of allocations (Count), the number of times queueing of the request had to wait for the completion of a previous request, i.e., no free queue entry was available for use (Waits), the percentage of allocations which had to wait (%Waits), and the average time in milliseconds until a queue entry became available (Avg. Wait (ms.)). If the allocation %Waits is greater than 2%, a transfer bottleneck probably exists at the subsystem, channel, or device level.

For each drive in the subsystem, the following meters are displayed.

Reads
: is the number of page or volume table of contents (VTOC) reads from the device.

Writes
: is the number of page or VTOC writes from the device.

Seek Distance
: is the average seek distance in cylinders on the device. If this average is greater than a third of the number of cylinders, then the pages are probably poorly distributed on the pack.

connects
: number of channel connections made.

int w/o term
: number of interrupts without terminate status.

run polls
: number of IO's seen by RUN polling.

get_io w/o term
: number of io_manager calls not returning terminate.

term not active
: number of interrupts with terminate on an inactive channel.

IOI
: the channel has been released to IOI use.

INOP
: the channel is deemed to be inoperative.

BROKEN
: the channel is deemed to be broken.

EXAMPLES

The following is an example of the information printed when the command is invoked with the -channels control argument specified.

```
dska:   Channel information.
A8:     42530 connects, 163 int w/o term, 385 run polls
A12:    4559 connects, 13 int w/o term, 129 run polls
A9:     1307 connects, 15 int w/o term, 225 run polls
```

The following is an example of the information printed when the command is invoked with the -detail control argument specified.

```
dska_04 #Seeks AveSeek Queue-wait Channel-wait Queued Multiplier
 PageRd 11338   70.38       44.5  0.8%    27.3      0      119.8
 PageWt  1482   51.98     1153.7  0.1%    32.4      0    50239.2
 VtocRd  1712   90.71       26.3  0.1%    24.4      0       23.8
 VtocWt  1518   89.38       26.4  0.1%    20.8      0       49.9
 TEST    0 UNLOADs,  39 TESTs
Channels 1.05% busy,  212 Combs.
```

The following is an example of the information printed when the command is invoked with the -queues control argument specified.

```
dska_04 Queue: Ave 16.1, Alloc 99, Max Depth 50/280, Cur Depth 0
```

This indicates the average queue depth for the specified number of queue allocations, the maximum depth since max_depth_meters were last reset and the current depth in the queue. Requests are only queued if a drive is busy and/or it already has requests queued.

The following is an example of the information printed when the command is invoked with the -system control argument specified.

```
FREE Queue: Ave 6.2, Alloc 60237, Max Depth 99/280, Cur Depth 1
System Factors: stagnate time 5.000 seconds, 378 bail outs.
   Maximum Depth Meters reset at: 06/07/84  2207.7 mdt Thu
   PageRd   Max Load    6, Depth   0 (PageRd), Fraction 1.0000
   PageWt   Max Load  210, Depth   0 (PageWt), Fraction 1.0000
   VtocRd   Max Load    6, Depth   0 (VtocRd), Fraction 1.0000
   VtocWt   Max Load   12, Depth   0 (VtocWt), Fraction 1.0000
```

This indicates FREE Queue use, stagnation time beyond which the system does disk combing and the number of times that the ALM driver had to call the PL1 driver to process complex interrupt information. The time that max_depth meters were last reset at is given, as is the current status of the system-wide load optimization algorithm.

The following is an example of the information printed when the command is invoked with no control arguments.

```
                    Metering Time:      11:23:01

Subsystem  dski:
                Locks    Waits    %Calls  Average      %CPU
  Call Lock:    11768        0   0.0000%    0.000  0.00000%
  Int Lock:     11764        0   0.0000%    0.000  0.00000%
  Run Lock:      3091        0   0.0000%    0.000  0.00000%
  Alloc Lock:   11755        3   0.0255%   12.223  0.00009%

     Drive  Reads     Writes      Seek     ATB       ATB      ATB
                                Distance  Reads    Writes     I/O

       1     3064      1831        68    13375     22382     8372
       2      729        36         9    56216   1138375    53570
       3     3065      1934        77    13370     21190     8197
       4      922        26        11    44448   1576212    43229
```

This indicates the metering period, the sub-system and lock information for the sub-system, and individual drive IO information for all drives which have performed IO in the metering period. Typically 0 counts are suppressed to highlight useful information.

Name: disk_queue, dq

SYNTAX AS A COMMAND

```
dq subsytem_name
```

FUNCTION

prints out the subsystem channel activity and the waiting I/O requests queued for a given secondary storage subsystem.

ARGUMENTS

subsytem_name
    is the subsystem name and can be of the form dska, dskb, dskc, etc.

ACCESS REQUIRED

This command requires access to phcs_ or metering_gate_.

NOTES

The disk_queue command accepts only one subsystem name at a time; therefore, multiple subsystem names should be enclosed in parentheses. For the subsystem specified, the number of connects on each of its channels is printed. For each waiting request, the following information is printed:

P
    is the priority of the request (1=high; 0=low).

RW
    indicates the type of request (R=read; W=write).

VP
    indicates the type of request (P=page; V=volume table of contents entry).

DV
: represents the device to which the request is directed.

SECTOR
: is the sector to or from which input/output is done.

MEM
: is the main memory address to or from which input/output is done.

Name: disk_stat_print

SYNTAX AS A COMMAND

```
disk_stat_print {path} {-control_args}
```

FUNCTION

prints the disk_stat segment that is created by the sweep command. Optional control arguments cause the information to be presented in a variety of ways. to facilitate analysis of disk usage patterns. By default, a header is printed. followed by one line for each directory in the disk_stat segment. and a totals line.

ARGUMENTS

path
: is the pathname of the disk_stat file to be printed. If path is not given, the disk_stat segment in the working directory is assumed.

CONTROL ARGUMENTS

-level N, -lev N
: summarizes each subtree that begins at level N: prints no lines with level numbers greater than N (where N is an integer from 0 to 16). The effect of this argument is to make the output appear as if no directories with quotas existed below level N in the hierarchy since the directories below level N have their usage figures included in those of whatever level N directory they are inferior to.

  The default value for N is 16, causing all directories in disk_stat to be printed individually. The root's level is 0. A directory is said to be below level N if its level number is greater than N. A value of 2 for N causes the disk usage of each project to be displayed in a single line; a value of 3 causes the usage of each user to be displayed (provided that the user directories have quotas).

-logical_volume, -lv
: prints the name of the logical volume on which segments contained in each directory reside and prints the usage figures for each logical volume. An extra column is printed in the directory line, giving the logical volume index (lvix). This is merely the line number in the table of logical volume totals that is printed after the regular totals lines. An lvix of 0 indicates that the subtree summarized by the line contains segments that reside on more than one logical volume; an lvix of -1 indicates a logical volume not known to the system.

-subtotal, -stt
: prints subtotal lines giving the totals for each subtree. Each time a directory is encountered whose level number is less than that of the preceding directory, one or more subtotal lines are printed. This argument causes the maximum amount of subtotal information to be printed. Some users may find the resulting output too cluttered to be easily read. To produce less information, but in an easier to read format, see the -level control argument.

-total, -tt
: does not print a line for each directory; rather prints a totals line (plus any other lines specified by other arguments).

NOTES

The first date printed in the header is the date of the last billing. It is filled in by charge_disk if the disk_stat file is the one used by the daily disk accounting job. This date is 0 in files created separately (e.g., for use in disk usage analysis).

Although the output of disk_stat_print is designed to be printed on a line printer (using the file_output and dprint commands, described in the *Multics Commands and Active Functions* manual, Order No. AG92), it can be printed on a terminal if the carriage is wide enough. Each line consists of approximately 10 characters followed by a pathname. Pathnames can be up to 168 characters long but are typically less than 50 characters long.

EXAMPLES

```
disk_stat_print Alpha.disk_stat -level 3
```

prints one disk usage line for each user on the Alpha project, assuming that the segment Alpha.disk_stat was created as shown in the example under the sweep command.

```
disk_stat_print Alpha.disk_stat -total
```

prints a single line giving the disk usage for the entire Alpha project.

Name: disk_usage_stat

SYNTAX AS A COMMAND

```
disk_usage_stat path
```

FUNCTION

gathers statistics on a hierarchy subtree. It produces a report describing: (1) the number of segments, links, directories, and names by hierarchy depth, (2) the distribution of segments by date of last reference and last modification,(3) the distribution by segment size, (4) the distribution by number of names, (5) the directory distribution by number of names, and (6) the counts by starname (see the *Multics Programmer's Reference Manual*, Order No. AG91, for a discussion of starnames).

ARGUMENTS

path
: is the pathname that specifies the hierarchy inferior to path to be scanned. The default is ">".

NOTES

The report is typed on the user's terminal.

The listing by starname is controlled by the contents of the segment starname_list in the user's working directory. This segment consists of starnames, one per line. If the starname_list segment does not exist, a single starname, **, which matches all entries, is assumed.

Name: disklow

SYNTAX AS A COMMAND

```
disklow {n_left {pct}}
```

FUNCTION

scans the projfile segment and reports on each project whose remaining disk storage (difference between quota and used) is low. The control arguments allow the system administrator to check the disk storage by either number of free records or percent of disk occupied or both.

ARGUMENTS

n_left
: prints a line for all projects with less than n_left free records, where n_left is an integer value.

pct
: prints a line for all projects occupying more than the specified percent of their disk storage quota. (pct is a value in the range 01 through 99).

NOTES

A value for n_left must be given in order to give pct. The default values for n_left and pct are 20 and 90, respectively. If the system administrator wants to receive only percentage information, n_left must be specified as 0.

The projfile segment is assumed to be in the working directory and is an implicit input to disklow.

Name: display_account_status, das

SYNTAX AS A COMMAND

```
das {Project_id} {-control_args}
```

FUNCTION

prints the latest accounting information for a project. The information is stored in the PDT of that project and is correct as of the last time the daily accounting job was run; it is usually run every night.

ARGUMENTS

Project_id
: is the Project_id of the project. If this argument is not given, the project under which the project administrator is currently logged in is assumed.

CONTROL ARGUMENTS

-brief, -bf
: prints a one-line summary of the account information.

-long, -lg
: prints all information found in the projfile (project registration segment) entry and the reqfile (requisition segment) entry.

-no_header, -nhe
    suppresses printing of the header.

ACCESS REQUIRED

The user must have read access to the PDT to use this command; usually only project administrators have such access.

NOTES

If neither the -brief nor -long control argument is given, all information about charges is printed.

See also the proj_usage_report command to get a brief summary of each user's resource consumption and the print_pdt command to get more detailed information about each user.

EXAMPLES

The following example illustrates the output of the das command when it is used with the -long control argument:

```
das Multics.pdt -long


                    Multics.pdt                 12/04/84  0855.3

                    Account information copied  12/03/84  2130.7

Projectid:          Multics;

REQFILE
 account:           SYSPROG
 reqno:             Multics
 rate structure:    default
 qflag:
 date on:           03/13/75  1254.4 est Thu
 date off:
 billing name:      System Administration
 billing addr:      Accounting Office
 charge this month: $     57285.74
 charge this req:   $   2414583.92
 req amount:        $         0.00     balance:  OPEN
 cutoff date:       01/01/99  1254.4 est Fri

PROJFILE
 title:             Worker Bees
 investigator:      John Gintell
 inv address:       Right Here
 supervisor:        CTC
 sup addr:          There
 sup phone:         phone_no
 date on:           03/13/75  1254.5 est Thu
 date off:
 disk page-months:  29296,  $ 87.89
 disk quota:        67538
 disk use:          27329
 dir disk use:      2214
 misc charges:      $ 0.00
 # misc charges:    0

SAT
 attributes:        primary_line, guaranteed_login, anonymous, nopreempt,
                    dialok, multip, bumping, brief, vinitproc, vhomedir,
                    nostartup, daemon, igroup, save_pdir, disconnect_ok;
 ring:              1,5
 alias:             m
 default group:     SysProg
 authorized groups: *,Admin
 max grace:         2880
 audit flags:       seg_init, dir_init, mc_seg_init, no_access, ipr_fault,
                    acv_mode, acv_ring, no_wakeup, sys_priv
 authorization:     system_low : system_high
 days to cutoff:    5143
 percent balance:   100%
 dollars to cutoff: $ 0.00
```

The following example illustrates the output of the das command without the -long argument:

```
das Multics.pdt

                    Multics.pdt                 12/04/84  0857.7

                    Account information copied  12/03/84  2130.7

Projectid:          Multics;
 charge this month: $     57285.74
 charge this req:   $   2414583.92
 req amount:        $         0.00     balance:  OPEN
 cutoff date:       01/01/99  1254.4 est Fri
 disk page-months:  29296,  $ 87.89
 disk quota:        67538
 disk use:          27329
 dir disk use:      2214
 misc charges:      $ 0.00
 # misc charges:    0
```

Name: display_anst

SYNTAX AS A COMMAND

```
display_anst {channel_name} {-control_args}
```

FUNCTION

displays the Answer Table, which contains information about interactive users. The Answer Table resides in >sc1>answer_table.

ARGUMENTS

channel_name
names a channel whose Answer Table entry is to be displayed.

CONTROL ARGUMENTS

-active
displays entries in Answer Table for active communications channels (channels that have been hung up, are listening for a user to dial up, are connected to a user (dialed), have logged in a user, or are attached to a running a process). This is the default.

-all, -a
displays all table entries.

-channel channel_name, -chn channel_name
names a channel whose Answer Table entry is to be displayed. The channel_name argument has the format described above.

-dial
displays entries in Answer Table for dialed communications channels (channels that are connected to a user (dialed), have logged in a user, or are attached to a running process).

-in
displays entries in Answer Table for logged in users.

-lock
displays entries in Answer Table which are locked.

-name Person_id, -nm Person_id
displays entry or entries associated with the given Person_id.

-no_header, -nhe
suppresses display of the heading information contained at the beginning of the Answer Table.

-octal, -oc
    displays header and entries in an octal dump, as well as in interpretive format.

-pathname path, -pn path
    gives the pathname of the table to be displayed. If omitted, the system Answer Table in >sc1>answer_table is used.

NOTES

The -all, -active, -dial, -in, -lock, -name and -channel control arguments and the channel_name argument select entries to be displayed. If more than one is given, an entry is displayed if it matches any of the conditions specified by the arguments. If none are given, the entries for all active communications channels are displayed.

EXAMPLES

To display the entire Answer Table, type:

```
display_anst
```

To display only the Answer Table entry for user Apple on any project, type:

```
display_anst -nhe -name Apple
```

To display the Answer Table entry for the user on channel "a.1008", type:

```
display_anst -nhe a.1008
```

Name: display_aste

SYNTAX AS A COMMAND

```
display_aste {pathname} {-control_args}
```

FUNCTION

displays the Active Segment Table Entry (ASTE) and, optionally, the file map of an active segment.

ARGUMENTS

pathname
    is the pathname of the desired segment. If a pathname is not specified, the -hardcore control argument must be used.

CONTROL ARGUMENTS

-hardcore NAME, -hc NAME
: display the ASTE and, optionally, the file map of the hardcore segment indentified by NAME, where NAME is the octal segment number or SLT name of the desired segment.

-offset OFFSET, -at OFFSET
: specifies that the ASTE at sst_seg/OFFSET will be displayed.

-file_map, -fm
: specifies that the file map will be displayed.

-no_file_map, -nfm
: specifies that no file map will be displayed. This is the default.

-no_octal, -noc
: specifies that the ASTE is not to be displayed in octal format. This is the default.

-octal, -oc
: specifies that the ASTE is to be displayed in octal format.

ACCESS REQUIRED

This command requires access to ring_zero_peek_, phcs_, and mdc_.

EXAMPLES

```
display_aste >

ASTE for > at 76530 in sst_seg
  fp:  7664, bp:  150764, infl:      0, infp:    7520
  strp:  30300, par_astep:      0, UID:  777777777777
  msl:  205, csl:  6, records:  6, np:  5
  DTU:  1984-01-23 08:44:18
  DTM:  1984-01-23 09:02:09

ON:     usedf gtms nqsw dirsw master_dir tqsw(S) tqsw(D) dnzp fms
OFF:    init gtus hc hc_sdw any_access_on write_access_on
        inhibit_cache explicit_deact_ok deact_error hc_part fm_damaged
        dius nid ehs volmap_seg npfs ddnp synchronized fmchanged
        fmchanged1 damaged pack_ovfl

  quota:  (137171, 147361), used:  (11141, 1019)
  pvtx:  7, vtocx:  0
  pvname:  rpv disk name:  dska_07


display_aste > -file_map
```

```
ASTE for > at 76530 in sst_seg
  fp: 76644, bp: 150764, infl:     0, infp:    7520
  strp: 30300, par_astep:    0, UID:  777777777777
  msl: 205, csl: 6, records: 6, np: 5
  DTU: 1984-01-23 08:44:18
  DTM: 1984-01-23 09:02:49

ON:    usedf gtms nqsw dirsw master_dir tqsw(S) tqsw(D) dnzp fms
OFF:   init gtus hc hc_sdw any_access_on write_access_on
       inhibit_cache explicit_deact_ok deact_error hc_part fm_damaged
       dius nid ehs volmap_seg npfs ddnp synchronized fmchanged
       fmchanged1 damaged pack_ovfl

  quota:  (137171, 147361), used:  (11141, 1019)
  pvtx:  7, votcx:  0
  pvname:  rpv disk name:  dska_07

File mape:  0 Memory address 16704000 ^er,phu,phul,^pmh,^phml,^wired,^os,valid
     Disk address:  Disk page    14504
 1 Memory address 12342000 ^er,phu,phul,^pmh,^phml,^wired,^os,valid
     Disk address:  Disk page    14506
 2 Memory address 12336000 ^er,phu,phul,^pmh,^phml,^wired,^os,valid
     Disk address:  Disk page    14510
 3 Memory address 12340000 ^er,phu,phul,^pmh,^phml,^wired,^os,valid
     Disk address:  Disk page    14512
 4 Memory address 12344000 ^er,phu,phul,^pmh,^phml,^wired,^os,valid
     Disk address:  Disk page    14514
 5 Disk page    42021
 6 Null address from fill_page_table
```

Name: display_aut

SYNTAX AS A COMMAND

```
display_aut {absN} {-control_args}
```

FUNCTION

displays the Absentee User Table (AUT), which contains information about absentee users. The Absentee User Table resides in >sc1>absentee_user_table.

ARGUMENTS

absN
    names the absentee process whose Absentee User Table entry is to be displayed (e.g., abs1, abs2).

CONTROL ARGUMENTS

-active
: displays entries in Absentee User Table for absentee processes that are currently running a job (this is the default).

-all, -a
: displays all table entries.

-lock
: displays entries in Absentee User Table which are locked.

-name Person_id, -nm Person_id
: displays entry or entries associated with the given Person_id.

-no_header, -nhe
: suppresses display of the heading information contained at the beginning of the Absentee User Table.

-octal, -oc
: displays header and entries in an octal dump, as well as in interpretive format.

-pathname path, -pn path
: gives the pathname of the table to be displayed. If omitted, the system Absentee User Table in >sc1>absentee_user_table is used.

NOTES

The -all, -active, -lock and -name control arguments and the absN argument select entries to be displayed. If more than one is given, an entry is displayed if it matches any of the conditions specified by the arguments. If none are given, the entries for absentee processes which are currently running a job are displayed.

EXAMPLES

To display the entire Absentee User Table, type:

```
display_aut
```

To display only the Absentee User Table entry for user Apple on any project, type:

```
display_aut -nhe -name Apple
```

To display the first absentee process, type:

```
display_aut abs1
```

Name: display_branch

SYNTAX AS A COMMAND

```
display_branch {-control_arg} target
```

FUNCTION

prints out information about directory entries that is not returned by the status command and lists the segment UID and the location of the branch. It does not access the VTOCE of the segment for any information.

ARGUMENTS

target
: indicates the segment whose branch is to be displayed. Any of the following forms can be used to specify the target segment: the pathname of the segment, the octal segment number of the segment, or the octal pointer representation of the branch address to be displayed (e.g., 260|1664).

CONTROL ARGUMENTS

-name, -nm
: must be specified before any pathname that is, or that can be construed as, a valid octal number.

ACCESS REQUIRED

This command requires access to the phcs_ gate.

Name: display_cache_threshold

SYNTAX AS A COMMAND

```
display_cache_threshold {-control_args}
```

FUNCTION

displays the current cache error threshold values.

CONTROL ARGUMENTS

-system
: displays the threshold values in the per-system data base.

-per_process
: displays the per-process threshold values. This is the default.

Name: display_cdt

SYNTAX AS A COMMAND

```
display_cdt {channel} {-control_args}
```

FUNCTION

enables a qualified user to display the contents of a channel definition table (CDT).

ARGUMENTS

channel
: is the name of the communications channel for which the CDT entry is to be displayed. The star convention is allowed.

CONTROL ARGUMENTS

-all, -a
: displays names and CDT indices for all channels in the CDT.

-brief, -bf
: displays only channel names and CDT indices (without channel or FNP details). This is the default for the -all and -subtree control arguments.

-cmf path
: creates a CMF in the segment named path in a form suitable to cv_cmf, based on the contents of the CDT.

-header, -he
: displays the CDT header variables in addition to other requested information.

-long, -lg
: displays detailed information for the specified channel or FNP. This is the default unless -all or -subtree is specified, in which case -brief is the default.

-no_header, -nhe
: suppresses display of the CDT header variables. This is the default.

-pathname path, -pn path
: displays the CDT whose pathname is path. By default, the CDT in the segment >sc1>cdt is displayed.

-subtree
: displays the names and CDT indices for all subchannels (if any) of the specified channel.

NOTES

If display_cdt is specified with no channel name and no control arguments, a usage error notification is returned. Specifying channel name only, with no control arguments, results in a -long display.

The display_cdt command enables the user to check for inconsistencies in a CDT before unnecessarily undertaking corrective action.

The user must have r access to the CDT to invoke the display_cdt command.

EXAMPLES

To display the entire system CDT, specify:

```
display_cdt -all -header
```

To display the system CDT entry for channel a.h000 only, specify:

```
display_cdt a.h000
```

To display all the subchannels of multiplexer a.h026.1, specify:

```
display_cdt a.h026.1 -subtree -long
```

Name: display_cpu_error

SYNTAX AS A COMMAND

```
display_cpu_error {-control_args}
```

FUNCTION

scans the syserr log and displays machine conditions and history registers.

CONTROL ARGUMENTS

-from DT, -fm DT
: starts scanning the log at the date/time given.

-to DT
: stops scanning the log at the date/time given.

-for T
: computes the ending time from the starting time, where T is a relative time (such as "1hour").

-cpu CPU_LIST
: displays information for the CPUs specified, where CPU_LIST is a list of CPU tags (e.g., "a c").

-nothread
: specifies that the history registers are not to be threaded. The history registers will be output in octal with no interpretation. The default is off.

-expand, -exp
: specifies that the history registers are not to be threaded but that they are to be interpreted.

-match STR...STRn
: matches strings against messages in the log, where STR is a text string. Messages are defined to contain process or machine condition information. Any message that contains a STRi is displayed.

-exclude STR1...STRn, -ex STR1...STRn
: matches strings against messages in the log as -match, where STR is a text string. Any message that contains a STRi is NOT displayed.

-all
: specifies that all log entries that are defined to contain processor machine information can be displayed.

ACCESS REQUIRED

Read permission is required on the log segments themselves and status permission is required on their containing directories.

NOTES

If -from DT is not given, the scan starts with the earliest entry in the syserr log. The ending time can be specified by using -for or -to, but not both. If both are omitted, the scan terminates with the last entry in the log. All dates and times must be in a format acceptable to the convert_date_to_binary_ subroutine described in the *Multics Subroutines and I/O Modules* manual, Order No. AG93.

Unless the control arguments -all, -match, or -exclude are specified, only hardware_fault entries will be processed.

Name: display_disk_label, ddl

SYNTAX AS A COMMAND

```
ddl DEVICE {-control_args}
```

```
ddl {PVNAME} {-control_args}
```

FUNCTION

displays information recorded in the physical volume label for a storage system disk volume and optionally displays information recorded in the physical volume table entry (PVTE) for the associated disk unit.

ARGUMENTS

DEVICE
: specifies the disk subsystem and unit on which the volume is mounted (e.g., dska_05 or dskk_00a).

PVNAME
: is the physical volume name (e.g., rpv).

CONTROL ARGUMENTS

-long, -lg
: displays information recorded in the PVTE.

-pvid PVID
: specifies the disk unit by the unique identifier assigned to the physical volume when it was registered (PVID), a 12-digit octal number.

ACCESS REQUIRED

You need access to phcs_.

NOTES

Specify a disk unit by only one of the following: DEVICE, PVNAME, or "-pvid PVID". The requested unit must represent a mounted storage system volume.

EXAMPLES

```
    ddl fpdir01 -long

  Label for Multics Storage System Volume fpdir01 on dskk_00a 3380

  PVID                  535341556672
  Serial                fpdir01
  Logical Volume        fpdir_1
  LVID                  535341556533

  Subvolume a 1 of 2
  Registered            05/24/86  2031.3 mst Sat
  Dismounted            11/07/86  2105.9 mst Fri
  Map Updated           11/08/86  0558.6 mst Sat
  Salvaged
  Bootload              11/08/86  0556.6 mst Sat
  Reloaded
  Dumped
    Incremental
    Consolidated
    Complete

  Inconsistencies                 0

  Minimum AIM                     0:000000
  Maximum AIM                     7:777777

  PVTE for Multics Storage System Volume fpdir01 on dskk_00a 3380
    at pvt|5374

  PVID                  535341556672
  LVID                  535341556533

  VTOCEs
    Number                 10204
    Left                   10131

  Records
    Number                 51024
    Left                   49434

  Subvolume Info
    sv_num                     0
    num_of_svs                 2
    record_factor              0
    records_per_cyl          127

  Inconsistencies              0
```

```
Volume Map
  volmap_seg ASTE    102|1120
  record stock       104|1440
  Page 0 - Base         11766
           Free         70712
  Page 1 - Base        105766
           Free         47520
  Page 2 - Base        205766
           Free             0
  vtoce stock        104|1703

ON:         used is_sv storage_system

OFF:        root_lv rpv permanent testing being_mounted being_demounted
            removable_pack check_read_incomplete device_inoperative
            scav_check_address deposit_to_volmap being_demounted2
            pc_vacating vacating hc_part_used volmap_lock_notify
            volmap_idle_notify vtoc_map_lock_notify dmpr_in_use(incr)
            dmpr_in_use(cons) dmpr_in_use(comp)

Volume Map from PVTE

First Record       Size
          0o         10o     Label   Region
         10o      11756o     VTOC    Region
      11766o     143520o     Paging Region
    1555060o     155704o     Partitions
                 1555060o    Total  Size
```

Name: display_dut

SYNTAX AS A COMMAND

```
display_dut {source_name} {-control_args}
```

FUNCTION

displays the daemon user table (DUT), which contains information about daemon users. The system DUT resides in >sc1>daemon_user_table.

ARGUMENTS

source_name
: is the message coordinator source name associated with the daemon process whose DUT entry is to be displayed (e.g., cord, prta, vinc, and vcomp).

CONTROL ARGUMENTS

-active
    displays entries in the DUT for all daemon processes that currently exist. (Default)

-all, -a
    displays all table entries.

-lock
    displays entries in the DUT that are locked.

-name Person_id, -nm Person_id
    displays the entry or entries associated with the given Person_id.

-no_header, -nhe
    suppresses display of the heading information contained at the beginning of the DUT.

-octal, -oc
    displays the header and entries in an octal dump, as well as in interpretive format.

-pathname path, -pn path
    gives the pathname of the table to be displayed. If omitted, the system DUT is used.

NOTES

The -a, -active, -lock, and -nm control arguments and source_name select entries to be displayed. If you give more than one, an entry is displayed if it matches any of the conditions specified by the arguments. If you give none, entries for active daemon processes are displayed.

EXAMPLES

To display the entire DUT, type:

```
display_dut -all
```

To display DUT entries for the daemon processes that currently exist, type:

```
display_dut
```

To display only the DUT entry for user IO on any project, type:

```
display_dut -nhe -name IO
```

To display the DUT entry for the process using the message coordinator source name vinc, type:

```
   display_dut vinc -nhe
```

Name: display_fnp_idle

SYNTAX AS A COMMAND

```
display_fnp_idle {fnp_names} {-control-args}
```

FUNCTION

displays information on the Front-End Network Processor (FNP) idle time stored by the meter_fnp_idle command. The display can be either a summary or a line graph (histogram).

ARGUMENTS

fnp_names
: are the names of the FNPs for which idle time information is to be displayed. If you supply no fnp_names, the display covers all FNPs for which information has been stored.

CONTROL ARGUMENTS

-directory path, -dr path
: specifies that information is to be taken from segments in the directory with pathname path (see meter_fnp_idle). (Default: to display information from idle time segments in the working directory)

This page intentionally left blank.

-from DT, -fm DT
: specifies that the display is to cover a period beginning no earlier than the date/time value (DT). (A description of valid DT values is provided in the *Multics Programmer's Reference Manual*, Order No. AG91.) The default is to start the display from the most recent idle time segment.

-histogram, -hist
: causes output in the form of a histogram, where a line shows the busy percentage for each FNP at a given time interval. The -histogram and -summary control arguments are mutually exclusive, but one or the other must be specified.

-interval N
: specifies that each line in the histogram represents an N minute interval. This control argument is ignored if -summary is specified. The default is 15 minute intervals.

-line_length N, -ll N
: specifies the line length of the histogram as N columns (N cannot be less than 38). This control argument is ignored if -summary is specified. The default is the user's terminal line length (or 80 if output is directed to a file).

-summary, -sum
: requests a summary display of FNP idle information for the specified time period. The -summary and -histogram control arguments are mutually exclusive, but one or the other must be specified.

-to DT
: specifies that the display is to cover a period ending no later than the date/time value (DT). (A description of valid DT values is provided in the *Multics Programmer's Reference Manual*, Order No. AG91.) The default is to end the display with the latest available information.

EXAMPLES

The following command and resultant display represent a sample display_fnp_idle summary request in which FNP "a" was found to be 90.3% idle for the specified period (i.e., starting from the beginning of the most recent idle time segment and including the latest available information). Note that the busiest interval within the period is identified.

```
display_fnp_idle a -summary

FNP A idle from 01/04/84 0600. to 01/06/84 1436.: 90.3%
Busiest sample interval:
01/05/84 1423. to 1424.: 43.7% idle
```

The following command and resultant display represent a sample display_fnp_idle histogram request for all FNPs for which idle time has been recorded. The time period starts no earlier than noon on 01/06/84 and includes the latest available information. The busy percentage for each FNP is given at 10-minute intervals.

```
display_fnp_idle -from "01/06/84 1200." -hist -interval 10

     %busy        0     10     20     30     40     50     60
                  |      |      |      |      |      |      |
01/06/84  1330.      A     B  C
          1340.          A   C    B
          1350.   A            CB
          1400.   A             B
          1410.      A          C
          1420.          A    B  C
          1430.            A B                              C
```

Name: display_ioi_data

SYNTAX AS A COMMAND

```
display_ioi_data {-control_args}
```

FUNCTION

displays the ring0 data base ioi_data. It can be used on a running system, an fdump, or a copy of ioi_data made with the copy_out command.

CONTROL ARGUMENTS

-all, -a
: causes all the channel table entries (cte's) and device table entries (dte's) associated with the group(s) selected to be displayed. This control argument can be used in conjunction with -group or -gte.

-force, -fc
: forces the display of certain control blocks and/or fields that the command might not otherwise display. For example, "-gte" displays only allocated group table entries (gte's); "-gte -force" displays all gte's, allocated or not.

-header, -he
: causes the ioi_data header to be displayed. This is the default if no control blocks are selected.

-no_header, -nhe
suppresses the display of the ioi_header. This is the default when a control block is selected.

CONTROL ARGUMENTS FOR SELECTING

The following control arguments are used to select where ioi_data is to be found. Only one control argument can be selected from this list. If none are specified, ioi_data is copied from ring_0 of the running system.

-erf erfno
specifies the number of the fdump to be analyzed.

-segment path, -sm path
specifies the pathname of the segment containing ioi_data. Normally, this segment is obtained from the running system using the copy_out command or from an fdump using the extract command.

CONTROL ARGUMENTS FOR DISPLAYING

The following control arguments specify which control blocks in ioi_data are to be displayed. Only one control argument can be selected from the following list. If none of these control arguments are specified, all control blocks are displayed.

-channel {channel_name}, -chn {channel_name}
displays the channel table entry (cte) for the channel specified. If channel name is omitted, all cte's are displayed. The channel name argument is in the form {tag}number, where tag is an IOM tag (a through h) and number is an octal channel number. If tag is omitted, IOM a is assumed.

-cte {octal_offset}
displays the cte at iom_data|octal_offset. If offset is omitted, all cte's are displayed.

-device {device_name}, -dv {device_name}
displays the device table entry (dte) for the device specified. If device_name is omitted, all dte's are displayed.

-dte {octal_offset}
displays the dte at iom_data|octal_offset. If offset is omitted, all dte's are displayed.

-group {device_name}, -gp {device_name}
displays the group table entry (gte) for the device specified. If device_name is omitted, all gte's are displayed. The device_name can be either the full name of a nonmultiplexed device, such as prta, or the full name or first four characters in the name of a multiplexed device, such as tape or tape_02.

-gte {octal_offset}
: displays the gte at iom_data|octal_offset. If offset is omitted, all gte's are displayed.

-user {Person_id.Project_id}
: displays the dte's of all devices assigned to the specified user. If Person_id.Project_id is omitted, the user's Person_id is assumed. Either Person_id or Project_id can be omitted or an asterisk (*) can be used in their place. This argument is not allowed for -erf and will not work with -segment if the segment was created during a previous Multics bootload or if the users have since logged out.

ACCESS REQUIRED

To use this command on a running system, access to the phcs_ gate is required.

NOTES

The default action of this commmand, when invoked with no arguments, is:

```
!  display_ioi_data -group -all -header.
```

Name: display_kst_entry

SYNTAX AS A COMMAND

```
display_kst_entry {-control_arg} target
```

FUNCTION

prints the contents of a KST (known segment table) entry. The KST entry to be dumped can be indicated by either a segment number or a relative pathname of the associated object.

ARGUMENTS

target
: is either a segment number or a relative pathname.

CONTROL ARGUMENTS

-name, -nm
: must appear if target is a relative pathname that looks like a segment number.

EXAMPLES

```
! display_kst_entry start_up.ec

        segno:      256  at  155|470
        usage:      0, 0, 0, 0, 2, 0, 0, 0
        entryp:     243|5452
        uid:        033100743603
        dtbm:       416334652254
        mode:       7 (4, 4, 4)
        ex mode:    000000000000 (0, 0, 0)
        infcount:   0 .
        hdr:        4
        flags:      write
```

Name: display_log_segment

SYNTAX AS A COMMAND

```
display_log_segment log_segment_ptr {-control_args}
```

FUNCTION

displays the internal information of a log segment.

ARGUMENTS

log_segment_ptr
: is a virtual pointer to the log segment to be displayed. It must be in a format acceptable to cv_ptr_ (this includes log segment entrynames and pathnames). You can supply only a log pointer.

CONTROL ARGUMENTS

-brief, -bf
: selects a short form for output.

-header, -he
: displays the log header. (Default)

-long, -lg
: selects a long form for output. (Default)

-no_header, -nhe
: does not display the log header.

-no_print, -npr
    does not displays the status and contents of message entries. (Default)

-no_trace
    does not display the status of messages. (Default)

-print, -pr
    displays the status and contents of message entries. The information displayed is a superset of that displayed by -trace.

-trace
    displays the status of message entries.

NOTES

This command is meant for analysis. Log perusal should be performed with monitor_sys_log, print_sys_log, or summarize_sys_log.

If you use -nhe, -npr, and -no_trace together, you get an error message and no display.

Name: display_mstb

SYNTAX AS A COMMAND

```
display_mstb path
```

FUNCTION

displays information about the status of a multisegment table (MSTB).

ARGUMENTS

path
    is the pathname of the multisegment table. The PNT and URF are multisegment tables.

EXAMPLES

The command

```
! display_mstb >sc1>PNT
```

displays

```
   MSTB >sc1>PNT is of type PNT.
   9 components, 10718 entries, 3943 used, 28 deleted,
                               6747 free.  (37% full)
   entry size = 45, header size = 16, component size = 65536.
```

Name: display_process_audit_flags

SYNTAX AS A COMMAND

```
display_process_audit_flags {-control_args}
```

FUNCTION

displays the current state of the process security audit control flags.

CONTROL ARGUMENTS

-brief, -bf
: specifies that the short format is to be displayed. (Default)

-long, -lg
: specifies that the long format is to be displayed.

ACCESS REQUIRED

You must have re access to access_audit_gate_.

NOTES

For a description of the process audit flags, see the new_user command in this section and the section "Security Auditing" in the *System Administration Procedures* (AK50).

See also the set_process_audit_flags and edit_process_audit_flags commands.

EXAMPLES

```
   ! display_process_audit_flags
     fsobj=N/N,fsattr=MA/MA,rcp=R/R,admin=R/R,special=R/R,...
        ...other=MA/R,admin_op,priv_op,fault,small_cc,moderate_cc
```

```
! display_process_audit_flags -long
  Object                     Grant Level      Deny Level

  File_System_Object         No_Audit         No_Audit
  File_System_Attribute      Modify_Access    Modify_Access
  RCP_Object                 Read             Read
  Administrative_Object      Read             Read
  Special_Object             Read             Read
  Other_Object               Modify_Access    Read

  Audited Events:
                   Administrative_Operation
                   Privileged_Operation
                   ACV/IPR_Fault
                   Small_Covert_Channel
                   Moderate_Covert_Channel
```

Name: display_psp

SYNTAX AS A COMMAND

```
display_psp {-control_args}
```

FUNCTION

displays selected information about distributed software found installed in online system libraries. The information includes marketing identifier, Software Technical Identifier (STI), copyright notice, and titles for the software requested.

CONTROL ARGUMENTS

-all, -a
: returns selected information of all products found installed in the system libraries. (Default)

-brief, -bf
: prints only the STI. (Default)

-copyright
: returns the copyright notice for selected products if found installed in the system library.

-long, -lg
: prints the marketing identifier, STI, copyright notice, and titles selected.

-match STR
: returns selected information for a specific product if it is installed in the system library, where STR is the marketing identifier.

-name, -nm
: returns selected information about a named product. The name of the product will be the long name by which the product is most commonly referred, i.e., compose, cobol, or ted.

NOTES

The -brief and -long control arguments are mutually exclusive; the -match, -name, and -all control arguments are mutually exclusive.

Name: display_pvte

SYNTAX AS A COMMAND

```
display_pvte {dskX_NN} {-control_args}

display_pvte {PVNAME} {-control_args}
```

FUNCTION

prints information recorded in the Physical Volume Table Entry (PVTE) for a storage system disk volume and optionally displays information recorded in the physical volume label.

ARGUMENTS

dskX_NN
: specifies the disk subsystem and unit on which the volume is mounted (e.g., dska_07).

PVNAME
: is the physical volume name of the disk volume (e.g., rpv).

CONTROL ARGUMENTS

-long, -lg
: causes information from the volume label to be printed also.

-pvid PVID
: specifies the disk volume by the unique identifier assigned to the volume at the time it was registered (PVID). PVID is a 12-digit octal number, which can be obtained by using the list_volume_registration (lvr) command. This control argument cannot be used if either dskX_NN or PVNAME is specified.

ACCESS REQUIRED

This command requires access to metering_gate_. If the -long control argument is specified, then access to phcs_ is required.

NOTES

The disk volume specified must be a mounted storage system volume.

Name: display_rtdt

SYNTAX AS A COMMAND

```
display_rtdt {type1 ... typeN} {-control_args}
```

FUNCTION

displays a resource type description table (RTDT).

ARGUMENTS

typei
: is the name of a resource type whose RTDT entry is to be displayed. If type is not given, all RTDT entries are printed.

CONTROL ARGUMENTS

-no_header, -nhe
: does not print the RTDT header comment.

-pathname path, -pn path
: displays the contents of the RTDT specified by path. If this control argument is not specified, the RTDT residing in >system_control_1 is displayed.

Name: display_syserr_log_part

SYNTAX AS A COMMAND

```
display_syserr_log_part {args} {-control_args}
```

FUNCTION

displays portions of the syserr logging partition that exist on the disk, in order to diagnose and correct problems that might occur in the syserr logging partition.

ARGUMENTS

header, he
: prints the syserr log partition header.

check D, ck D
: checks message threads in direction specified by D and validates message formats. Direction is specified by one of the following:

```
forward, f    forward checking only
reverse, r    reverse checking only
```

message N, msg N
: displays a single message with message header information, message text and expanded binary output, and octal message words. The message to be displayed is specified by appending a positive or negative integer to the message argument:

```
 N   displays the log message that is N from the top
-N   displays the log message that is N from the last
```

CONTROL ARGUMENTS

-offset ADDR, -ofs ADDR
: displays message at word offset ADDR from the beginning of the syserr log.

-number N, -nb N
: displays the syserr log message whose message number is N (decimal).

ACCESS REQUIRED

Read permission is required on the log segments themselves and status permission is required on their containing directories.

Name: display_system_audit_flags

SYNTAX AS A COMMAND

```
display_system_audit_flags
```

FUNCTION

displays the system parameters that control auditing of user access to system resources.

NOTES

See also the set_system_audit_flags command.

Name: display_vtoce

SYNTAX AS A COMMAND

```
display_vtoce PATHNAME|PVNAME VTOCX -control_args
```

FUNCTION

displays the vtoce of a specified segment.

ARGUMENTS

PATHNAME
: is interpreted as the pathname of a segment or directory whose VTOCE should be displayed if you supply one noncontrol argument.

PVNAME
: if you give two noncontrol arguments, the first is the physical volume name of the segment or directory whose VTOCE should be displayed.

VTOCX
: if you specify two noncontrol arguments, the second is the octal vtoce index of the segment or directory whose VTOCE should be displayed.

CONTROL ARGUMENTS

-file_map, -fm
: displays the VTOCE file map.

-header, -he
: displays the commonly used attributes of the segment stored in the VTOCE. (Default)

-long, -lg
displays the commonly used attributes (as in -header) as well as additional attributes with less commonly used information.

-no_file_map, -nfm
does not display the VTOCE file map.

-no_header, -nhe
does not display the commonly interesting attributes of the segment stored in the VTOCE.

-no_octal
does not dump the entire VTOCE in octal.

-octal
dumps the entire VTOCE in octal.

ACCESS REQUIRED

You need access to phcs_.

EXAMPLES

```
display_vtoce >
vtoce ">" (Directory), vtove 0 on pvtx 7 (rpv) - 01/23/84 0908.9 est Mon

UID = 777777777777, msl/csl/rec = 205 6 6
      Quota (S D) = (137171 147361)
      Quota used (S D) = (11413 1006)
      Quota received (S D ) = (260543 0)
      Time-record product (S) 7.99551e14 page-seconds
           updated at 01/21/84 1801.2 est Sat.
Created      05/21/76  1846.3 est Fri
Dumped       01/23/84  0844.3 est Mon
Used         01/21/84  1801.2 est Mon
Modified     01/21/84  1801.1 est Sat

Switches:  nqsw master_dir dnzp gtpd fm_checksum_valid

Access class:  system_low


display_vtoce > -long

vtoce ">" (Directory), vtove 0 on pvtx 7 (rpv) - 01/23/84 0908.9 est Mon

display_vtoce >

 vtoce ">" (Directory), vtove 0 on pvtx 7 (rpv) - 01/23/84 0908.9 est Mon
```

```
UID = 777777777777, msl/csl/rec = 205 6 6
      Quota (S D) = (137171 147361)
      Quota used (S D) = (11413 1006)
      Quota received (S D ) = (260543 0)
      Time-record product (S) 7.99551e14 page-seconds
           updated at 01/21/84 1801.2 est Sat.
Created      05/21/76  1846.3 est Fri
Dumped       01/23/84  0844.3 est Mon
Used         01/21/84  1801.2 est Mon
Modified     01/21/84  1801.1 est Sat

Switches:  nqsw master_dir dnzp gtpd fm_checksum_valid

Access class:  system_low
UID path: >
Parent vtocx -1 -of rpv of LV root
Last incremental volume dumped on volume v-14
Last consolidated volume dumped on volume v-1231
Last complete volume dumped on volume v-2

display_vtoce > -file_map -nhe

vtoce ">" (directory), vtocs 0 on pvtx 7 (rpv) - 01/23/84 0910.3 est MMon

File Map:
   0 Record 14504
   1 Record 14506
   2 Record 14510
   3 Record 14512
   4 Record 14514
   5 Record 42021
   6 Undocumented null address 777020
  ======
  64 Null address from update_vtoce
  ======
 255 Null address from update_vtoce
```

Name: dm_lock_meters

SYNTAX AS A COMMAND

```
dm_lock_meters {-control_args}
```

FUNCTION

displays metering information about data management locking. The long version of the report provides information on locking for the entire system. The reset capability sets the meters to zero for the process only.

CONTROL ARGUMENTS

-brief, -bf
: selects the brief format for reporting on locking. (Default)

-long, -lg
: selects the long format for reporting on locking.

-report_reset, -rr
: reports the current meters and then resets them.

-reset, -rs
: resets the meters to zero without printing the report.

ACCESS REQUIRED

This command requires e (execute) access to dm_admin_gate_.

EXAMPLES

Two sample invocations of this command appear below. The first displays the brief format (default), the second, the long format.

```
! dm_lock_meters

  Total metering time:  3:09:51

  Calls to per system lock:     208.
  Calls to unlock all:          208.
  Locks by mode:
                           S       X       IS       IX      SIX
       FILES:              0     154        3       51        0
        CI'S:              0       0

  Waits for locks:              0.
  Deadlocks:
   Total Deadlock checks:       0.
   Deadlocks detected:          0.
```

```
! dm_lock_meters -lg

  Total metering time:  3:09:51


  Calls to per system lock:      208.
  Calls to unlock all:           208.
  Locks by mode:
                           S       X       IS      IX      SIX
   NEW FILES:              0      154       3      51       0
   OLD FILES:              0       0        0       0       0
                  ---------------------------------------------------
       FILES:              0      154       3      51       0

    NEW CI'S:              0       0
    OLD CI'S:              0       0
                  ---------------------------------------------------
        CI'S:              0       0

   Waits for locks:               0.
   Deadlocks:
   Total Deadlock checks:        0.
   Two or more transactions:     0.
   36 or less transactions:      0.
   36 to 72 transactions:        0.
   More than 72 transactions:    0.
   Deadlocks detected:           0.
     Self youngest:              0.
     Other youngest:             0.

  Lock segments:                 1. (max size 4096.)
  Block allocates:               416.
  Block frees:                   416.
```

Name: dm_lock_status

SYNTAX AS A COMMAND

```
dm_lock_status {-control_args}
```

FUNCTION

displays the status of all locks currently held or awaited by the current transaction or all transactions running under the current invocation of DMS.

CONTROL ARGUMENTS

-all_transactions, -atxn
    displays lock data for all transactions.

-control_interval_locks, -cilk
    displays control-interval-level locks. This is the default.

-current_transaction, -ctxn
    displays lock data for the transaction currently owned by the process. This is the default.

-file_locks, -flk
    displays file-level locks. This is the default.

-header, -he
    displays data from the header of the lock database.

-no_control_interval_locks, -ncilk
    suppresses the display of control-interval-level locks.

-no_file_locks, -nflk
    suppresses the display of file-level locks.

-no_header, -nhe
    suppresses the display of data from the header of the lock database. This is the default.

ACCESS REQUIRED

This command requires e (execute) access to dm_admin_gate_.

EXAMPLES

```
! dm_lock_status


  Pierret.Multics 010500236015  Transaction 5
     Lock  UID=135165413304 CI=4
       Owner Pierret.Multics 010500236015 Txn 5  Mode   X
     Lock  UID=135165413304 CI=3
       Owner Pierret.Multics 010500236015 Txn 5  Mode   X
     Lock  UID=135165413304 CI=2
       Owner Pierret.Multics 010500236015 Txn 5  Mode   X
     Lock  UID=135165413304 CI=0
       Owner Pierret.Multics 010500236015 Txn 5  Mode   S
     Lock  UID=135165413304 FILE
       Owner Pierret.Multics 010500236015 Txn 5  Mode  IX


! dm_lock_status -he

   lock_seg
      header
         lock=                  142970279124797489152
         n_lock_segments=       1
         lock_seg_size=         4096
         max_lock_segments=     100
         n_transaction_table_entries= 32
         n_hash_table_entries=  256
         hash_mask=             377o
         free_list_ptr
            seg_inx=            1
            offset=             578
         transaction_table_offset= 16
         hash_table_offset=     272
         n_lock_blocks=         588
         lock_array_offset=     566


   Spratt.Multics 017400343566  Transaction 3
        Lock  UID=000000000001 CI=-1
       Owner Spratt.Multics 017400343566  Mode  IX
```

Name: dm_send_request

SYNTAX AS A COMMAND

```
dm_send_request keyword {value}
```

FUNCTION

enables administrators to send requests to the data management (DM) daemon to perform actions it normally performs through internal interfaces. It is an efficient mechanism for administrators to activate operations that require access to dm_daemon_gate_.

ARGUMENTS

keyword can be any of the following:

adjust_tdt, adjtdt
: calls on the DM daemon to scan the transaction definition table (TDT) and to abort those transactions associated with dead processes and those that have been abandoned. This is the same function that the daemon performs as caretaker of the data management system (DMS) upon receiving a periodic wakeup call.

adjust_tdt_entry value, adjtdte value
: calls on the DM daemon to check on the specified transaction and abort it if it belongs to a dead process or if it has been abandoned; value is the transaction index associated with the TDT entry.

adjust_txn value, adjtxn value
: calls on the DM daemon to check on the specified transaction and abort it if it belongs to a dead process or if it has been abandoned; value is the unique transaction identifier assigned when the transaction was started.

kill_txn value, kill value
: calls on the DM daemon to expunge the specified transaction from the TDT without rolling it back or committing it; value is the unique transaction identifier assigned when the transaction was started. Any DM files modified by the transaction may be left inconsistent, so use this request only as a last resort.

new_proc
: calls on the DM daemon to do a new_proc. This request is useful for setting the daemon to use a newly installed version of DMS.

new_process_notifications, notify
: calls on the DM daemon to send a dm_shutdown_scheduled_ IPS to the process sending this request, which implies that the process logged in after the Multics shutdown was scheduled.

shutdown
calls on the DM daemon to schedule a DM shutdown coinciding with the scheduled Multics shutdown (see the dm_system_shutdown command). This keyword enables a Multics operator to shut down data management using the reply operator command. Administrators should use the dm_system_shutdown command, as it affords the full range of shutdown specifications.

ACCESS REQUIRED

You must have re access to dm_admin_gate_.

Name: dm_set_journal_stamps

SYNTAX AS A COMMAND

```
dm_set_journal_stamps
```

FUNCTION

sets the time stamp for all journals currently in use to the time at which the command is invoked.

ACCESS REQUIRED

You need re access to dm_admin_gate_.

NOTES

The effect of altering the time to the present subverts the write_ahead_log (WAL) protocol, which guarantees that modified control intervals of protected data management (DM) files are not written to disk until their before images have been safely stored in a before journal.

Without the protection imposed by the time stamp, sync-held pages (as modified control intervals are called when they are detained in main memory) can be flushed to disk prior to their before images. In the event of a system crash without emergency shutdown (ESD), some before images may not be available on rollback during recovery.

You may have to invoke this command if before journals become damaged, preventing transactions from being committed, or if there is an unusually large number of sync-held pages, stalling system performance. This command keeps the system running instead of shutting it down; use it only as a last resort.

Name: dm_set_system_dir

SYNTAX AS A COMMAND

```
dm_set_system_dir path {-control_args}
```

FUNCTION

sets the per-system directory for the user process, so that when you invoke Data Management for the first time in your process (setting off a first-reference trap), your process will automatically attach to this directory.

ARGUMENTS

path
: is the pathname of the directory to be used as the per-system directory for test purposes.

CONTROL ARGUMENTS

-brief, -bf
: suppresses a message confirming that the directory was setup as requested.

-long, -lg
: causes a message to be printed confirming that the directory was setup as requested. This is the default.

NOTES

The directory established by this command should only be used for test purposes. For normal operations, the per-system directory is >site>Data_Management.

If your process is already attached to a per-system directory, you will be instructed to do a user shutdown (dm_user_shutdown command) before you can successfully setup a system directory with this command.

Name: dm_system_shutdown

SYNTAX AS A COMMAND

```
dm_system_shutdown {-control_arguments}
```

FUNCTION

schedules a shutdown of Data Management. The DMS can be specified by the pathname of the per-system directory, i.e., the directory in which it resides. If no pathname is specified, a shutdown is scheduled for the DMS being used by the process from which the command is issued. All processes using this DMS are notified of the scheduled shutdown via the dm_shutdown_scheduled_ IPS signal.

A DMS shutdown can be scheduled to prevent further activity in the current invocation. It can also be used to reduce the time of recovery during the next DMS initialization.

A shutdown of DMS occurs in five stages, usually, but not necessarily, in the following sequence:

1. User Warning Time
   A warning message is printed in each process using the invocation of DMS to be shut down, as follows:

   ```
   Data management is shutting down; no new transactions
   will be allowed after <begin shutdown time>.  Please
   use the dm_user_shutdown command, or, at <user shutdown time>,
   a forced shutdown will be executed. Shutdown reason:  {STR}
   ```

2. Begin Shutdown Time
   The DM daemon (Data_Management.Daemon) prevents the start of any new transactions in the DMS invocation. This is considered the beginning of the DMS shutdown.

3. User Shutdown Time
   A user shutdown is executed in each process still active in the DMS invocation. A message is printed in each affected process, as follows:

   ```
   The data management system is shutting down; starting DM user
   shutdown.  Shutdown reason:  {STR}
   ```

4. User Bump Time
   The DM daemon bumps each process still using the DMS being shutdown.

5. Daemon Logout Time
   The shutdown completes with the logging out of the DM daemon.

The actual occurrence of each stage is governed by the following rules:

- If a control argument containing "_time" is specified, that stage is scheduled to occur at the specified time.
- Starting from each specified time, the times for subsequent stages are calculated using the specified or default delays, until a stage is reached which has a specified time.
- If user warning time is not specified, then times are calculated backward from the first specified time, using the specified or default delays, until the first stage is reached.
- If no times are specified, the user bump time defaults to the scheduled Multics shutdown time, and all other stages are calculated from there using the specified or default delays.
- If no times are specified and there is no Multics shutdown scheduled, an error message is returned.

CONTROL ARGUMENTS

-dm_system_directory PATH, -dmsd PATH
: is the pathname of the per-system directory in which the DMS to be shut down resides. There may be several invocations of DMS in the per-system directory; the one selected is the one for the current bootload at the AIM classification of the caller. If no pathname is specified, the DMS invocation in use by the process is scheduled for shutdown. If no pathname is specified and the process is not currently using an invocation of DMS, an error message is returned, and no other action is performed.

-reason STR
: is the reason why the DMS is being shut down. The stated reason appears in the messages notifying users at user warning and user shutdown times. If this control argument is not specified, a prompt is issued for the reason. STR can be a maximum of 64 characters and must be quoted if it contains whitespace.

-user_warning_time TIME, -uwtm TIME
: specifies user warning time. TIME may be absolute or relative to the time at which the dm_system_shutdown command is invoked, and must be in a form acceptable to the convert_date_to_binary_ subroutine. If no user warning time is specified, it is calculated back from begin shutdown time, using the specified or default begin shutdown delay.

-begin_shutdown_delay DELAY, -bsd DELAY
: specifies the time period between user warning and begin shutdown. DELAY must be in a form acceptable to the convert_date_to_binary_ subroutine. If begin shutdown time is not specified (see below), it is set by calculating forward from user warning time, using DELAY. If DELAY is not specified, it defaults to the begin-shutdown-delay time in the configuration file for the DMS being shutdown.

-begin_shutdown_time TIME, -bstm TIME
specifies begin shutdown time. TIME may be absolute or relative to the time at which the dm_system_shutdown command is invoked, and must be acceptable to the convert_date_to_binary_ subroutine. If no begin shutdown time is specified, it is calculated forward from user warning time, using the specified or default begin shutdown delay.

-user_shutdown_delay DELAY, -usd DELAY
specifies the time period between begin shutdown and user shutdown. DELAY must be in a form acceptable to the convert_date_to_binary_ subroutine. If user shutdown time is not specified (see below), it is set by calculating forward from begin shutdown time using DELAY. If DELAY is not specified, it defaults to the user-shutdown-delay time in the configuration file for the DMS being shutdown.

-user_shutdown_time TIME, -ustm TIME
specifies user shutdown time. TIME may be absolute or relative to the time at which the dm_system_shutdown command is invoked, and must be acceptable to the convert_date_to_binary_ subroutine. If no user shutdown time is specified, it is calculated forward from begin shutdown time, using the specified or default user shutdown delay.

-user_bump_delay DELAY, -ubd DELAY
specifies the time period between user shutdown and user bump. DELAY must be in a form acceptable to the convert_date_to_binary_ subroutine. If user bump time is not specified (see below), it is set by calculating forward from user shutdown time using DELAY. If DELAY is not specified, it defaults to the user-bump-delay time in the configuration file for the DMS being shutdown.

-user_bump_time TIME, -ubtm TIME
specifies user bump time. TIME may be absolute or relative to the time at which the dm_system_shutdown command is invoked, and must be acceptable to the convert_date_to_binary_ subroutine. If no times are specified (i.e., no control argument specifying TIME is used), user bump time defaults to the scheduled Multics shutdown time. If some other time is specified, but not user bump time, the specified or default user bump delay is used to calculate forward to user bump time.

-daemon_logout_delay DELAY, -dld DELAY
specifies the time period between user bump and daemon logout. DELAY must be in a form acceptable to the convert_date_to_binary_ subroutine. If daemon logout time is not specified (see below), it is set by calculating forward from user bump time using DELAY. If DELAY is not specified, it defaults to the daemon-logout-delay time in the configuration file for the DMS being shutdown.

-daemon_logout_time TIME, -dltm TIME
specifies daemon logout time. TIME may be absolute or relative to the time at which the dm_system_shutdown command is invoked, and must be acceptable to the convert_date_to_binary_ subroutine. If daemon logout time is not specified, it is set by calculating forward from user bump time, using the specified or default daemon logout delay.

ACCESS REQUIRED

You must have re access to dm_admin_gate_ to execute this command.

EXAMPLES

Suppose a Multics shutdown is scheduled for 8:00, and all default delays are five minutes, the command

```
dm_system_shutdown -bsd 20min
```

would result in a shutdown of data management in five stages, as follows:

```
user warning:    7:30 (specified delay)
begin shutdown:  7:50 (default delay)
user shutdown :  7:55 (default delay)
user bump:       8:00 (default time)
daemon logout:   8:05 (default delay)
```

while the command

```
dm_system_shutdown -uwtm 5:00 -ustm 6:10 -ubtm 6:00
```

would result in a shutdown of data management in five stages, as follows:

```
user warning:   5:00 (specified time)
begin shutdown: 5:05 (default delay)
user bump:      6:00 (specified time)
daemon logout:  6:05 (default delay)
```

Since user shutdown is scheduled after user bump, it is not executed.

Name: dump_firmware

SYNTAX AS A COMMAND

```
dump_firmware path mem {addr count}
```

FUNCTION

dumps the contents of a segment containing MPC firmware.

ARGUMENTS

path
    is the pathname of the segment containing the firmware.

mem
: must be "cs" to dump the control store overlay, "rw" to dump the read/write overlay, or "size" to print the locations and lengths of overlays in the module. If it is "size," you need give no further arguments; otherwise, give addr and count.

addr
: is the starting address to dump, in hexadecimal.

count
: is the number of words to dump, in hexadecimal.

NOTES

If the firmware segment to be dumped is for an MSP800 device adapter unit (DAU), you must still specify cs or rw, which does not affect the output because the DAU has only one memory and no control store. A DAU word contains one eight-bit byte, as opposed to the MPC's two eight-bit byte format.

Name: dump_mpc

SYNTAX AS A COMMAND

```
dump_mpc mpc_name {-control_args}
```

FUNCTION

performs a dump of the read/write memory of an MPC and selectively edits the dump, the trace table, and MPC and device statistics.

ARGUMENTS

mpc_name
: is the name of the MPC to be dumped. This name must appear on an mpc card in the config deck. If you omit mpc_name, give -channel.

CONTROL ARGUMENTS

-dump
: displays a hexadecimal dump.

-extend, -ext
: extends the output file if it exists. (Default: to overwrite the file)

–mpc
displays MPC error data only.

–stat
displays the MPC and device statistics.

–trace
displays an interpreted trace of the MPC.

This page intentionally left blank.

-channel channel_name, -chn channel_name
: specifies a channel name, where channel_name is of the form [iomtag] [channel_no] (i.e., a14). The iomtag field must be a tag of a configured IOM and the channel_no must be a decimal channel number. If this control argument is used, the mpc_name argument is optional. If both are used, the channel must be connected to the MPC specified.

-output_file {path}, -of {path}
: directs dump output to the segment specified by path. If path is not given, a default segment name of [mpc_name].list is used. If this control argument is not given, the default is to direct output to your terminal.

-long
: formats output for devices with 132 columns or more. The default is based on output type and can be used to override the file output default.

-short
: formats output for devices with fewer than 132 colums. The default is based on output file type and can be used to override the file output default.

ACCESS REQUIRED

You must have re access to rcp_priv_ to use the dump_mpc command.

NOTES

If neither the -stat, -dump, -mpc, nor -trace control arguments are specified, only the MPC and device statistics are displayed.

Switch 4 on the MPC maintenance panel is used to control tracing in the MPC. Tracing is only done if this switch is in the down position. If the trace table is being dumped to see the events leading up to a particular error condition, it may be useful to place switch 4 in the up position as soon as possible after the error occurs. This inhibits further tracing of I/O in the MPC and reduces the chances of losing trace data caused by the table wrapping around before the dump can be taken.

Name: dump_partition

SYNTAX AS A COMMAND

```
dump_partition pvname partname offset {length} {-control_args}
```

SYNTAX AS AN ACTIVE FUNCTION

```
[dump_partition pvname partname offset {length}]
```

FUNCTION

displays data from a named disk partition. By default this data appears in octal, four words per line, although other output formats can also be selected. See also the clear_partition and list_partitions commands. As an active function, dump_partition returns the contents of the specified words in octal, separated by spaces.

ARGUMENTS

pvname
: is the name of the physical volume on which the partition to be dumped exists.

partname
: is the name of the partition to be dumped. It must be four characters or less in length.

offset
: is the octal offset at which to begin dumping.

length
: is the number of words to be dumped. If not supplied, one word is dumped.

CONTROL ARGUMENTS

-bcd
: produces data including the BCD character representation.

-long, -lg
: produces data in long form, similar to dump_segment -long.

-no_header, -nhe
: suppresses the header.

-short, -sh
: produces data in short form, similar to dump_segment -short.

ACCESS REQUIRED

You need access to the phcs_and hphcs_ gates.

Name: ed_installation_parms

SYNTAX AS A COMMAND

```
ed_installation_parms {path}
```

FUNCTION

edits installation_parms and rate_structure segments. Normally, the segments edited should be those in the directory >system_control_1.

ARGUMENTS

path
: is the pathname of an installation_parms segment. If path is not given, a segment named installation_parms, in the working directory, is edited. The segment is created if it does not already exist. If the installation_parms segment defines multiple rate_structures, then segments named rate_structure_N (where N is a digit in the range 1 through 9) in the same directory as installation_parms are edited. Editor requests allow the user to switch between installation_parms and the rate_structure segments within one invocation of the editor. The entire set of segments is edited as a single unit in order to maintain consistency among them. All editing is done with temporary copies of these segments; the actual segments are updated only by the "w" request described below.

NOTES

After this command is invoked, the editor enters its main request loop, and prompts the user with the message "type". To return to this request loop from any editor query, press the QUIT key and invoke the "program_interrupt" command. Certain operations of the editor are defined as critical, as they may require updating many interdependent fields. If the user attempts to return to the request loop using QUIT and program_interrupt when such an operation is taking place, the editor asks the user to confirm the abandonment of the operation. The user may elect to restart the interrupted operation. After an abandonment, it may be necessary to retype a number of tables, or quit the editor and redo the editing.

If the user is editing multiple rate_structures, then the following notes apply. When editing any segment except installation_parms, the user is said to be editing a rate_structure, and is limited in the operations which may be performed. In general, it is possible to change the pricing information of the rate_structure, but not values (such as device names) that affect all rate_structures. In addition, there are items specific to installation_parms that cannot be accessed while editing a rate_structure. See "Notes on Rate Structure" for details.

Some of the requests accept keywords. If the keyword is not given with the request, the system prompts with the message "id"; the system administrator then responds with one of the keywords described below. At this point, the system either prints out the current value of the item or prompts for a new value. (See "Example" at the end of this command description.) All requests, keywords, and values can be typed ahead, avoiding the prompting messages. In several cases, certain typed ahead values alter the operation of the request, causing it to operate on a subset of the data that it normally operates on. See the descriptions below for details.

LIST OF REQUESTS

The edit requests for the ed_installation_parms command are listed below.

add XX, a XX
: adds new items to the entry specified by the XX keyword. This request may only be used for device_prices, config_table, rate structure_names, and resources entries. (See "Keywords" below.)

change XX, c XX
: allows an entry, or a portion of an entry, specified by the XX keyword to have its value changed. (See "Keywords" below.)

copy
: prompts for an existing rate structure name and copies pricing information from that rate_structure to the one being edited.

default
: sets all entries in the installation_parms segment to default values. Requests of the form <release name>_default (e.g., mr10.0_default) are also provided, with any release in which new parameters have been added to installation_parms, to initialize just the new parameters. See the installation instructions for the release, for more information. When specified while editing a rate_structure, the default request causes pricing information from the rate_structure named "default" (the one in installation_parms) to be copied to the rate_structure being edited.

help {long}, h {long}
: prints a summary of available requests and keywords. Normally, only the primary name of each keyword is printed. If the long argument is given, then the keyword abbreviations are also printed.

print XX, p XX
: prints the entry specified by the XX keyword. (See "Keywords" below.)

retype XX, r XX
: allows all portions of an entry specified by the XX keyword to be replaced. (See "Keywords" below.)

w
: writes the changes that have been made back into the input segments; all editing is done on temporary copies.

q
: quits from editor. Must be preceded by a "w" request to save editing.

rate_structure RS, rates RS
: specifies that subsequent editor requests are to affect the pricing information of the rate_structure RS. The installation_parms segment is referred to as rate_structure "default". The site may change this or any rate structure name. The single character RS "." always refers to installation_parms, even if the name has been changed from "default". To return to editing installation_parms, type the request: "rate_structure .". If RS is omitted, the name of the current rate structure is printed.

verify, v
: performs certain verifications of the integrity of the installation_parms and rate_structure_N segments. Most checks are designed to detect inconsistencies between installation_parms and rate_structure segments. If -force, -fc, force, or fc is typed on the same line as the verify request, ed_installation_parms forces table consistency, zeroing prices for affected entries. The user is notified of each such alteration.

LIST OF KEYWORDS

The various keywords and abbreviations that can be used in requests are summarized below.

abs_cpu_default_limit, abscpudf
: default absentee CPU time limit (in seconds), for each queue. These default limits are used when no time limit is specified by the user who enters a job in one of the background absentee queues. This parameter is included among those selected by the abs_queue_parameters keyword, described below.

abs_cpu_max_limit, abscpumx
: maximum absentee CPU time limit (in seconds), for each queue and shift. These limits allow jobs with high CPU time limits (either specified by the user or determined by the abs_cpu_default_limit parameter) to be run during lightly-loaded shifts, by specifying the longest-running jobs, from each queue, that will be permitted to log in during each shift. This parameter is included among those selected by the abs_queue_parameters keyword, described below.

abs_default_queue, absdfq
: default absentee queue for submission of jobs by the enter_abs_request, pl1_abs, etc. commands. This parameter may have a value in the range 1 to 4 and defaults to 3 unless changed by the system administrator.

abs_timax
: absentee timax per queue (in microseconds). This parameter is among those selected by the abs_queue_parameters keyword, described below.

abs_queue_parameters, abs_
absentee timax, per queue (in microseconds), default absentee CPU time limit, and maximum absentee CPU time limit, the latter for each shift, and the latter two for each queue and in seconds. These parameters can also be selected individually, by keywords described above.

access_ceiling, acce
maximum sensitivity level that may be used, and all categories that may be used. Categories are specified by an octal string, levels by a decimal digit.

all
every entry. This keyword may only be used with the print or retype requests.

all_structures
every entry of installation_parms and every rate_structure. This keyword has the
* effect of "all" for installation_parms, and "rates" for all rate_structures.

authentication, auth
default level of security for volume authentication under RCP Resource Management. Acceptable values for this keyword are:

none
do not authenticate volume labels on mount requests.

nominal
authenticate volume labels via an algorithm that rejects volumes only if they are obviously incorrect. Additionally, allow manual authentications of "***" (as an alternative to typing the correct authentication code) to authenticate a volume whose identity is in doubt.

automatic
authenticate volume labels via an algorithm that accepts volumes only if they are obviously correct. Does not allow manual authentications of "***" to authenticate a volume whose identity is in doubt (requires the operator to type the correct authentication code).

Automatic authentication means that the system checks IBM and ANSI magnetic tape volumes for an authentication code in the tape label that matches the tape label name. Such volumes created on Multics meet this requirement. IBM and ANSI tape volumes created on other systems require manual authentication when automatic authentication is in effect.

manual
require operator to manually authenticate every mount request with the proper authentication code. Does not allow an authentication code of "***" to be used under any circumstances.

auto_registration, auto

automatic volume registration by RCP. The string "on" sets the flag on; "off" is the default. Automatic registration of an unregistered volume is performed when the operator allows a user to mount the volume. The volume is registered, with default attributes, to the user requesting the volume. This keyword may only be used with the change, print, or retype requests.

category_names, cate

short (up to eight characters) and long (up to 32 characters) names for up to 18 access categories. Embedded blanks are allowed in long names only. Category names may not be blank.

config_table, conf

elements of the configuration table.

It is required that elements be in order by number of CPUs, within that by amount of main memory, within that by amount of bulk store, and within that by shift; smaller numbers first.

The config table is automatically sorted after an add, change, or retype operation. Duplicate elements and elements containing -1 in the CPU field are deleted by this sort. Thus, config elements can be deliberately deleted by changing the CPU field to -1.

A configuration table element consists of 9 numbers, in order, as follows:

```
cpu mem bulk shift min max base absu absq
```

The first four parameters describe the configuration and shift, and the last 5 give the load control parameters to be used for that configuration and shift. All are decimal numbers.

The mem and bulk parameters are the number of pages of main memory and bulk store, respectively. The other two are the number of CPUs and the shift number.

The load control parameters min and max are only used when response control (automatic load leveling) is enabled (by the operator command maxu level). Use of this feature is not recommended, since the load leveling algorithm is unsatisfactory. Nevertheless, values for these parameters must be given when entering a config table element.

The base parameter is the maximum number of load units allowed. The min, max, and base parameters may optionally contain a decimal point and a single digit fraction (tenths).

The absu and absq parameters are the maximum number of background absentee users and the number of the lowest priority (highest numbered) absentee queues from which jobs are to be taken. (Note that the absu parameter is ignored if the number of background absentee users is made a function of interactive load. (See "Notes on Absentee Load Control Parameters.") Nevertheless, this parameter must always be given when entering a config table element.

The print request prints the entire config table.

The retype request completely replaces the entire config table; all of its elements must be entered.

The add request adds a new element to the table. The automatic sort described above properly positions the new element.

The change request changes one element. The first four parameters (cpu, mem, bulk, and shift) are prompted for (or they may be typed ahead). The element matching all four of those parameters (exactly) is the one changed. All nine parameters of the element must be entered. Thus, the first four parameters must be typed twice: once to locate the element to be changed, and once to give new (or possibly the same) values to those parameters.

When entering an element, for the retype, add, or change request, the nine parameters must be typed, as decimal numbers, in the order shown above. Each parameter is prompted for; the prompting may be avoided by typing all nine parameters of the element in response to the prompt for the first one (CPUs).

cwe_count, cwec
count parameter of the terminal channel wakeup loop detector. If a user causes more than cwe_count interactions within cwe_time while logging in, the channel is hung up. An interaction occurs every time the Answering Service is notified of an event for the terminal channel. The number of interactions perceived by the Answering Service is typically the same as the number of input lines (terminated by a newline character) entered by the user. The default value for this parameter is 10.

This loop detector protects the Answering Service from overload due to accidental or malicious generation of a large number of wakeups in rapid succession by devices such as intelligent terminals. If this is a problem at the site, the system administrator should establish the smallest possible ratio of cwe_count/cwe_time that still allows high-speed channel users to log in.

The administrator should estimate the maximum number of wakeups which could reasonably occur in a burst. For example, a hardwired terminal might generate five wakeups in quick succession: hangup, dialup, answerback, login line, and password line. It would therefore be imprudent to set cwe_count to a value less than 5 on a system having hardwired terminals and answerback reading enabled. The value of cwe_time would be set according to the fastest channel configured at the site.

cwe_time, cwet
: time parameter of the channel wakeup error loop detector in seconds. Default time is 3 seconds. See also the description of cwe_count above.

default_pdir_seg_quota, df_pdsq
: process directory segment quota. The default is the value used at system startup.

default_pdir_dir_quota, df_pddq
: process directory directory quota. the default value is 1000.

device_prices, devi
: names and prices (dollars per hour, for each shift) for each of up to 16 miscellaneous devices (teletype channels, high-speed channels, tapes, etc.). For the print request, if the name of a device is typed ahead on the input line after the device_prices keyword, the prices for that device are printed; otherwise the prices of all devices are printed. For the retype request, the name of each device is typed and new prices are requested for all shifts.

device_names, devname, devn
: names of the miscellaneous devices. Allows changing the name of an existing device or printing the names of all the defined devices. Changing a device name to devN, where N is a number from 1 to 16 indicating the location of the device in the table, has the effect of deleting the device, since unused device table locations contain names of that form. If a deleted device is the last one in the list, the list is shortened by one.

  Charging for certain devices is built into the system. These devices must be defined in the device table, with their names spelled exactly as the system expects to find them. These devices are defined automatically by the default request. A warning message is printed by the print device_prices and print devnames requests for any of these devices that is not defined.

devices, device
: names and prices (dollars per hour, for each shift) for each of up to 16 miscellaneous devices. For the print request, if the name of a device is typed ahead on the input line after the devices keyword, the prices for that device are printed; otherwise the names and prices of all devices are printed. For the retype request, allows the user to enter new prices for the system defined devices, and enter a new set of site-defined device names and prices. For the add request, allows the user to define a new device.

foreground_cpu_default_limit, fgcpudf
: default CPU time limit (in seconds) for the foreground absentee queue. This limit is used when no time limit is specified by the user who enters a job in the foreground absentee queue.

fpe_count, fpecount, fpec
: count parameter of the fatal process error loop detector. Default is 3. If interactive users get fpe_count fatal process errors within fpe_time, they are considered to be in fatal process error loops and are logged out.

fpe_time, fpetime, fpet
time parameter of the fatal process error loop detector. Default is 1 minute.

idle_time_constant, idle
time over which moving average of foreground load is taken, for use in adjusting maximum background absentee users (abs_maxu). Default is 15 minutes.

inactive_time, inac
number of real-time seconds a process may remain blocked before being bumped for inactivity.

installation_id, inst
installation name, city, and state (maximum of 32 characters). Example:

```
Company, City, State
```

level_names, leve
short (up to eight characters) and long (up to 32 characters) names for up to eight sensitivity levels. Embedded blanks are allowed in long names only; "." means leave name blank (level 0 only).

log_parameters, log_
number of pages (1 to 63) to which ring 0 syserr log may grow before being copied. A value of -1 means disable automatic copying; 0 means use default value.

login_time, logi
number of real-time seconds in which login must be completed.

max_abs
per-shift upper limit in the abs_maxu equation. See "Notes on Absentee Load Control Paramaters" and "Notes on Shift and Queue Specification" below.

max_qres
per-shift and per-queue upper limit in the qres equation. See "Notes on Absentee Load Control Parameters" and "Notes on Shift and Queue Specification" below.

min_abs
per-shift lower limit in the abs_maxu equation. See "Notes on Absentee Load Control Parameters" and Notes on Shift and Queue Specification" below.

min_qres
per-shift and per-queue lower limit in the qres equation. See "Notes on Absentee Load Control Parameters" and "Notes on Shift and Queue Specification" below.

new_proc_change_auth
controls the use of the -auth control argument to the new_proc command. If this parameter is set on, users can use the -auth control argument. If this parameter is set off users cannot use the -auth control argument.

The -auth control argument to the new_proc command permits users to create a new process at a different access authorization. This feature can be used by subverting "trojan horse" programs to gain access to information at inappropriate levels. The feature should be disabled by security conscious sites. Users can change authorization by disconnecting and then logging in using the -auth control argument to the login command.

operator_inactivity_limit
time limit (in number of seconds) imposed on operator inactivity. Operators who enter no input for more than the specified number of seconds are automatically signed out. The default value is zero (specifying no time limit).

password_change_interval, pwci
specifies the maximum time, in days, that a password can be used before it must be changed. If a password is used after the expiration date, an error message is printed. The user must use the -change_password option to the login command to change the password. If this parameter is set to zero, passwords will not automatically require change. The default value is zero.

password_expiration_interval, pwei
specifies the maximum time, in days, that a password can remain unused. If used after the specified period of non-use, the login is denied and the user must seek administrative aid to revalidate the password. If this parameter is set to zero, passwords do not automatically expire after a period of non-use. The default is zero. The administrator can use the new_user command to revalidate the password.

password_gpw_length, pwgl
specifies the length of passwords generated by the -generate_password login option. It must be between 5 and 8. The default is 6.

password_min_length, pwml
specifies the minimum length (in characters) for a user-supplied password. It must be between 3 and 8, or 0. If it is zero, password length is not checked. This parameter applies only to new passwords: existing passwords are not checked. The default is zero (no check).

pct_abs
percent of idle units to be made available to background absentee jobs. Used in the abs_maxu equation. See "Notes on Absentee Load Control Parameters" and "Notes on Shift and Queue Specification" below.

pct_qres
percent of absentee slots to be reserved for each queue. Used in the qres equation. See "Notes on Absentee Load Control Parameters" and "Notes on Shift and Queue Specification" below.

prices
: prices of disk storage (per page-second or page-30-days), per-month registration, and per-shift prices of CPU time (dollars/virtual CPU hour), connect time (real hours), terminal I/O operations (kilo lines), and memory usage (kilo memory units).

  The change request allows the user to specify a shift and set the CPU, connect time, terminal I/O, and memory prices. The retype request asks for the disk and registration prices, and then optionally asks for all other prices for each shift.

queue_prices
: prices for absentee virtual CPU time (dollars/virtual CPU hour), absentee memory usage (dollars/kilo memory unit), and I/O daemon usage (dollars/kilo lines), for up to four queues.

rate_structure_names, rate_structure_name, rsn
: allows changing the name of a rate_structure, printing the names of the defined rate structures, or adding a rate_structure.

rates
: all pricing fields: device_prices, prices, queue_prices, resource_prices.

require_operator_login
: controls the operator identification and authentication option. The value "yes" requires operators to use the sign_on command before entering commands; the value "no" specifies that there will be no such requirement. The default value is "no."

resources, resource
: names and prices of resources for which users are to be charged. For the print request, if the name of a resource is typed ahead on the input line after the resources keyword, the price for that resource is printed; otherwise the names and prices of all resources are printed. For the retype request, allows the user to re-enter all resource names and prices. For the add request, allows the user to define a new resource.

resource_prices, reso
: names and prices of resources for which users are to be charged. The resource price list is currently used only for charging for special forms in I/O daemon requests. For the print request, if the name of a resource price is typed directly after the resource_price keyword, that price is printed; otherwise all the prices are printed. For the retype request, the name of each resource is typed and a new price is requested.

resource_names, rscname, rscn
: name of an existing entry in the resource price list. Allows changing the name of an existing resource.

resource_wait_time, rsctime, rsct
: time interval at which to keep checking for resource availability when an absentee job is waiting for a resource and there are no other jobs logging in or out. Default is 5 minutes.

revalidate
: the users password is revalidated (after having expired following a period of non-use).

rsc_mgmt_enabled, rsc_
: activation of the RCP Resource Management subsystem. The string "on" sets the flag on. Initially this flag is off; it should not be turned off again once it has been turned on. This keyword may only be used with the change, print, or retype requests.

shift_table, shif
: shifts for each half-hour of the week. Shifts are numbered 0 to 7. By default, the change request changes the shift of a single half hour, specified by day, hour, and half hour (see example below). However, if the word "thru" is typed ahead on the input line after the first half hour, then a second day, hour and half hour are read, and all half hours from the first through the second are set to the specified shift (see the example below).

sus_cpu_time, susc
: CPU time limit imposed on a suspended process. The default is 5 seconds. The default handler for the sus_ signal, in a user process, goes blocked and waits to be released. If the process fails to respond in this way, and continues running, it is destroyed after it has used the specified amount of CPU time (after the sus_ signal is sent).

sus_real_time, susr
: real time limit imposed on a suspended process. The default is 180 seconds. Before going blocked, a suspended process sends a wakeup to the answering service, specifying the event channel on which it is about to go blocked, and over which it can be released. If the process fails to respond in this way within the specified interval of real time, it is destroyed.

titles
: company name and department name single spaced (maximum of 64 characters) and double spaced (maximum of 120 characters). Example:

```
Company Name, Inc.

C o m p a n y   N a m e ,   I n c .
```

tries
: number of login tries allowed before Multics hangs up line.

trm_cpu_time, trmcputime, trmc
cpu time limit imposed on a process that is being terminated and has been sent a trm_ signal. Default is 5 seconds.

A process being terminated involuntarily by the answering service (e.g., by the bump command), as opposed to a process that terminates itself voluntarily (e.g., by the logout command), is sent a trm_ signal, and is given a small amount of time to terminate itself. The default handler for trm_ signals the finish condition in the user's process.

trm_real_time, trmrealtime, trmr
realtime limit imposed on a process that has been sent a trm_ signal. Default is 120 seconds.

unload_on_detach, unlo
detached volumes are unloaded automatically. The string "on" sets the flag on; "off" is the default. When the flag is off a detached volume remains on its drive until the drive is needed for another volume. This flag has no effect when the user specifies "-retain all" in the attach description of the volume being detached (see the tape_ibm_ module in the *Multics Subroutines and I/O Modules* manual, Order No. AG93). This keyword may only be used with the change, print, or retype requests.

update_time, upda
number of real-time seconds between accounting updates.

warning_time, warn
number of real-time seconds between warning (of an automatic logout) and actual logout.

This value also affects the automatic bumping of users prior to a shutdown scheduled through the initializer down command (described in Section 4). The bump is scheduled for N minutes before the time specified in the down command (where N is warning_time divided by 60, rounded up to the next whole minute). At that time, all interactive users are bumped (i.e., warned that they will be logged out in N minutes), and the login word is changed to prevent further logins.

validate_daemon_commands
controls activation of the secure daemon facility. The string "on" sets the facility on. the string "off" sets the facility off. Initially the facility is set "off." See "Notes On Secure Daemon Facility," later in this command description.

NOTES ON RATE STRUCTURE

The following keywords may be used when editing a rate_structure: all (same as "rates"), device_prices, device_names, devices, prices, queue_prices, rates, rate_structure_names, resource_prices, resource_names, and resources. Further restrictions on this set of keywords exist within some requests. For example, none of the keywords are allowed to change device or resource names when editing a rate_structure because such a change must be reflected in all rate_structures.

Users must have read access to rate_structure_N segments in >sc1 when multiple rate_structures are in effect. In addition, all daemons that charge users for services must have read access to the SAT, to determine the appropriate rate_structure for a user's project. These access settings are unnecessary when a site is using a single rate structure.

NOTES ON ABSENTEE LOAD CONTROL PARAMETERS

The installation_parms segment contains a number of parameters that enable the site to regulate the background absentee load. (Background absentee jobs are those from the four background absentee queues, numbered 1 to 4, as opposed to jobs from the foreground absentee queue. The latter are regulated by foreground load control.

By default, abs_maxu, the number of absentee slots (i.e., the maximum number of concurrently logged in background absentee jobs) is a constant, for each shift-configuration combination in the configuration table (see below). However, it can optionally be made a function of interactive load. The abs_maxu figure in the configuration table is only used when the parameters described immediately below (pct_abs, min_abs, and max_abs) have not been set.

Background absentee jobs are intended to occupy idle capacity. The number of idle units is computed by the equation:

```
idle_units = max_units - n_daemons - n_interactive
```

where max_units is the system maximum units figure for the current shift and configuration, as specified in the configuration table, and n_daemons and n_interactive are the current number of units occupied by daemon and interactive users, respectively.

Since this figure can fluctuate rapidly as the number of interactive users changes, it is smoothed in the following way: a moving average of idle units over the last M minutes is kept (where M is the value, in minutes, of the idle_time_constant parameter in the installation_parms segment), and the lower of the current value and the average value of idle units is used to compute abs_maxu each time an absentee login decision is made. This algorithm causes the increase in abs_maxu to lag behind a decrease in interactive load, but the decrease in abs_maxu occurs immediately in response to an increase in interactive load.

The abs_maxu figure is computed by the equation:

```
   abs_maxu = min (max_abs, max (min_abs, pct_abs * idle_units))
```

that is, abs_maxu varies between min_abs and max_abs, as a function of pct_abs and idle_units. For example, with pct_abs = 10%, max_abs = 6, and min_abs = 1, abs_maxu varies between 1 and 6 as idle_units varies between 10 and 60. If idle_units goes above 60 or below 10, abs_maxu does not rise above 6 or fall below 1.

The three parameters max_abs, min_abs, and pct_abs are kept in the installation_parms segment, and can have different values on each shift.

It is possible to reserve a portion of abs_maxu for the use of the higher priority queues. This avoids delaying jobs from the higher priority queues that are entered into the queues after all the slots have been occupied by jobs from lower priority queues. Two algorithms are described here: the way the qres figures for each queue are computed, and the way all the qres figures are used to determine whether or not an absentee job from an individual queue is permitted to log in.

The number of reserved slots for each queue is computed by the equation:

```
   qres(q) = min (max_qres(q), max (min_qres(q), pct_qres(q) * abs_maxu))
```

where qres(q) is the number of slots reserved for queue q. Thus qres(q) varies between min_qres(q) and max_qres(q), as a function of pct_qres(q) and abs_maxu. The min_qres, max_qres, and pct_qres parameters are kept in the installation_parms segment and have separate values for each queue and each shift.

The reserved slot facility does not place an absolute limit on the number of jobs from any (except the lowest priority) queue. It prevents jobs from lower priority queues from logging in if their logging in would leave too few slots available for the higher priority queues, but it allows jobs from high priority queues to occupy all of the slots and exclude jobs from lower priority queues. The algorithm is as follows: to determine if a job from queue q can log in, first reduce the current abs_maxu by the number of jobs currently logged in. This gives the number of available slots. (If it is zero, no job can log in.) Then, for each queue of higher priority than queue q, if the number of jobs currently logged in from that queue is less than qres(i) (where i is the number of that higher priority queue), reduce the number of available slots by the difference. If there is at least one available slot after that has been done, the job can log in.

NOTES ON SHIFT AND QUEUE SPECIFICATION

The max_abs, min_abs and pct_abs keywords of ed_installation_parms define values for each shift. The max_qres, min_qres and pct_qres keywords define value for each queue, in each shift. For these keywords, the print request prints values for all shifts and queues, but the change and retype requests can select particular shift and queue groups to be changed or respecified.

For max_abs, min_abs and pct_abs, you can specify which shift to change on a change request line, either by using the shift keyword followed by a shift number or range of numbers, or by giving a single shift number. The new value can also be given on the request line. For the retype request, you can use the shift keyword to specify which shift(s) are to be changed, followed by the new values. If new values aren't given, they are prompted for. For example:

```
type ! c max_abs shift 1 4
type ! c max_abs shift 2-4 6
type ! c max_abs 5
max_abs: shift 5: ! 7
type ! r max_abs shift 2-3
max_abs: shifts 2-3: ! 5
type ! p max_abs
max_abs - PER SHIFT
   0     1     2     3     4     5     6     7
  -1     4     5     5     6     7    -1    -1
```

For max_qres, min_qres and pct_qres, you can specify which queue and shift to change in a change request line, either by using the queue and shift keywords followed by a number or range of numbers, or by giving a single queue or shift number. The new value can also be given on the request line. If neither queue nor shift keyword is used, then the queue number must be given first, followed by the shift number and the new value. If queue or shift number or new value are omitted, you are prompted for these values. For the retype request, you may use queue or shift keywords to limit retyping to a specified set of queues or shifts. By default, retype prompts for value for all queues of each shift. In a retype request line, numeric values not preceded by a queue or shift keyword are treated as new values. For example:

```
type ! c max_qres queue 1-4 shift 0-7 0
type ! c max_qres queue 1-3 shift 1 3
type ! c max_qres 4 1 2
type ! r max_qres shift 2-3
max_qres: shifts 2-3: queue 1: ! 4
max_qres: shifts 2-3: queue 2: ! 3
max_qres: shifts 2-3: queue 3: ! 2
max_qres: shifts 2-3: queue 4: ! 1
type ! c max_qres shift 4-5
queue: 2
max_qres: queue 2: shifts 4-5: ! 3
type ! p max_qres
max_qres - PER QUEUE AND SHIFT
Q     0     1     2     3     4     5     6     7
1     0     3     4     4     0     0     0     0
2     0     3     3     3     3     3     0     0
3     0     3     2     2     0     0     0     0
4     0     2     1     1     0     0     0     0
```

NOTES ON SECURE DAEMON FACILITY

The secure daemon facility enables administrators to control operator access to daemon processes. The administrator must set appropriate access rights on the ACS segment associated with the specified daemon. (This procedure is described in the *Multics Programmer's Reference Manual*, Order No. AG91.) When the secure daemon facility is set "on," the following actions occur:

1. If the operator uses the login command to login a daemon, the system will check to determine if an access control segment exists for the specified daemon. If no ACS segment exists, the command will not be executed. If the ACS segment exists, the system will check to determine if the operator has sufficient access to login the daemon. Additionally, the system will check to determine if the selected daemon is permitted to login. If the operator or daemon do not have the required access, the daemon is not logged in.

2. If the operator uses the reply, quit, or logout command to communicate with a daemon, the system will check the access control list to determine if the operator has sufficient access to perform the specified operation. If the operator has not been granted sufficient access, the command will not be executed.

See the *Multics System Administration Procedures* manual, Order No. AK50 for additional information on the secure daemon facility.

EXAMPLES

In this sample, the system administrator first prints the current value of the "warning time" variable (the number of seconds before shutdown that a user is warned), and then changes the value. Finally, the change is made to the original installation_parms segment (w), and the administrator exits (q). (Lines, or portions of lines, typed by the administrator are preceded by " ").

```
!  ed_installation_parms
   type  !p
   id    !warning_time
   warning_time:  300 sec.
   type  !c
   id    !warning_time
   warning time (secs)  !  6000
   type  !w
   type  !q
   <ready>
```

```
!  ed_installation_parms
   type  ! c shif Fri 18 00 thru Mon 07 30 5
   type  ! w
   type  ! q
   <ready>
```

This defines shift 5 to start at 1800 Friday and end at on 0800 Monday.

Name: ed_mgt

SYNTAX AS A COMMAND

```
ed_mgt {path}
```

FUNCTION

allows the system administrator to edit a copy of the master group table (MGT). The MGT defines work classes and load control groups, which determine the number of users from each group that are permitted to log in, and the percentage of system resources that they are allocated while logged in. A complete discussion of load control groups and work classes is found in the *Multics System Administration Procedures* manual, Order No. AK50.

ARGUMENTS

path
: is the pathname of the copy of the MGT to be edited. If the pathname is not given, the default is the segment MGT.mgt, in the working directory. If the segment does not exist, it is created and initialized. (The suffix, mgt, is not assumed, or required, by ed_mgt -- the MGT to be edited may have any name. However, the suffix is required by the install command.) Normal practice is to edit the MGT.mgt segment in >udd>SysAdmin>admin and then install it, using the install command. The install command merges it with the system copy in the >sc1 directory, preserving the current load figures in the system copy. The system copy of the MGT should never be edited directly. (The load_ctl_status command can be used to display the contents of the system copy.)

NOTES

This command operates in response to requests from the user. Most of the requests have a full name and one or more abbreviated names. Either the full names or the abbreviations may be typed. Many of the requests take one or more arguments. If these arguments are not supplied on the same line as the request, ed_mgt prompts the user for them. Multiple complete requests (including their arguments) may be typed on the same line. (The line may end with an incomplete request, i.e., with some arguments missing.) The only limit on the number of requests that may be typed on one line is the length of the buffer used to read request lines (128 characters).

One of the requests, change, puts ed_mgt into "change mode"; once in this mode only change requests are recognized (other ed_mgt requests are not recognized). To return to ed_mgt request level from change mode, it is necessary to type "." or "*". The prompting message for the next request, at request level, is "type"; in change mode, it is "code".

The user may issue a quit signal, and then type the program_interrupt (pi) command, to return to ed_mgt request level. This is useful to abort the output of certain requests that may produce a large amount of output. Unless it is issued while a request that modifies the MGT is actually being executed, the quit-pi sequence has no adverse effects on the contents of the MGT.

The MGT consists of a header and an array of entries. The first 16 are always work classes 1 through 16; the rest are site-defined load control groups. Some requests operate on only one of these two entry types; others operate on both types. There is a conceptual pointer that always points to one of the entries. There are several requests that move this pointer. Some requests operate only on the entry specified by the pointer; others operate on all entries of one of the types, or on all entries in the MGT.

Requests that operate on more than one entry do not move the pointer. Requests that do move the pointer print the type and name of the entry to which it is moved (unless there are further requests on the line after the one that moved the pointer). When ed_mgt is entered, the pointer is set to the first load control group (the 17th entry). If there are no groups (which is the case if the MGT did not exist), then ed_mgt prompts the user for the name and parameters of a group to add.

LIST OF REQUESTS

The ed_mgt requests, and their abbreviations, are listed below. Detailed descriptions follow, in the same order as the list. The codes accepted in change mode are not included here; they are listed within the detailed description of the change request.

```
find, f
next, n
- (minus sign)
top, t
add, a
delete
define
redefine
undefine
print, p
pall, pa, p*
write, w
quit, q
verify, v
global_change, gc
change, c
```

NOTES ON REQUEST DESCRIPTIONS

For convenience, the following format is used in the request descriptions.

1. If a request has one or more short names, they are listed below the long name.

2. Any arguments the request takes are shown only with the long name.

3. The shift_spec argument used by the define, redefine, and undefine requests is described under the change request, which also uses it.

LIST OF DETAILED REQUEST DESCRIPTIONS

find entry, f
: moves the pointer to the entry (group or work class) whose name is given, and prints its type and name. If the entry does not exist, the pointer is not moved.

next, n
: moves the pointer to the next entry, and prints its type and name. If the pointer is already at the last entry, it remains there, and "EOF" is printed.

- (minus sign)
: moves the pointer to the previous entry and prints its type and name. If the pointer is already at the first entry, it remains there, and "TOP" is printed.

top, t
: moves the pointer to the first entry and prints its type and name.

add group constant work_class, a
: adds a new group, after the last group in the MGT. The first argument is the group name, which can be a maximum of eight characters, and must begin with a capital letter. The second argument is the constant in the maximum primary users equation for the new group. (The group is allowed at least this number of primary users, on any configuration of the system.) The third argument is the work class that all users in this group (both interactive and absentee) are assigned to, on all defined shifts. After the group has been added, the change request can be used to modify its parameters or add new ones.

  The pointer is set to the new group. If the group already exists, a message to that effect is printed, and the pointer is set to the existing group. The add request can only add groups. Entries for the 16 possible work classes always exist, although the unused ones are marked as undefined. The change request can be used to define an undefined work class. An attempt to add one of the 16 work classes produces an error message, but leaves the pointer set to that work class.

delete
: deletes the current load control group, if the current pointer points to a load control group. If the current pointer points to a work class, an error message is printed. See the note under the verify request regarding restrictions on deleting groups.

define shift_spec {like shift_number}
: sets the group and work class values for the specified shifts to valid, consistent values. If the optional "like shift_number" parameter is given, the values are set to the respective values from the specified shift. Otherwise, the values are set to defaults: all groups are placed in work class 1; work class 1 is given 100% of available virtual CPU time; its scheduling mode is set to normal; and normal scheduling mode is set to percent. If the specified shifts are already defined, an error message is printed, and no changes are made. For example, to define shifts 2 through 4 to be exactly like shift 1, type:

```
define shifts 2-4 like shift 1
```

redefine shift_spec {like shift_number}
: operates like the define request, except that it can be used to change the values for a shift that is already defined.

undefine shift_spec
: sets the group and work class values for the specified shifts to null values, and undefines all work classes on the specified shifts. It is recommended that shifts not used at the site be undefined. This minimizes the output produced by the pall request and prevents the verify request from printing extraneous error messages.

print, p
: prints all information about the current entry.

pall type1 type2 ..., pa, p*
: prints all information about some or all entries or the header. When no type arguments are given, the default is to print all information in the MGT. Valid type arguments are:

total, tot
: print header

group, lcg
: print all load control groups

work_class, wc
: print all work classes

xref, cref
: print cross reference showing which groups are in each work class, on each shift

Printing of a cross reference should only be attempted for an MGT that passes the consistency tests made by the verify request. The results of a cross reference of an inconsistent MGT are unpredictable.

write, w
: writes the edited copy of the MGT back into the original. (Editing is performed on a temporary copy.)

quit, q
: exits from ed_mgt. If a write request is not issued before quitting, the editing is lost, and the original copy remains unchanged.

verify max_errors, v
: examines the edited MGT for internal consistency and correctness, and reports errors that it finds. The max_errors argument is an integer that indicates how many errors are reported before the user is asked if the verify request should be continued. This argument is optional; if omitted, a default value of 5 is used.

  Errors usually involve conflicts between group and work class definitions, and these fall into repeating patterns, especially since errors are reported separately for each shift. Thus, a single editing error can produce a large number of error messages, only the first few of which are useful. The output of verify can safely be aborted with the quit-pi sequence, but the max_errors argument usually makes this unnecessary. After max_errors messages have been printed, the user is asked if the verify request should be continued. If the answer is yes, the question is repeated every max_errors messages thereafter.

  The verify request detects all errors that cause installation of the MGT to be refused, except for one: an attempt to install an MGT that deletes or renames a load control group that is currently in use. An in-use load control group is one that some logged-in users belong to or one that is named in the SAT as the default group for one or more projects. Deleting or renaming such a group is not allowed, but this error cannot be detected until the attempt is made to install the new MGT.

  In addition, the verify request detects two conditions that are not fatal and do not prevent installation of the MGT, but are probably oversights on the part of the user: a work class with no groups (and therefore no users) in it, and the sum of work class percentages being less than 100% on any defined shift.

global_change entry_type change_arguments, gc
: allows the specified changes to be made to all groups or all defined work classes. The entry_type argument may be "group", "lcg", "work_class", or "wc". This request enters change mode, and accepts a series of code-qualifier-value groups, just as the change request does. Each change is made to all entries of the specified type before the next change is processed. Values of "." or "*" for code cause an exit from change mode and return to ed_mgt request level.

change code qualifiers values ..., c

puts ed_mgt into change mode, in which successive code-qualifier-value groups are read and processed. A code of "." or "*" causes an exit from change mode and return to ed_mgt request level. Except for the header codes (prio_sked, normal_mode, and shifts), changes are made to the entry pointed to by the pointer. (See the global_change request to make changes to all groups or all work classes.) Except for one code (absentee), each code applies to only one of the entry types. The set of possible qualifiers and values is, in general, different for each code. Some of the codes accept no qualifiers and only one value. Most of the codes have a full name and one or more abbreviations. Either the full name or the abbreviation may be typed. The codes and their abbreviations are listed below, grouped according to the entry type to which they apply. Detailed descriptions follow.

```
Group
  id                                  Header
  constant, const, con                  prio_sked, prio
  numerator, num                        normal_mode, norm
  denominator, denom,                   shifts
  max_prim, maxp, maxu, m
  abs_max, abs,

Group and Work Class                Work Class
  absentee                            percent, pct, %
  num1                                defined, def
  denom1, den1                        int_resp, ir
  work_class, wc                      int_quantum, iq
  absentee_max                        resp, r
  absentee_min                        quantum, q
  absentee_pct
```

There are several qualifier and value formats and many relationships between the two that are common to several of the codes. These are described here, instead of being repeated under each code.

There are three types of qualifiers: the shift specification (shift_spec), the queue specification (queue_spec), and the interactive/absentee indicator (int/abs). There are three types of data values: floating point (float), integer (int), and yes/no (y/n).

The ed_mgt command prompts for codes, qualifiers, and values, if they are not typed on the same line as the change request. For those codes that accept qualifiers, if the code and the values are typed on the same line and the qualifiers are omitted, defaults are assumed for the qualifiers. (The defaults are a function of the number of values given.) If the values are not typed on the same line as the code, then the ed_mgt command prompts for the qualifiers and they must be given. For example, the line:

```
change work_class 3 .
```

assumes default qualifiers of "shift all" and "interactive", while the line:

```
change work_class
```

makes the command prompt for both the shift specification and the interactive/absentee indicator (as well as the work class value).

The shift specification (shift_spec) has three formats:

```
shift s
shift s1-s2
shift all
```

When a single shift number is given (the first format), the values that follow are assigned to shifts s, s2, ... sN for as many values as are given, where s2 ... sN represents the set of defined shifts whose numbers are greater than s. Shifts for which no values are given remain unchanged.

The second format, giving a pair of shifts (where s2 must be greater than s1), is followed by one value, which is assigned to shifts s1 and s2 and to all of the defined shifts whose numbers fall between s1 and s2. Shifts not included in the s1-s2 remain unchanged.

The third format is equivalent to giving the second, where s1 and s2 are the lowest and highest numbered defined shifts, respectively.

When the shift specification is omitted, and one or more values are typed on the same line as the code, the default shifts are a function of the number of values given; that is, a single value is assigned to all defined shifts, while each value in a group of values is assigned to shift s1, s2, ... respectively, for all defined shifts.

The interactive/absentee indicator (int/abs) can be one of the following: interactive, absentee, int, or abs. It is used with the work_class code and may be given before or after the shift specification. If it is not given, the default is interactive.

The queue specification (queue_spec) consists of the word "queue" (or "q") followed by one or more absentee queue numbers, or the word "none". The queue numbers may be in the range 1 to 4. The notation q1-q2 may be used, to indicate a range of queues; for example:

```
queue 1 2 3
queue 1-3
```

are equivalent. This specification is used with the absentee code while changing a group.

Integer values (int) are simply decimal integers.

Floating point values (float) may be given for any of the group codes except "id" and "work_class". They differ from integer values only in that they (optionally) end with a decimal point and a single-digit fraction (tenths), or a two-digit fraction (hundredths).

The yes/no values (y/n) can be represented by a variety of positive or negative words:

yes	no
y	n
on	off
1	0
allowed	not_allowed
	^allowed

LIST OF DETAILED CHANGE CODE DESCRIPTIONS

prio_sked y/n, prio
sets the prio_sked_enabled switch in the header of the MGT. When this switch is on, users are placed in the work classes specified in the body of the MGT, and those work classes are given the percentages of system resources specified in the body of the MGT. When the switch is off, the work class information in the body of the MGT is ignored and all users are placed in work class 1; thus, work class 1 is given 100% of system resources, effectively disabling the priority scheduler. This can be useful to compare system performance with and without the priority scheduler enabled and also to recover immediately from the effects of having made an unwise change to the MGT, causing undesirable system behavior.

normal_mode {shift_spec} MODE, norm
sets the normal scheduling mode for the specified shifts to MODE, where MODE can be "deadline" or "percent" ("pct","%"). This scheduling mode is in effect for all work classes whose mode is set to normal (rather than realtime) on the given shifts.

shifts shift_list
sets the list of defined shifts (those in use at the site) to the given list. The list may consist of any combination of single digits or pairs of digits separated by hyphens. For example:

```
c shifts 1-4 7
```

indicates that shifts 1, 2, 3, 4, and 7 are used. The list of defined shifts is used in interpreting the shift_spec parameter that is an argument to a number of the ed_mgt requests, and it is used to supply a default shift specification when none is given.

id group_name
changes the name of the group. See the note under the verify request regarding restrictions on renaming groups.

constant float, const, con
: is the constant in the maximum primary units equation (described in "Load Control Equations and Group Parameters" below).

numerator float, num
: is the numerator in the maximum primary units equation (described in "Load Control Equations and Group Parameters" below).

denominator float, denom, den
: is the denominator in the maximum primary units equation (described in "Load Control Equations and Group Parameters" below).

max_prim -1, maxp, maxu, m
: The only valid value for max_prim is -1. Normally, max_prim is the lefthand side of the maximum primary units equation, and is computed each time a user logs in, from the constant, numerator, denominator, and the current system maximum units figure. A value of -1 indicates that, instead of being computed that way, the maximum primary units figure for this group should be computed by deducting from the system maximum units figure the computed maximum primary units figures for all the other groups, and giving this group what is left over. Thus, only one group may have a max_prim value of -1.

  To remove a -1 max_prim from a group so that its max_prim value is computed normally, type a value of zero for max_prim and ignore the resulting warning message.

abs_max float, abs, minamax
: is the constant in the absolute maximum units equation (described in "Load Control Equations and Group Parameters" below).

num1 float
: is the numerator in the absolute maximum units equation (described in "Load Control Equations and Group Parameters" below).

denom1 float, den1
: is the denominator in the absolute maximum units equation (described in "Load Control Equations and Group Parameters" below).

work_class shift_spec int/abs int ... int, wc
: sets the per-shift work classes to which users in this group are assigned. A separate set of work class assignments is maintained for interactive and absentee processes. If int/abs is omitted, the default is interactive. A work class of zero (for both interactive and absentee) should be set for shifts not in use at the site.

absentee_max float
: is the upper limit in the absentee_limit equation (described in the Notes section of the ed_installation_parms command). A value of 3276.7 indicates that no absolute upper limit should be imposed.

absentee_min float
: is the lower limit in the absentee_limit equation (described in the Notes section of the ed_installation_parms command).

absentee_pct float
: is the maximum percent of absentee slots that may be occupied by users from this group. It is used in the absentee_limit equation (described in the Notes section of the ed_installation_parms command). A value of 100.0 indicates that no percentage limit should be imposed.

absentee
: The absentee code applies to both groups and work classes, but it has a different format for each of the entry types:

```
Group: absentee y/n
```

sets the absentee_allowed switch, which determines whether absentee jobs for users who are normally in this group are allowed to remain in the group or are moved to a default absentee group.

```
Group: absentee queue_spec
```

determines whether or not this group is the default absentee group for one or more absentee queues. A queue_spec of "queue none" removes the group from the set of default absentee groups.

The default absentee group feature allows absentee jobs to be put into work classes separate from interactive users, either to guarantee or to limit the portion of system resources that they consume in competition with interactive users. It is possible to put jobs from one or more queues into separate work classes, by putting them in separate groups, and assigning those groups to separate work classes.

There may be only one default group for each absentee queue; however, a single group may be the default group for more than one queue.

When the default absentee group feature is used, the absentee_allowed switch should be turned off in all groups except the default absentee groups to force absentee jobs to be removed from their normal groups and placed in the default absentee groups during login.

```
Work Class:  absentee shift_spec y/n ... y/n
```

sets the per-shift absentee_allowed switch of the work class. This switch is provided for consistency checking. It should be turned on for those work classes in which absentee jobs are expected to run, and off for all others. Then, the verify request detects any instances where the parameters of a group assign absentee jobs to a work class that is reserved for interactive users.

percent shift_spec int ... int, pct, %
: sets the per-shift minimum percent of the work class. The sum of percentages of all work classes, on each shift, must be <= 100%. Work classes must have nonzero percentages on any shift on which they are defined. Refer to "Scheduler Parameters" below for a more complete discussion.

max_percent shift_spec int ... int, max_pct
: sets the per-shift maximum percent for the work class. This parameter is effective only for non-realtime work classes when the scheduler is operating in percent mode. In that case, this parameter represents the maximum percent of system resources that may be consumed in aggregate by this work class during each shift. A value of 0 means that the work class has no resource limit (this is the default). The value specified must be in the range 0 to 100. Refer to "Scheduler Parameters" below for a more complete discussion.

defined shift_spec y/n ... y/n, def
: sets the per-shift defined switch of the work class. All unused work classes should be set to undefined on all shifts, and all used work classes should be set to undefined on those shifts that are not in use at the site.

int_resp {shift_spec} float ... float, ir
: sets the per-shift interaction response time for the work class. This time is used only when scheduling in deadline or in realtime mode. An attempt is made to run the process within the specified time, after an interaction. The time is in seconds, and is kept in hundredths of a second. The default is 4.00 seconds.

int_quantum {shift_spec} float ... float, iq
: sets the per-shift interaction quantum for the work class. The quantum is only used when scheduling in deadline or in realtime mode. It is the amount of virtual CPU time a process in this work class receives, after an interaction, before losing eligibility. The time is in seconds, and is kept in hundredths of a second. The default value is 0.50 seconds.

resp {shift_spec} float ... float, r
: sets the per-shift steady state response time for the work class. This time is only used when scheduling in deadline or in realtime mode. An attempt is made to run the processes in this work class every N seconds, where N is the specified response time. The time is in seconds, and is kept in hundredths of a second. The default is 32.00 seconds.

quantum {shift_spec} float ... float, q
: sets the per-shift steady state quantum for the work class. The quantum is only used when scheduling in deadline or in realtime mode. It is the amount of virtual CPU time a process in this work class is given before it loses eligibility. The time is in seconds, and is kept in hundredths of a second. The default is 1.00 seconds.

mode {shift_spec} NORM/RT
: sets the per-shift scheduling mode for the work class to either realtime mode, or to the normal mode in effect for the shift. NORM/RT can be "normal" ("norm") or "realtime" ("rt"). When realtime mode is used, the deadline scheduling parameters -- response times and quanta -- are used. The response times are interpreted as real times, and processes in the work class are run within the specified real times, if at all possible.

NOTES ON SCHEDULER PARAMETERS

The hardcore scheduler (traffic control) operates in either "deadline mode" or "percent" mode, as specified for each shift by the "normal_mode" change code, described above. Regardless of the scheduler mode, the scheduler attempts to meet the specified responses for realtime work classes. If necessary, processes in realtime work classes are given priority over processes in non-realtime work classes. Also, among processes in realtime work classes, those whose response is late are given priority over those whose response is not late. The general effect of this is that realtime work classes are given priority over non-realtime work classes, and realtime work classes with shorter values of resp are given priority over realtime work classes with longer values.

The distribution of CPU resources among non-realtime work classes is determined by the scheduler mode in effect at the time. If the scheduler is in deadline mode, the scheduler attempts to meet the specified response times for each work class. Generally, work classes with lower values of resp are given priority for eligibility (memory) over those with higher values. Processes in work classes with higher values of quantum remain in memory longer than those with lower values.

If the scheduler is in percent mode, the CPU time available for use (i.e., not used by realtime work classes and not idle) is distributed among the non-realtime work classes on the basis of their assigned values of percent. For example, if two work classes each have values of 20 for percent, they will get approximately the same amount of CPU time in any interval where both have sufficient demand. If one work class has a value of 10 for percent, and another work class has a value of 20, then the first will get approximately half the amount of CPU time as the second. Again, it is assumed that there is sufficient demand for this to be possible.

By using the max_percent change code, the system administrator can limit a work class to a specified percentage of system resources. This control mechanism is effective only for non-realtime work classes, and only when the scheduler is operating in percent mode. If, for example, a work class is assigned a max_percent of 10, then it is limited to 10-percent of the total CPU time on the system in any interval of time. To enforce this restriction, the system will go idle, if necessary, rather than allow a work class to consume more than its max_percent of total CPU time. In practice, it may be useful to assign non-zero values of max_percent to some work classes, while assigning a value of zero (no limit) to others. In such a case, the scheduler will limit the resources of the first set of work classes to the percentages specified. It will then divide the remaining resources among the work classes with no limit, in proportion according to their values of percent.

NOTES ON LOAD CONTROL EQUATIONS AND GROUP PARAMETERS

The load control equations are:

```
max_prim     = minu    +   (num/denom)*system_maxu
absolute_max = minamax + (num1/denom1)*system_maxu
```

The max_prim equation for a group determines the maximum number of users in the group that may be logged in as primary users. Additional users from the group may be logged in as secondary users (subject to preemption) if the system is not full and the group's absolute_max figure is not exceeded. The absolute_max equation determines the absolute maximum number of users from the group (both primary and secondary) that may be logged in. This limit is not exceeded, even if the system is not full. Setting the value "3276.7" for the minamax parameter of a group indicates that no absolute_max limit is to be imposed on the group.

The equations actually determine the maximum load units (rather than the maximum users) for the group. By default each user is given a load control weight of one unit, making the number of users from each group equal to the number of units. Weights other than 1 can be assigned (see the description of the SAT in the *Multics System Administration Procedures* manual, Order No. AK50 and the admin_util command in this manual).

The system_maxu figure is the maximum units for the system (which is a function of shift and configuration), less the number of consoleless daemons and absentee jobs currently logged in. For example, if the maximum units is 80, but there are 7 daemons and 3 absentee jobs in the system, the value of system_maxu used in the load control equations will be 70, not 80.

The num, denom, num1 and denom1 variables in the load control equations are used to specify a percentage of the system_maxu figure. If the denom and denom1 figures are set to the typical system system_maxu figures (70 in the above example, rather than 80), then the num and num1 figures can be set to the desired number of users from the group (less the minu or minamax figure) that should be given primary status on a full configuration. If system_maxu falls below the full configuration value, the number of primary users allowed in each group gets reduced proportionally.

Setting num or num1 to zero disables the proportional load unit setting for the group, leaving the respective units figures as a constant, not dependent on system_maxu.

EXAMPLES

On a system with an 80-user capacity that normally runs 7 daemons and 3 absentee users, a group that should normally have 15 primary users, with a minimum of 3, set:

```
minu  = 3
num   = 12
denom = 70
```

To limit this group to 30 users with a full configuration, but to no fewer than 15 users on any configuration, set:

```
minamax = 15
num1    = 15
denom1  = 70
```

Since the system_maxu figure is a function of current load, as well as shift and configuration, the two equations are re-evaluated for each group, each time any user tries to log in. Thus, the max_prim and absolute_max figures are changing continually, and cannot be set or displayed by ed_mgt. There is one exception: the max_prim parameter for (only) one group may be set to "-1" to indicate "all the rest." The max_prim equation for that group is not evaluated, but rather, its max_prim figure is determined by deducting from system_maxu the max_prim figures computed for all other groups.

The following list gives the keywords used in the ed_mgt change request to change the values of the variables in the load control equations:

Variable	Keywords
max_prim	max_prim, maxp, maxu, m
minu	constant, const, con
num	num, numerator
denom	denom, den, denominator
minamax	minamax, abs_max, abs
num1	num1
denom1	denom1, den1

The variable shown as "minu" in the equation is the only one whose name is not one of the keywords accepted by the change request to change that variable. (Three variations on the word "constant" are used instead.)

EXAMPLES OF WORK CLASS PARAMETER SETTING

The following examples are present only to illustrate the use of ed_mgt, and the use of the load control group and work class mechanisms to achieve the objectives of management regarding the division of resources among users. They are not intended to be recommendations of specific ways of allocating those resources. This is especially true of the work class percentages used in the examples -- the only meaning that may safely be attached to those numbers is that they must add up to no more than 100%.

The first example is based on the assumption that the MGT is in the state in which it is left when a new Multics site is initialized (3 groups, 1 work class). The others follow sequentially, with the assumption that the MGT is in the state in which it was left by the previous example.

There are several places in the examples where a group of parameters is set to the same (usually null) value by one request, and then a subset of those is set to other (nonnull) values by subsequent requests. This takes advantage of the global-change and change-all-shifts features of the ed_mgt syntax, to minimize the amount of typing needed to accomplish a change.

The ed_mgt prompting messages are omitted from the examples.

It is recommended that the user work through these examples at a terminal, using a temporary copy of the MGT, and using the "p*" request to display the contents of the MGT at the end of each example.

EXAMPLE 1

The following simple set of group and work class definitions could be used to gain experience with the priority scheduler, before more complicated uses of it are attempted:

Group	Work_Class	Example_Percent
System	1	15
SysProg	2	15
Other	3	70

Assuming that the groups System, SysProg, and Other, and work class 1 (the defaults that are defined when a Multics system is first set up), and no others, are defined, and that shifts 1-4 (the defaults, as above) are in use, the following ed_mgt requests edit the MGT to define the above work class assignments:

```
gc group wc int 0
    wc abs 0 .
gc wc defined no .
f 1 c defined shift 1-4 yes
 % shift 1-4 15
 absentee shift 1-4 allowed .
f 2 c defined shift 1-4 yes
 % shift 1-4 15
 absentee shift 1-4 allowed .
f 3 c defined shift 1-4 yes
 % shift 1-4 70
 absentee shift 1-4 allowed .
f System c wc int shift 1-4 1
     wc abs shift 1-4 1 .
```

```
f SysProg c wc int shift 1-4 2
      wc abs shift 1-4 2 .
f Other c wc int shift 1-4 3
    wc abs shift 1-4 3 .
```

The verify request may then be used to check for possible typing errors, and the pall request may be used to display the contents of the edited MGT. Neither of those requests should be used when the above editing is only partly done, since the MGT will be inconsistent (for example, the first three requests leave no work classes defined, and all groups in work class zero -- an invalid state) and pall or verify would produce many error messages.

EXAMPLE 2

Add groups IO and Backup; assign them to work classes 4 and 5 respectively. Define those work classes, assigning 5% to each. Take 10% from work class 1, which corresponds to group System (this group originally contained the SysDaemon and SysAdmin projects, but now has -- practically -- only SysAdmin, since the two most active daemons have been moved to separate work classes).

```
a IO 1 0
c wc int shift 1-4 4
  wc abs shift 1-4 4 .
a Backup 1 0
c wc int shift 1-4 5
  wc abs shift 1-4 5 .
f 4 c defined shift 1-4 yes
 % shift 1-4 5
 absentee shift 1-4 allowed .
f 5 c defined shift 1-4 yes
 % shift 1-4 5
 absentee shift 1-4 allowed .
f 1 c % shift 1-4 5 .
```

Then, using edit_proj, give the SysDaemon project the igroup attribute, and set its authorized groups to IO and Backup. Edit the SysDaemon PMF, to give IO and Backup the igroup attribute, and assign them to groups IO and Backup, respectively. Of course, the PMF must be compiled, and the PDT and SAT installed. The required order of installations is: MGT, SAT, then PDT. If the daemons are already logged in, they will remain in their old groups and work classes until the next time they are logged in (presumably at the next bootload).

EXAMPLE 3

To assign absentee jobs to work classes different from those of interactive users, take the following steps.

Clear all absentee allowed switches and zero all absentee work class assignments. Then define groups Abs1 and Abs2-4, for queues 1 and 2-4 respectively. Put queue 1 in work class 6 and give it 10%; put the others in work class 7, giving it 5%; take 15% away from the interactive users' work class.

```
gc wc absentee shift 1-4 not_allowed .
gc group absentee not_allowed
    wc abs shift 1-4 0 .
a Abs1 3 0
c wc int shift 1-4 6
  wc abs shift 1-4 6
  absentee allowed
  absentee queue 1 .
a Abs2-4 1 0
c wc int shift 1-4 7
  wc abs shift 1-4 7
  absentee allowed
  absentee queue 2-4 .
f 6 c defined shift 1-4 yes
 % shift 1-4 10
 absentee shift 1-4 allowed .
f 7 c defined shift 1-4 yes
 % shift 1-4 5
 absentee shift 1-4 allowed .
f 3 c % shift 1-4 55 .
```

EXAMPLE 4

This example shows how to assign work class percentages and memberships that vary with shift.

There are likely to be more interactive users on the system during normal working hours (shift 1) than at other times. Therefore, decrease the interactive users' percent by 20% and increase that of absentee, on shifts 2-4. On shift 3 (midnight to 8 am), put all four absentee queues into one work class.

```
f 3 c % shift 2-4 35 .
f 6 c % shift 2-4 20
 % shift 3 35 .
f 7 c % shift 2-4 15
 % shift 3 0
 defined shift 3 no .
f Abs2-4 c wc int shift 3 6
      wc abs shift 3 6 .
```

Name: edit_process_audit_flags

SYNTAX AS A COMMAND

```
edit_process_audit_flags {flags_str}
```

FUNCTION

turns on or off only those flags given in the flags_str.

ARGUMENTS

flags_str
: is an audit flags string acceptable to convert_access_audit_flags_. If you omit it, the command enters a prompt loop in which you are asked for an audit flags string. A "." entered alone on a line causes the loop to exit.

ACCESS REQUIRED

You must have re access on system_privilege_ and access_audit_gate_ and r to >udd>SysAdmin>admin>sys_admin_data if you use the "default" keyword.

NOTES

For a description of the process audit flags, see the new_user command and the section "Security Auditing" in the *System Administration Procedures* (AK50).

This command is meant for experimentation with security audit control flags by the system security administrator. Permanent settings for processes should be placed in the appropriate PNT/SAT entries with the new_user and edit_proj commands.

The special keywords "none", "all", and "default" are also recognized for the audit flags string. They specify

```
"none"     all audit flags turned off
"all"      all audit flags turned on
"default"  audit flags set to default found in sys_admin_data.
```

See also the display_process_audit_flags and set_process_audit_flags commands.

EXAMPLES

```
edit_process_audit_flags rcp=ma/ma,admin=n/,special=/n
display_process_audit_flags
fsobj=N/R,fsattr=N/R,rcp=MA/MA,admin=N/R,special=R/N,...
   ...other=N/R,admin_op,priv_op,fault,^small_cc,^moderate_cc

edit_process_audit_flags
current flags: fsobj=N/R,fsattr=N/R,rcp=N/R,admin=R/R,...
  ...special=R/R,other=N/R,admin_op,priv_op,...
     ... fault,^small_cc,^moderate_cc
enter flags:   fsobj=r/
new flags:     fsobj=R/R,fsattr=N/R,rcp=N/R,admin=R/R,...
  ...special=R/R,other=N/R,admin_op,priv_op,...
     ...fault,^small_cc,^moderate_cc
enter flags:   fsobj=/m
new flags:     fsobj=R/M,fsattr=N/R,rcp=N/R,admin=R/R,...
  ...special=R/R,other=N/R,admin_op,priv_op,...
     ... fault,^small_cc,^moderate_cc
enter flags:   ^admin_op,^fault,other=ma/ma
new flags:     fsobj=R/M,fsattr=N/R,rcp=N/R,admin=R/R,...
  ...special=R/R,other=MA/MA,^admin_op,priv_op,...
     ...^fault,^small_cc,^moderate_cc
enter flags:   .
```

Name: edit_proj

SYNTAX AS A COMMAND

```
edit_proj Project_id {keyword {newvalue}}
```

FUNCTION

provides a single facility for the editing of all project data. You can invoke it in two ways. Either all data items that are part of project registration can be printed one at a time or a single data item can be changed.

ARGUMENTS

Project_id
is the Project_id of the project whose registration data items are to be edited. If this is the only argument you specified, edit_proj prints each data item one at a time and waits for a response from the accounting administrator before proceeding. The accounting administrator can respond with any one of the following:

carriage return
to leave the item unchanged

a new value
to replace the printed value

stop
to exit immediately from the edit_proj command without making any changes

keyword
is the particular data item to be changed. Valid keywords are

title	t
investigator	inv
investigator_address	inv_addr
supervisor	sup
supervisor_address	sup_addr
supervisor_phone	sup_phone
account	acct
requisition	req
requisition_amount	amt
cutoff_date	cutoff
billing_name	billto
billing_address	billat
group	grp
attributes	attr
grace	gr
administrator	admin
quota	q
alias	aka
groups	gps
min_ring	min
max_ring	max
max_foreground	maxfg
max_background	maxbg
abs_foreground_cpu_limit	absfgcpulim
pdir_quota	pdq
rate_structure	rs
accounting_category	acct_cat

newvalue
: is the new value of the data item identified by the specified keyword. When you give newvalue, give keyword. If you omit newvalue, edit_proj asks for it. In both cases, the old value is printed.

NOTES

The meanings of the following parameters are explained in the Project Administrator's Manual (AK51), under the PMF description.

```
group               max_foreground
attributes          max_background
grace               abs_foreground_cpu_limit
min_ring            pdir_quota
max_ring
```

EXAMPLES

To review all the registration data items of the States project and change the project's quota to 200, type:

```
   edit_proj States

Title:                 Data Reduction for all States

Investigator:          Mr. M. Lee

Address:               Room 22

Supervisor:            Ms. Hilkka Bjorn

Address:               Room 6

Phone:                 555-1212

Requisition:            P03344-J

Amount:                open

Cutoff date:           01/01/86 1935.1

Billing name:          Fiscal Office

Address:               Room 8

Rate structure:        default

Group:                 Other
```

```
Attributes:            anonymous, preempting, brief, vinitproc,
                       vhomedir, nostartup, v_outer_module

Grace:                 2880

Project administrators.  Type "." to delete.
Administrator ID: Brekke.States

Administrator ID:

Quota:                 100
     200
Alias:

Do you wish to review?
     no
r 19:39 2.661 197

From Initializer.SysDaemon.z (install)  1940.0:
installed sat for SA1.SysAdmin
```

Another way to accomplish the same quota change is to type:

```
   edit_proj States quota 200
```

If the accounting administrator requests a change to either the requisition or account number of the project, he is asked a question to which he must respond with one of the following words:

drop
: to cause the charges to the old account and requisition to be eliminated

transfer
: to cause the charges to be transferred from the old account and requisition to the new account and requisition

bill
: to cause the charges to be billed to the old account and requisition, and to cause the new account and requisition to start off with a clean slate.

For example:

```
   edit_proj States req P05566-J
What is the disposition of charges of
$1233.79 to account 70906, req P03344-J?
   bill
r 19:39 2.661 197
```

Name: edit_proj$change_all

SYNTAX AS A COMMAND

```
edit_proj$change_all keyword {old_value} {new_value {-all}} {-long}
```

FUNCTION

changes project registration information for all projects registered on the system. (The accounting administrator can use the edit_proj command to change the registration information for a single project.)

You can invoke this command in several ways, as described below:

1. In the format "edit_proj$change_all keyword," in which case you are queried for a change to "keyword" for each project on the system.

2. In the format "edit_proj$change_all keyword old_value," in which case you are queried for a change to "keyword" for each project for which the value of "keyword" is "old_value."

3. In the format "edit_proj$change_all keyword old_value new_value," in which case the "old_value" of keyword is set immediately (without a query) to "new_value."

4. In the format "edit_proj$change_all keyword new_value -all," in which case you are queried to make sure you want to set the value of "keyword" to "new_value." If you respond "yes," the change is made immediately.

ARGUMENTS

keyword
is the particular keyword whose value is to be changed. Valid keywords are

title	t
investigator	inv
invetigator_address	inv_addr
supervisor	sup
supervisor_address	sup_addr
supervisor_phone	sup_phone
account	acct
requisition	req
requisition_amount	amt
cutoff_date	cutoff
billing_name	billto
billing_address	billat
group	grp

This page intentionally left blank.

attributes	attr
grace	gr
administrator	admin
quota	q
dir_quota	dq
alias	aka
groups	grps
min_ring	min
max_ring	max
max_foreground	maxfg
max_background	maxbg
abs_foreground_cpu_limit	absfgcpulim
pdir_quota	pdq
rate_structure	rs
accounting_category	acct_cat
authorization	authorization
audit	audit

old_value
: is the current value associated with keyword. This argument cannot be specified unless the keyword argument is also specified.

new_value
: is the new value to be associated with keyword. This argument cannot be specified unless (1) the keyword argument and the old_value argument are both specified, or (2) the keyword argument and the -all control argument are both specified.

CONTROL ARGUMENTS

-all
: specifies that all the specified project keywords are to be immediately changed after a single query. This control argument can be used only as described in format 4, above.

-long
: specifies that the system issue a message when a change becomes effective.

EXAMPLES

to specify that the system query for a new value for the title of all projects on the system, type:

```
edit_proj$change_all title
```

To specify that the system query for a new value of dir_quota for all projects in which the current value of dir_quota is 10, type:

```
edit_proj$change_all dir_quota 10
```

To specify that the system change the value of dir_quota for all projects to 10 after a single query, type:

```
edit_proj$change_all dir_quota 10 -all
```

To specify that a new dir_quota value of 100 be specified for all projects whose current dir_quota value is 10 and that the change is to take place without a system query, type:

```
edit_proj$change_all dir_quota 10 100.
```

Name: eis_tester, et et >t>eis -bf

SYNTAX AS A COMMAND

```
et path {-control_args}
```

FUNCTION

sets up and tests EIS instructions in a controlled environment. You must prepare an input script describing the EIS instructions to be tested. From this input script the EIS tester builds the EIS instructions (one at a time) and the indirect words, descriptors, and data that each instruction needs. The instruction to be tested is set up in a special ALM segment (etx). The eis_tester command calls etx in order to execute the EIS instruction; etx returns to eis_tester when the instruction has been executed. After executing the instruction, eis_tester tests correct execution of the instruction. If one of the test scripts in the eis_tester data base fails and the successful execution of that test is dependent upon installation of a particular FCO, the FCO number is displayed in the error message.

ARGUMENTS

path
: is the pathname of a segment that contains input script data that defines the instructions to test.

CONTROL ARGUMENTS

-brief, -bf
: suppresses all output except identification and error messages.

-nox
: sets up the instruction but does not execute it; used to test the validity of the input script.

-debug
: runs the test in a debugging loop where each instruction is tested 10 times but results from the test are not checked. Each time through the loop the instruction is set up completely, including all the specified faults.

-select N, -sel N, -do N
: processes only test N (where N is a positive decimal number). This number has no relationship to the -ns field in any test.

-help
: displays a brief usage statement.

-instruction_type INSTR, -inst INSTR
: processes only tests that contain the instruction INSTR.

-long, -lg
: displays all the related test information prior to executing a test.

-repeat N, -rpt N
: repeats the entire execution of the selected tests N times.

-stop_on_failure, -sof
: displays the failing data, machine condition, and history register information and return to command level if an error is detected in a test. The default is to display the failing data and continue with the next test.

-from N, -fm N
: starts processing test N (where N is a positive decimal number) and continues processing all remaining tests in the input segment unless -to is used.

-to N
: stops processing after test N (where N is a positive decimal number). If -from is not used, tests one through N are processed.

NOTES

Before eis_tester calls etx to execute the instruction, it sets up some special padding around the data field that is modified by the EIS instruction. Eight special characters (octal 717) are put in front of and at the end of the result data string. The result area itself is initialized to all zero bits.

When called to execute the EIS instruction, etx does the following:

1. Saves the current pointers and registers.

2. Loads the pointers and registers from values set up by eis_tester. These are the values of the pointers and registers when the EIS instruction is actually being executed.

3. Sets indicators to preinstruction test values.

   a. All indicators except b and c below are off.

   b. The BAR MODE indicator is always on.

   c. If the test instruction is expected to turn on any of the three overflow indicators (ov, eo, eu), then the om (overflow mask) indicator is turned on so an overflow fault is not taken.

4. Transfers to the instruction area in etx itself where eis_tester has set up the EIS instruction and its descriptors.

5. After the EIS instruction has been executed, etx stores the values of the indicators, pointers, and registers so that eis_tester can examine them.

6. Reloads the pointers and registers that were saved by etx.

7. Returns to eis_tester.

After the execution of the EIS instruction, eis_tester makes the following tests:

1. Checks to see that the data resulting from the instruction is correct.

2. Checks to see that the indicators are set correctly.

3. Checks to see that a truncation fault was correctly taken or not taken.

INSTRUCTION AREA

The EIS instruction is set up in a special area in etx. This area consists of seven words. The first three words of the instruction area are set up in the last three words of a page. The last four words of the instruction area are the first four words of the next page. By positioning the instruction in the instruction area, you can position the instruction on a page boundary. Those words in the instruction area that are not used for the EIS instruction itself are set up as nop instructions. The default position of the instruction word is in instruction area word 4. This places the instruction at word 0 of a page.

```
PAGE  A            WORD 1
                   WORD 2
                   WORD 3
                ---------------
PAGE  B            WORD 4             <--  Default position of
                   WORD 5                  instruction word.
                   WORD 6
                   WORD 7
```

DATA AREAS

The data referenced by each descriptor of the instruction is set up in a special data area. There is one data area used for every descriptor of the instruction. Each data area consists of three pages. The default starting position of the data is character 0 of word 0 of page 2. The last 32 words of page 1 and the first 32 words of page 3 can also be used to hold the test data. Thus the maximum data size of any string is 1088 words or 4352 (9-bit) characters. You can position the start of the data string so that it starts in page 1. Thus you can define data strings that cross page boundaries. A long data string can cross two page boundaries.

Depending on what modification is used by the instruction. the data areas used may or may not be in the same segment.

If a descriptor is referenced via an indirect word, then the descriptor is set up in a special page of its own. Depending upon the modification used in the indirect word, the descriptors may be in different segments.

PAGE FAULTS

You have control over a maximum of 13 page faults during the testing of any EIS instruction. These 13 pages have consistent meaning to eis_tester, even though for different tests they may actually be physically different pages in different segments.

The 13 pages are:

1. Page 1 of the instruction area
2. Page 2 of the instruction area
3. Page containing indirect descriptor 1
4. Page 1 of data area 1
5. Page 2 of data area 1
6. Page 3 of data area 1
7. Page containing indirect descriptor 2
8. Page 1 of data area 2
9. Page 2 of data area 2
10. Page 3 of data area 2
11. Page containing indirect descriptor 3
12. Page 1 of data area 3
13. Page 2 of data area 3

REGISTER ASSIGNMENT

You can control the type of modification used by each EIS instruction tested. However, for each type of modification (depending upon the descriptor number), eis_tester assigns the register to be used. The specific use of pointers and registers is not under your control when using the eis_tester script input method.

Pointer registers not used during the instruction are set to null (77777|1). Index registers and the A and Q registers that are not used are set to 8191 decimal (17777 octal).

AR modification involves the use of a pointer register. Both descriptors and indirect words can use AR modification. A general rule followed by eis_tester is that AR modification implies the data referenced is in an external segment. The pointer registers used by eis_tester for the EIS instruction are:

```
AR modification in a descriptor

     descriptor 1     pr1
     descriptor 2     pr2
     descriptor 3     pr3


AR modification in an indirect word

     descriptor 1     pr4
     descriptor 2     pr5
     descriptor 3     pr7
```

pr6 is used to point to your current stack frame and must be preserved in a valid state in order for any conditions to be signaled correctly.

SEGMENTS USED BY EIS_TESTER TO EXECUTE AN INSTRUCTION

Index register modification can be specified for descriptors and for indirect words. The effective offsets used for index modification are always set up by eis_tester in terms of words. For some descriptors, the value in the index register must be in units of characters. The character size also varies with the value of the ta field of the descriptor. The index registers assigned by eis_tester and the effective word offset they contain are given below.

```
Index register modification of a descriptor

     descriptor 1     X1  1 word
     descriptor 2     X2  2 words
     descriptor 3     X3  3 words

Index register modification of an indirect word

     descriptor 1     X4  4 words
     descriptor 2     X5  5 words
     descriptor 3     X7  7 words
```

RL modification can be specified for the descriptors of certain instructions. The value put in the register is specified by you. The register assigned is controlled by eis_tester. The following registers are used:

```
descriptor 1     A
descriptor 2     Q
descriptor 3     X6
```

The execution of an instruction by eis_tester can involve up to seven segments:

```
etx     eti1     etd1
        eti1     etd2
        eti3     etd3
```

The notation etiX means eti1, eti2, or eti3, depending on which of the up-to-three descriptors or operands is of current interest. Similarly, etdX means etd1, etd2, or etd3.

The first list below states the possible segments in which various items can be located, while the second list states what segment a descriptor or an operand is in under various conditions.

ITEM	SEGMENTS		
	etx	etiX	etdX
Instruction word	x		
Indirect word pointing to descriptor	x		
Descriptor	x	x	
Operand	x	x	x

Is AR Used to Access Descriptor?	Is AR Used to Access Operand?	Descriptor Location	Operand Location
No	No	etx	etx
No	Yes	etx	etdX
Yes	No	etiX	etiX
Yes	Yes	etiX	etdX

EIS_TESTER PRINTOUT

The eis_tester program prints a message noting the beginning of each instruction test. It also prints the number of this test. If there were errors, it prints the incorrect data or incorrect indicators.

If you do not specify -bf (see "Usage" above), then the data that eis_tester has set up for this instruction is printed before the instruction is executed. The following notes describe this printout:

1. Pointers enclosed in parentheses point to where the data is set up in the eis_tester segments.

2. If none of the pointer registers are used by the instruction, then none are printed. The same is true of the registers.

3. The names of the pages that take faults cannot be the names of all the pages specified in a page statement. See the last two complete examples at the end of this description for clarification.

4. If the first word of a data string does not begin at character 0 of a word, or if the string does not use all four characters of the last word, then the unused characters of the first and last words of the string are printed as blank characters.

5. The test string is not printed from one of the areas used by the instruction but rather from one of the buffers used by eis_tester.

6. The test and result strings are both padded by eis_tester with special characters. These special characters are not printed out in octal like the rest of the string; instead, each of these special characters is printed as three x's (xxx).

HOW TO CALL EIS_TESTER

The eis_tester program is the main procedure in the EIS instruction tester. It calls et_test to parse the statements in your data file. It translates these statements into the data needed to build and test an EIS instruction in the external segment etx. After building the instruction, this procedure calls etx in order to execute the EIS instruction. When etx returns, the results of the EIS instruction are examined. The eis_tester program continues to build and test EIS instructions until there is no data left in the input file. The failure of one instruction only causes the termination of that one instruction test. Any remaining instructions specified in the input file are processed and tested.

HOW TO WRITE SCRIPT INPUT TESTS

The script input test consists of a series of eis_tester statements. The first statement in any test must be an inst statement. This statement signifies the beginning of one test.

An input script segment can contain several tests. All statements from the beginning of the inst statement to the beginning of the next inst statement (or, if none is found, to the end of the segment) are considered part of the same test.

The format of a statement is as follows:

```
   name required_field {-control_args};
```

where:

name
: is the four-character statement name. There are four types of eis_tester statements:

  inst
  : defines the instruction word and many control variables.

  desc
  : defines a descriptor.

  data
  : defines the data associated with a descriptor.

  page
  : defines the page faults taken by the instruction.

  These statements are discussed in detail below.

required_field
is required information used by all but the page statement.

control_args
are optional control arguments, explained in the individual statement descriptions.

SYNTAX AND METALANGUAGE

All statements must end with a semicolon (;). There can be any number of blanks, tabs, and newline characters between any fields in the statement, including before the name field. Wherever blanks are permitted, there can also be comment fields. A comment field begins with a /* character pair and ends with the next */ character pair.

In this description, lowercase letters are used to indicate characters that are to be typed in for input to eis_tester. Uppercase letters are to be replaced with the desired character before the script is typed.

INST STATEMENT

The inst statement defines the beginning of an eis_tester test. It is used to define all of the fields in the instruction word of the EIS instruction. It is also used to set up the following special control arguments:

1. Instruct eis_tester to execute this instruction several times.
2. Position the instruction within the instruction area.
3. Define an identifying string that is printed with the test.

An inst statement has the following format:

```
inst opcode_mnemonic {-control_args};
```

where:

inst
is the four-character statement name.

opcode_mnemonic
is the mnemonic name of a storage type EIS instruction.

CONTROL ARGUMENTS

are optional and can be chosen from the following:

-tbA
turns on the truncation bit. The A is either y or n to signify whether or not the instruction is to take a truncation fault (y = yes, n = no). The default is n.

-fb
turns on the fill bit. The default is off.

-pb
turns on the plus sign bit. The default is off.

-rb
turns on the rounding bit. The default is off.

-fcA
defines the fill character to be the character specified by A. (No space between c and A and no quotes are permitted.)

-mcA
defines the mask character to be the character specified by A. (No space between c and A and no quotes are permitted.)

-ln N
defines the loop number as X. This is the number of times this instruction test is performed. The default is 1. The maximum value of X is 4.

-io N
defines the instruction offset. It is used to position the instruction relative to a page boundary. The default is 0. This places the instruction at word 0 of the second page of the instruction area. X indicates the number of words of the instruction to be placed in the first page of the instruction area. The maximum value of X is 3.

-nt "A...A"
defines a note. It can be used to identify each test. The term consists of a character string between quotes. Up to 32 characters can be used. No embedded quotes are allowed.

-bo AAA
defines a Boolean operator. AAA is the name of the operator. The names eis_tester has assigned to the Boolean operators are given below. Next to these names are the actual BOLR codes they represent.

zer	0000
and	0001
axr	0010
mov	0011

xra	0100	
ra2	0101	
xor	0110	
orB	0111	Type in orB, where B is a space
nor	1000	
nox	1001	
iv2	1010	
xrx	1011	
inv	1100	
xxr	1101	
nan	1110	
set	1111	

-ir {terms}
is a multifield control argument that defines the correct state of the indicator registers after the EIS instruction has been executed. An -ir control argument can be followed by any number of specific terms. These terms can be in any order and can be separated by any number of skip fields. Each term is a two-character identifier of an indicator register bit.

A control argument of "-ir zr" means that the zero indicator is expected to be on at the end of the EIS instruction. Valid indicator register term values are:

zr	zero
ng	negative
cr	carry
ov	overflow
eo	exponent overflow
eu	exponent underflow
om	overflow mask
tr	tally runout
pe	parity error
pm	parity mask
bm	BAR mode (always turned on by eis_tester)
tn	truncation
mw	multiword instruction interrupt fault
ab	absolute mode

If the script turns on eo, eu, or ov, then eis_tester automatically turns on the overflow mask bit in the expected indicator's result.

-mfX {terms}
is a multifield control argument that defines one mf field of the instruction. Some instructions do not have mf fields in the instruction word for all of their descriptors. The -mfX control argument is then used to specify any ar or reg modification in the descriptor itself. An example is the mvt instruction. X denotes which mf field is being defined. It must be from 1 to 3 and is associated with descriptor X. This descriptor number can be followed by up to four terms. All four terms are optional and can be specified in any order. The valid terms are:

```
ar
rl L
idA
reg
```

THE AR TERM

The ar term specifies that, for this descriptor, the address register modification is be used to access the operand. In Multics, it is called pointer register modification. The pointer assigned is prX. When this term is specified, the data referenced by this descriptor is placed in the segment etdX.

THE RL L TERM

The rl L term specifies that, for this descriptor, the register length modification is be used. This term must be followed by a decimal number L, which specifies the character length of the data. The character size is defined within a desc statement (for 4-, 6-, or 9-bit characters) or inferred from the instruction mnemonic (for bit strings). This value is placed in the selected register, and the N field of descriptor X contains the register modification tag code. The registers assigned are:

```
X = 1  -  A
X = 2  -  Q
X = 3  -  x6
```

THE IDA TERM

The idA term specifies that descriptor X is to be referenced via an indirect word in the instruction. In the idA term, the A denotes what modification is to be used in the indirect word: a for address register, r for register, or b for both.

If no A character is given in the idA term, then there is no A modification in the indirect word.

ida
specifies that address register modification is to be used to access the descriptor. When this is specified, the descriptor is placed in the segment etiX.

The pointer registers assigned to the indirect word are

```
indirect word 1  =>  pr4
indirect word 2  =>  pr5
indirect word 3  =>  pr7
```

idr
: specifies that register modification is to be used to access the descriptor. The indirect word is modified by index register 4, 5, or 7. *NOTE*: This modification is in terms of words.

idb
: specifies both a and r modification as described above.

THE REG TERM

The reg term specifies that descriptor X is to be modified by an index register. The value in the index register is a character offset and is (X*4). The index register assigned is index register X. The value placed in index register X is dependent upon the type of instruction and the appropriate character size. It is in the following units:

WORDS
: for those descriptors that have no mf field in the instruction word

BITS
: for all bit string instructions

CHARS
: for all others. The actual units depend upon the character size. The default is a 9-bit character size.

If it is necessary to write a script in which the placement of the instruction, indirect words, descriptors, and operands in specific segments is important, the following list is helpful.

Script Elements Used in-mfX Fields					Descriptor Location		Operand Location		
				ar	etx				etdX
id				ar	etx				etdX
	ida			ar		etiX			etdX
		idr		ar	etx				etdX
			idb	ar		etiX			etdX
id					etx		etx		
	ida					etiX		etiX	
		idr			etx		etx		
			idb			etiX		etiX	

EXAMPLES OF INST STATEMENTS

```
/* Example 1. */

inst     mlr      -nt  "Example 1 "

-fc*   /* Comments can go anywhere except inside a term */

-fb

-mf2  /* Note order is not important. */
          rl  3   id  ar  reg

-mf1  ar idb  reg  rl 3 ;     /* Statement must end with  ";"  */


/* Example 2. */    inst    cmpc        /* mnemonic name must
                                        *  be first term. */
-mf1  ar
-nt  " example 2"  -mf2  rl 3
-fc      /* Use escape to enter octal character   */
-ir  cr  zr ;       /*  indicator  bm  is on by default.  */
```

```
/* Example 3. */

inst      scm       -nt "scm examp."
-mc9
-ln  3              /* Make this test  3 times.  */
-io  2              /* Put instruction word and first descriptor
                    *  in page 1 of instruction area.  */
-mf1  reg ar
-mf2  ida;


/* Example 4. */

inst      ad3d
-mf3 ar  -mf2 reg -mf1 idr   /* -mfx items can be in any order */
-rb   -pb;


/* Example 5. */

inst csr -fb -bo and  -mf 2 rl  36;
```

DESC STATEMENT

The desc statements are used to specify certain fields in the descriptors. Each desc statement deals with only one descriptor. The fields in a descriptor not specifically set up by a desc or an inst statement are set to zero. If zero bits in all of the fields are needed, then no desc statement need be specified for that descriptor.

The -cp, -bp, and -cn fields of a desc statement interact with the -do field of the associated data statement. See the complete examples at the end of the eis_tester description for illustrations of the interactions.

In general, the order of the desc statements is not important, and the can be mixed in with any other statements. However, if the instruction is CMPC, SCD, SCDR, SCM, or SCMR, then the desc 2 statement cannot specify a -ta field. Descriptor 2 must use the value specified in descriptor 1. To use this feature, the desc 1 statement must precede the desc 2 statement.

A desc statement has the following format:

```
desc num {-control_args};
```

where:

desc
: is the four-character statement name.

num
: is the number of the descriptor. It must be 1, 2, or 3.

CONTROL ARGUMENTS

can be chosen from the following:

-cp N
: is used in bit string instructions to specify a (9-bit) character offset when developing an operand address where N must be a number from 0 to 3.

-bp N
: is used in bit string instructions to specify a bit offset within a 9-bit character when developing an operand address where N must be a number from 0 to 8.

-cn N
: is used in character string instructions to specify a character offset when developing an operand address where N must be a number from 0 to 7. The quantity of bits associated with each character (4, 6, or 9 bits) is specified by the N argument supplied with the -ta or -tn control argument.

-ta N
: defines the alphanumeric character type where N must be 9, 6, or 4. The default value is 9.

-tn N
: defines the type of numeric character where N must be either 9 or 4. The default value is 9.

-sd STR
: is the sign and decimal type. The STR argument must be one of the following characters:

```
f  -  Floating point, leading sign
l  -  Leading sign, scaled
t  -  Trailing sign, scaled
n  -  No sign, scaled
```

-sf N
: is the scaling factor where N is a signed (or unsigned) decimal number.

-ns N
: is the number of characters or bits in a string where N is an unsigned decimal number. There is no default value.

-nn N
: is the number of characters in a numeric string where N is an unsigned decimal number that must not be greater than 64. There is no default value.

EXAMPLES OF DESC STATEMENTS

```
/*        Example 1. */

desc  1   -ns 8  -ta 6  -cn 5;


/*        Example  2.  */

desc  3  -cp  2
-bp  /* Comments can come between control argument names
    *  and the term. */  5;
/* No -ns control argument.  This is valid if -mf3
*  control argument in inst statement
*  specified  rl  term.  */


/*        Example  3.  */

desc      2
-tn 4  -cn 3  -sd  n  /* No sign. */
-sf -100  -nn 12;
```

DATA STATEMENT

The data statements are used to describe the data that a descriptor references. Every test requires at least as many data statements as there are descriptors for the EIS instruction being tested.

The eis_tester program can determine which descriptor references the result data. The data entered for this descriptor is not set up in the data area referenced by the descriptor. Instead, this data area is initialized to all zero bits. The input data is saved and used to test the result of the instruction. Some special notes about data statements are given below:

1. For those instructions that both read and write data into the same string (e.g., ad2d, sb2d), you must enter a data 3 statement that describes the resulting data referenced by descriptor 2. The data input via the data 2 statement is the data initially referenced by descriptor 2.

2. The data pointer for each descriptor is set by default to character 0 of word 0 of page 2 of the data area for that descriptor. You can adjust this data pointer by certain (9-bit) character offsets.

3. The input string defined by you is placed in the data area starting at the first character referenced by the effective data pointer. It is important to remember this. If the descriptor associated with this data area specifies that the first character of the string is not character 0 of the first word, then the missing data must be reserved when the input string is specified.

4. The -do field of a data statement interacts with the -cp, -bp, and -cn fields of the associated desc statement. See the complete examples at the end of the eis_tester description for illustrations of the interactions.

A data statement has the following format:

```
data num {-control_arg} data_fields;
```

where:

data
: is the four-character statement name.

num
: is the number of the data field. It must be either 1, 2, or 3. In some cases, a data 3 statement is valid even when there is no third descriptor. In this case, it is used to input test data. See the last complete example (csl instruction) at the end of this description. If the descriptor that points to this data does not use address register or register modification, then only offsets that are a multiple of 4 are accepted. The data used by EIS instructions is always string type data, and thus the input modes are limited to the two described below.

CONTROL ARGUMENTS

-do X
: where X must be a decimal integer from -128 to +4096 that represents a 9-bit character offset from character 0 of the middle page of the data area.

DATA_FIELDS

are the following types. They can be intermixed. The maximum size of the data is 1088 words (4352 characters).

ASCII
: is an ASCII string. It must be enclosed in quotes. The maximum size of any one field is 256 characters. Quote characters can be entered in the string by expressing them as double quotes ("").

OCTAL
: is a string of octal digits. The first nonoctal-digit-type character found indicates the end of a string of octal data. The converted octal string is padded on the right with zero bits to make it an integral number of 9-bit characters. For example, data 123 45 6 7777; becomes 123 450 600 777700.

  The repetition factor (XX), an unsigned decimal number enclosed in parentheses, can be used to specify the repetition of a field. Only the data field immediately following the repetition field is repeated.

EXAMPLES OF DATA STATEMENTS

```
/*        Example 1.  four characters of data starting at the
*         beginning of the default data area.  */

data 3    "abcd";


/*        Example 2.   Moves the same data field back two
*         characters.  This splits the string across a page.

*         NOTE:  The input string is the same even though it is
          entered differently.  */

data  2   -do -2    "ab"  "cd";


/*        Example 3.  The same as example 2 only it specifies
*         some of the data in octal.  */

data 1    "ab"  143144        -do -2;


/*        Example 4.  A string of:
*         "12121212121212121212", that is 10 "12" strings.  */

data 2  "12" 061062 "1" "2"
061 062
(3) "12"
(3) 061062;


/*        Example 5.  The effective data address to be
*         word 1 of page 2 of the data area.  However, the cn
*         field of the descriptor specifies that the first
*         character of the word that is used is character
*         3.  Put some fill characters in the first
*         three characters.
*/

data 2    -do 4 "***"          /* Fill characters.  Not
                               *  referenced by the instruction. */
"abcd"    ;          /* The actual data string with which
                     *  the instruction works.  */
```

PAGE STATEMENT

The page statement is used to control page faults during the execution of the EIS instruction. The default case is that no page faults occur. The eis_tester program requires that you specify those pages on which faults are to be taken.

If you specify a page that is not actually used by the instruction (for example, the third page of a data area that has a one-character string), there is no harm. There is also no page fault.

All the pages used by an EIS instruction have been assigned names. For pages other than the two instruction area pages, the names can reference physically different pages. Their use by the EIS instruction is always the same.

The format of a page statement is:

```
  page {-control_args};
```

ARGUMENTS

page
: is the four-character statement name.

CONTROL ARGUMENTS

specify what pages are to have page faults and be chosen from the following:

-in1 -in2
: the two pages of the EIS instruction itself take a page fault.

-id1 -id2 -id3
: the pages used by descriptors referenced via indirect words take a page fault.

-d11 -d12 -d13
: the three pages of data referenced by descriptor 1 take a page fault.

-d21 -d22 -d23
: the three pages of data referenced by descriptor 2 take a page fault.

-d31 -d32
: the two pages of data referenced by descriptor 3 take a page fault.

-all
: specifies that all of the pages defined for this instruction take a page fault. If other control arguments are entered along with the -all control argument, then the pages specified do not have page faults.

RUNNING EIS_TESTER WITH OTHER USERS

If eis_tester is to be run while other users are on the system, it is not possible to positively guarantee that selected pages will not take a page fault. The "page -all;" statement causes eis_tester to flush all the pages of the etx, eti1, eti2, eti3, etd1, etd2, and etd3 segments out of main memory. Using "page -all -in2;" results in flushing all pages, touching page in2, and transferring control to etx. The touching of page in2 brings it into main memory. However, the overall system activity may be such that eis_tester loses control before reaching page in2, eis_tester and etx being displaced by pages for other users, control being returned to etx, execution continuing, and page in2 being no longer in memory. Then, when page in2 is needed, a page fault occurs. Therefore, a general guideline is: if eis_tester is run when other users are on the system, use the "page -all;" statement.

However, if eis_tester is to be run as the only user (nondaemon) process on the system and the "page" statement is not used, the pages should be in main memory when wanted. Some hardware problems may require running tests with and without page faults to isolate the problem. You should be aware that just because eis_tester attempts to avoid a page fault and the eis_tester output does not state that a page fault will occur does not necessarily mean that a page fault will not occur.

EXAMPLES OF PAGE STATEMENTS

```
/*        Example 1. */

page  -in2  -id3 -d32
-d12  -d12  -d11  -id1;


/*        Example 2. */

page  -all;


/*        Example 3.  Take faults on  ALL pages  EXCEPT
*         pages in2  and  id3  */

page       -in2      -all      -id3;  /* Notice order is not
                                      *  important.  */
```

EXAMPLES OF ACTUAL TEST SCRIPTS AND THEIR OUTPUT

```
/* mlr10 * * This test is the same as the test mlr3 except that * the
descriptors use AR, REG, and RL modification * and use indirect
descriptors. The indirect * words use both REG and AR modification. */

inst mlr -nt "10." -io 1 -mf1 rl 20
          ar /* This puts the data in etd1. */
          reg /* Use index register 1.. */
          idb /* This adds indirect descriptors. Descriptors
                    * go in segments eti1 and eti2. */

-mf2 idb
          rl 20
          reg
          ar;

desc 1 -cn 2;

desc 2 -cn 2;

data 1 -do -20 " " (5) "abcd" ;

data 2 -do -20 000 000 (5) "abcd" ; /* Fill for -cn 2
                                         * must be zeros. */

page -in1 -in2 -d22 -d21 -d11 -d12 -id1 -id2 -id3 -d32;
```

```
et mlr10 -nox

          /*The absence of any output from the
          *et mlr10 -nox input line means that the
          *script passes the validity checks that eis_tester performs.
*/

et

TEST 1 (mlr)

EIS instruction:  ( 262|3777 ) Ind Desc.
     000172100571
    - - - - -- - - - -
     400034000114 -> 100007200005 ( 327|100 )
     500043000115 -> 200016200006 ( 330|100 )

Pointer Registers:  ( 262|20 )
     pr0 - pr3 777777|1 332|1763 333|1753 777777|1
     pr4 - pr7 327|40 330|30 777777|1 777777|1

Index Registers:  ( 262|70 )
      X0 - X7 17777 4 10 777 4 5 17777 17777
         A 000000000024 Q 000000000024

Test Indicators:  ( 262|111 )
     000000000200

This test takes 8 page faults.
     in1 in2 id1 d11 d12 id2 d21 d22

data field 1 ( 332|1773 )
     000000141142 143144141142 143144141142 143144141142
     143144141142 143144

data field 2 ( 333|1773 )
     Result data field initialized to all zero bits.

test data ( 262|15776 )
     xxxxxxxxxxxx xxxxxxxxxxxx 040040141142 143144141142
     143144141142 143144141142 143144141142 143144xxxxxx
     xxxxxxxxxxxx xxxxxx

/* Test mvt instruction.  */

inst mvt -nt "3" -fc /* Char is octal 1.  */ -mf1 rl 3 ar reg idr -mf2
ar reg ida -mf3 reg ar;
```

```
desc 2 -ns 8;

data 1 -do -2 003 002 001;

data 2 -do -6 "321111" "11";

data 3 -do -1 "0" "123";


page -all -in2;


et

TEST 1 (mvt)

EIS instruction:  ( 262|4000 ) Ind Desc.
    - - - -- - - -
     001132160571
     051774000014 -> 100007000005 ( 262|52000 )
     500050000100 -> 200016000010 ( 330|100 )
     300025000113

Pointer Registers:  ( 262|20 )
     pr0 - pr3 777777|1 332|1767(18) 333|1756(18) 334|1747(27)
     pr4 - pr7 777777|1 330|30 777777|1 777777|1

Index Registers:  ( 262|70 )
     X0 -X7 17777 4 10 777 4 5 17777 17777
         A 000000000003 Q 000000017777

Test Indicators:  ( 262|111 )
     000000000200

This test takes 11 page faults.
     id1 d11 d12 d13 id2 d21 d22 d23 d31 d32 d33

data field 1 ( 332|1777(18) )
           003002 001

data field 2 ( 333|1776(18) )
     Result data field initialized to all zero bits.

data field 3 ( 334|1777(27) )
              060 061062063

test data ( 262|15776 )
     xxxxxxxxxxxx xxxxxxxxxxxx 063062061061 061061061061
     xxxxxxxxxxxx xxxxxxxxxxxx
```

STANDARD SCRIPTS

A standard set of scripts is provided that can be used with eis_tester. If CPU problems with the EIS instructions are suspected, these scripts should be run. The ets segment's standard location in the storage hierarchy is >tools>ets, but an installation can locate ets somewhere else.

EXAMPLES

```
/*  An example to illustrate the interaction between
 *  the "-do" and the "-cn" fields.
 */
inst mlr   -ir tn  -nt "-do and -cn interaction"
     -mf1 idb ar reg    -mf2 idb ar reg;
/*  The uppercase letters in the two data statements could have been
 *  typed in as  "ABCDEFGH"  and  "CDEF"  but were typed in as they are
 *  shown so that explanatory remarks could be placed on the adjacent
 *  lines.  The symbols used above and below the desc and data lines
 *  mean:
 *        P       The boundary of a page, and hence, also a word
 *                boundary.
 *        S----S  The operand string portion of the data field.
 *        W       A word boundary.
 *                                 W               P               W   */
desc 1 -cn 2 -ns 6;data 1 -do -5 "A" "B" "C" "D" "E" "F" "G" "H" "I" ;
/*                                          S-------------------S
 *  Each uppercase letter in the above data statement occupies 9 bits.
 *  Note that the data field for the first operand starts five 9-bit
 *  bytes to the left of a page boundary.  This is due to the "-do -5"
 *  field.  However, the operand string excludes the first two bytes
 *  of the data field, because of the "-cn 2" field.
 *
 *                                         P               W           */
desc 2 -cn 3 -ns 4;data 2 -do -1     000 000 000 "C" "D" "E" "F";
/*                                                 S-----------S
 *
 *  The data field for the second operand starts one 9-bit byte to the
 *  left of the page boundary due to the "-do -1" field.  The "-cn 3"
 *  field results in the operand skipping over the first three bytes of
 *  the data field.  Another way to specify the CDEF string to fall
 *  where it does would be to use these desc 2 and data 2 statements:
 *      desc 2 -ns 4;  data 2 -do 2 "CDEF";
 *  The "-cn 3" and  000 000 000  were used to show how to do it when
 *  the person writing the script wants to use the CN field in the
 *  second descriptor.     */
page -all;
```

```
COMMENT:  The output from running eis_tester with the above script is shown
COMMENT:  below.  Explanatory remarks have been inserted in the
COMMENT:  output, on the lines that start with COMMENT:

ET
TEST   1 (mlr)

Test Description:  -do and -cn interaction

Eis instruction:     ( 340|4000 )    Ind  Desc.
    - - - - -- - - - -
     000132100531
     400034000114        ->       100007400006    ( 341|100 )
     500043000115        ->       200016600004    ( 342|100 )

Pointer Registers:   ( 340|20 )
     pr0 - pr3     77777|1   344|1766(27)   345|1757(27)   77777|1

COMMENT:  The value in the parentheses following a word offset, which
COMMENT:  is in octal, is the bit offset, in decimal.

     pr4 - pr7     341|40   342|30   77777|1   77777|1

Index Registers:     ( 340|70  )
      X0 - X7    17777    4    10    17777    4    5    17777    17777
         A   000000017777       Q   000000017777

Test Indicators:     ( 340|111 )
     000000000300

This test takes 7 page faults.
     in2 id1 d11 d12 id2 d21 d22

COMMENT:  Page d21 is included here because data_field_2 crosses the
COMMENT:  boundary between the first and second pages of the etd2
COMMENT:  segment.  However, because of the "-cn 3" field, operand_2
COMMENT:  actually resides in only the second page.  Therefore, the
COMMENT:  first page is not be touched, and no page fault occurs
COMMENT:  for page d21.

data field 1          ( 344|1776(27) )
                101   102103104105  106107110111
COMMENT:                 S-----------------S
```

```
COMMENT:  It is true that data_field_1 begins in bit 27 of word 1776.
COMMENT:  However, because of the "-cn 2" field, operand_1 begins
COMMENT:  with bit 9 of word 1777.  The address development
COMMENT:  (in octal) for the start of operand_1 is:
COMMENT:
COMMENT:  ITEM   SEGMENT|WORD   9-BITBYTE
COMMENT:  ----   ------------   --------
COMMENT:  desc_1              7    2
COMMENT:  pr1           344|1766    3
COMMENT:  x1           04
COMMENT:       =        344|1775   11
COMMENT:  which is      344|1777    1    or    344|1777(9)
COMMENT:  If the same calculations are carried out for the second
COMMENT:  operand, it is seen that the data field starts in
COMMENT:  one page but the operand starts in the next page.
COMMENT:  Refer to the script line above that contains the
COMMENT:  "desc 2" and "data 2" statements and then examine the
COMMENT:  adjacent lines.

data field 2         ( 345|1777(27) )
     Result data field initialized to all zero bits.

test  data           ( 340|15776 )
     xxxxxxxxxxxx  xxxxxxxxxxxx  000000000103  104105106xxx
COMMENT:                                      S-----------S
COMMENT:  The xxxxxx represent fill supplied by eis_tester.  The nine
COMMENT:  leading octal zero digits are present because they were
COMMENT:  supplied in the "data 2" statement.
     xxxxxxxxxxxx  xxxxxxxxx


/*  An example to illustrate the interaction between
 *  the "-do" and "-cp" and "-bp" fields.     */
inst csl  -bo or  -nt "-do and -cp and -bp interaction"
     -mf1 idb ar reg    -mf2 idb ar reg;
desc 1 -cp 2 -bp 3 -ns 30;
/*  The symbols used above and below the data lines mean:
 *        P       The boundary of a page, and hence, also a word
 *                boundary.
 *        S----S  The operand string portion of the data field.
 *        W       A word boundary.
 *                           W               P             */

data 1 -do -7 123 456 701 020 203 040 123 456 765;
/*                     S-----------S
```

```
 *  Data_field_1 starts seven 9-bit bytes before the page boundary,
 *  due to the "-do -7" field.  The "-cp 2" field causes the processor
 *  to skip over the first two bytes (123 456 octal), so that
 *  operand_1 starts somewhere in the 701 octal byte.  The
 *  "-bp 3" field causes the processor to skip over the first three
 *  bits (7 octal) of the 701 byte, thereby starting at bit 30
 *  (bits numbered 0-35) of the next to last word of a page.
 */
desc 2 -cp 1 -bp 6 -ns 30;
/*          Pxxx           W               W          */
data 2 -do 1    432 103 030 405 050 765 432 101;
/*                    S-----------S
 *  The "Pxxx" above the "data 2" statement is intended to
 *  indicate that the page boundary is three octal digits (the xxx)
 *  i.e., nine bits, before the start of data_field_2, as
 *  specified by the "-do 1" field.  The "-cp 1" field specifies
 *  skipping over the first 9-bit byte, to the 103 octal byte.
 *  The "-bp 6" field specifies skipping the first six bits of
 *  that byte, to the octal 3, which begins in bit position 24
 *  (of 0-35) in the first word of a page.
 *                         W               W          */
data 3          432 103 132 425 354 765 432 101;
/*                    S-----------S
 *  The "data 3" statement is used because the  csl  instruction
 *  stores its result in the same bit locations from which the
 *  second operand was fetched.  No "-do", "-cp", or "-bp" fields
 *  are needed for the "data 3" statement because eis_tester
 *  associates the attributes of data_field_2 and operand_2 with
 *  the data supplied by the "data 3" statement.
 */
page -all;

ET
TEST   1 (csl)

Test Description:  -do and -cp and -bp interaction

Eis instruction:    ( 334|4000 )    Ind  Desc.
    - - - -- - - -
     007132060531
     400034000114        ->       100007430036   ( 335|100 )
     500043000115        ->       200016260036   ( 336|100 )
```

```
Pointer Registers:   ( 334|20 )
     pr0 - pr3     77777|1  340|1766(9)  341|1760(9)  77777|1
     pr4 - pr7     335|40  336|30  77777|1  77777|1

Index Registers:     ( 334|70  )
      X0 - X7   17777   44    110    17777   4   5   17777   17777
         A  000000017777        Q  000000017777

Test Indicators:     ( 334|111 )
     000000000200

This test takes 7 page faults.
     in2 id1 d11 d12 id2 d22 d32
COMMENT:  Page d12 is listed here because data_field_1 is in both
COMMENT:  pages 1 and 2.  However, operand_1 is in only page 1,
COMMENT:  so a page fault does not occur for page d12.

data field 1         ( 340|1776(9) )
        123456701  020203040123  456765
COMMENT:        S----------S

data field 2         ( 341|2000(9) )
        432103030  405050765432  101
COMMENT:      S----------S
COMMENT:  The address development for the start of operand_2 is
COMMENT:  shown below.  For x2, 110 octal = 72 decimal = 2 words
COMMENT:  and no bits.
COMMENT:                                 9-BIT BYTE          BIT IN
COMMENT:               SEGMENT|WORD      IN WORD             BYTE
COMMENT:  ITEM          (OCTAL)          (BINARY)            (BINARY)
COMMENT:
COMMENT:
COMMENT:  desc_2               16        01                  0110
COMMENT:  pr2            341|1760        01                  0000
COMMENT:  x2                    2        00                  0000
COMMENT:                 ------------------------------------------
COMMENT:       =         341|2000        10                  0110
COMMENT:
COMMENT:  which is segment 341, word 0 of the second page, 9-bit
COMMENT:  byte number 2 (numbering is 0-3), and bit number 6
COMMENT:  (of 0-8), i.e.,  341|2000(24).

test  data           ( 334|23776 )
     xxxxxxxxxxxx  xxxxxxxxxxxx  432103132425  354765432101
COMMENT:                                 S----------S
COMMENT:  The leading fill of 43210  and the trailing fill of
COMMENT:  765432101 were not affected by the execution of the
COMMENT:  instruction, proving that bits outside the operand
COMMENT:  strings did not enter into the instruction's execution.
     xxxxxxxxxxxx  xxxxxxxxxxxx
```

Name: enter_lss

SYNTAX AS A COMMAND

```
enter_lss path
```

FUNCTION

causes the command processor to compare each command to a supplied Limited Service Subsystem (LSS) control segment, which was created using the make_commands command. Any command not found in the control segment is refused. Those found are mapped into the command specified by the control segment.

ARGUMENTS

path
    is the pathname of the LSS control segment.

NOTES

The LSS control segment must be previously created with the make_commands command. The command line as shown in the example above is usually included in the project_start_up.ec. See the make_commands command for more details on the LSS facility and the use of the project_start_up exec_com.

EXAMPLES

To limit users on a project to a subsystem as specified in an LSS control segment named student_commands, place the following command line in their project_start_up.ec:

```
   enter_lss >udd>Students>student_commands
```

Name: excerpt_mst

SYNTAX AS A COMMAND

```
excerpt_mst reel_id {names}
```

FUNCTION

| selects given segments from a system tape (a BCE/Multics bootload tape).

ARGUMENTS

reel_id
: is the reel identification number of the tape to be extracted. The reel identification number, which is site dependent, can be up to 32 characters long. The reel_id can also include a density specification to indicate the density of the tape being excerpted, as in "060341,den=1600".

names
: are the names of the specific segments to be extracted. The star convention is allowed. If you supply no names, all of the segments on the tape are extracted. If a given segment has separate linkage and definitions on the tape and has been extracted, the separate linkage and definitions are extracted as well. Segments extracted are created in the current working directory. Bit counts are set from the SLT entry on the tape, as opposed to the actual length of the segment on the tape.

NOTES

A message is printed whenever a segment is extracted. A diagnostic is issued if you provide names that match no segments on the tape.

Name: exercise_disk

SYNTAX AS A COMMAND

```
exercise_disk disk_type volume_id {-control_args}
```

FUNCTION

exercises a disk drive. Maximal arm motion occurs all over the disk, and data is written and read back later for checking at each point. This activity can be used to make unstable drives fail repeatedly.

ARGUMENTS

disk_type
: a valid Multics disk device type (e.g., d451, d500, and d501).

volume_id
: the label of the disk pack on which the test is to be run.

CONTROL ARGUMENTS

-write_read, -wr
: writes a known pattern over the entire disk pack, and then reads this information back for checking purposes. This is the default.

-write, -w
: writes a known pattern over the entire disk pack. The default is -write_read.

-read, -r
: reads back the information on the disk pack, for checking purposes. The default is -write_read.

-device STR, -dv STR
: specifies the device on which you want to run the test, where STR can be dska_02, dskb_13, etc.). Either this control argument or the assign_resource (ar) command must be used to attach an I/O disk.

-no_data_compare, -ndc
: makes no data compare on the read pass; only errors detected by the hardware are reported. This enables testing of a disk pack without knowing what data is recorded on it. The default is to compare the data with a known pattern.

-random
: the test performs random rather than sequential seeks; the test takes several hours. This is the default.

-sequential, -sq
: the test runs sequentially, writing and reading from sector 0.

-alternate_track, -altrk
: removes the alternate track area of the disk from the test parameters. The default is to use the entire pack.

-from M, -fm M
: sets the lower limit of the range of addresses to be tested to M, where M is a decimal integer specifying a valid cylinder number for the device to be tested.

-to N
: sets the high limit of the range of addresses to be tested to N, where N is a decimal integer specifying a valid cylinder number for the device to be tested.

-system

NOTES

The exercise_disk command requests the mounting of a scratch pack.

The assign_resource command must be used in conjunction with this command to exercise a given drive. Such drives must be configured as user I/O drives (nonstorage system) via the udsk config card (described in the *Multics System Maintenance Procedures* manual, Order No. AM81) or by the use of the set_drive_usage (sdu) initializer command.

When the -from or -to control arguments are used, testing is confined to the range of addresses specified. The seek pattern used in this mode is from inner cylinder to outer cylinder, with M incrementing to N or the maximum address of the device, and N decrementing to M or cylinder zero. When M reaches its inner limit, the pattern is repeated. Testing continues until you stop it, by hitting the break key and then typing the release (rl) command.

Name: file_system_meters, fsm

SYNTAX AS A COMMAND

```
fsm {-control_args}
```

FUNCTION

used to meter certain storage system variables and functions.

CONTROL ARGUMENTS

-ast
  prints certain meters about active segment table (AST) usage.

-brief, -bf
  generates a shortened report. Those meters not printed if -brief is specified are indicated by a plus (+) in "Notes" below.

-page, -pg
  prints certain meters about paging.

-report_reset, -rr
  generates a report and then performs the reset operation. The report can be shortened by using the -brief control argument.

-reset, -rs
  resets the metering interval for the invoking process so that the interval begins at the last call with -reset specified. If -reset has never been given in a process, it is equivalent to having been specified at system initialization time.

ACCESS REQUIRED

This command requires access to phcs_ or metering_gate_.

NOTES

If the file_system_meters command is given with no control arguments, it prints a full report.

The following meters, which reflect the activity of the AST lists, are printed if the -ast control argument is specified. The two columns printed by this command contain the number of occurrences of the specified item and the average time between occurrences.

Activations
    is the number of segment activations.

    segfault
        is the number of activations caused by segment faults; also expressed as a percentage of all activations.

    makeknown
        is the number of activations not caused directly by segment faults, but resulting from explicit calls from the makeknown_ routine: also expressed as a percentage of all activations.

    backup
        is the number of activations resulting from calls to activate$backup_activate; also expressed as a percentage of all activations.

    directories
        is the number of directories activated; also expressed as a percentage of all activations.

Deactivations
    is the number of segment deactivations.

Demand deactivate

    attempts
        is the number of deactivations explicitly requested by users.

    successes
        is the number of demand deactivations which succeeded; also expressed as a percentage of attempts and as a percentage of all deactivations.

Seg Faults

fault
: is the number of segment faults.

call
: is the number of calls to the segment fault handler to activate a segment without taking a segment fault; also expressed as a percentage of segment faults.

activations
: is the number of segment faults that resulted in an activation; also expressed as a percentage of segment faults.

Bound Faults
: is the number of bound faults.

+ Setfaults
: is the number of setfaults performed during segment deactivation and during the handling of bound faults. Setfaults are segment faults forced when dynamic segment attributes are changed (e.g., access to a segment is revoked by another process).

access
: is the number of setfaults performed because the access was changed on a segment; also expressed as a percentage of all setfaults.

+ ASTE Trickle
: is the number of times the VTOCE was updated from the ASTE because the information in the VTOCE was noticed to have changed during ASTE allocation. ASTE Trickle is a hedge against system crashes, as it updates ASTEs into the VTOCEs periodically, rather than on demand.

+ Steps
: is the number of steps taken through the AST lists searching for a free, usable AST entry.

+ Skips
: is the number of times an entry was skipped; also expressed as a percentage of Steps.

+ ehs
: is the number of times an entry was skipped in the search for a free, usable entry because the entry-hold-switch was on. The entry-hold-switch is set for certain segments that cannot be deactivated. Also expressed as a percentage of Skips.

+ mem
: is the number of times an entry was skipped because it had pages in memory; also expressed as a percentage of Skips.

+ init
    is the number of times an entry was skipped to give it a grace lap after all of its pages were removed from core; also expressed as a percentage of Skips.

+ Searches
    is the number of full AST searches required because no entry was readily available.

+ Avg. Cost
    is the average "cost" in I/Os of deactivations arising from full searches.

Cleanups
    is the number of calls to cleanup. The percentage of real time spent in cleanup is also given.

Force writes
    is the number of calls to force_write. The three following meters relating to force_writes are printed only if any force_writes occurred.

    without pwrites
        is the number of times force_write wrote no pages.

    pages written
        is the number of pages written by force_write.

    force updatev
        is the number of calls to update_vtoce resulting from force_writes.

Lock AST
    is the number of lockings of the AST.

The following meters provide information about AST lock contention.

AST locked
    is the average real time during which the AST lock is held locked and the percentage of the metering interval during which the AST was locked. This percentage cannot exceed 100%, and the closer the 100% figure is approached, the more AST lock contention becomes the limiting function in system throughput.

AST lock waiting
    is the average real time delay between an attempt to lock the AST and successful locking of the AST. The total real time spent by all processes waiting for the AST lock, expressed as a percentage of the metering interval, is also given. This number may exceed 100% if, on the average, more than one process was waiting for the AST lock.

The following items represent a table indexed by page table size and they show the activity and use of the four AST lists.

AST Sizes
: indicates the page table sizes being used by the system (constant).

Number
: is the number of entries of the specified size.

Need
: is the number of entries of the specified size that were needed.

Steps
: is the number of steps taken while scanning the specified list.

Ave Steps
: is the average number of steps taken in the specified list to find a usable entry in the list.

Lap Time (sec)
: is the average time for the replacement algorithm to make one pass over the list.

The following meters are printed if the -page control argument is specified. The two columns printed by this command contain the number of occurrences of the specified item and the average time between occurrences.

Needc
: is the number of times a frame of main memory was needed (for page faults, process loadings, etc.).

Ceiling
: is the number of times too many write requests were queued at once. Not printed if zero.

Claim runs
: is the number of times the page removal algorithm could not queue an additional write until a previous write was completed. If the average time between claim runs is less than .010 minutes, then an I/O bottleneck probably exists in the system. If the value of claim runs is zero, then the value is not printed.

Ring 0 faults
: is the percentage of page faults that occurred while executing in ring 0.

PDIR faults
: is the percentage of page faults that occur on pages of segments in process directories.

Level 2 faults
: is the percentage of page faults on pages of segments in directories directly off the root. This is a measure of the activity of the system libraries. hhw

DIR faults
: is the percentage of page faults on directory pages.

New Pages
: is the percentage of page faults that resulted in the creation of pages. This happens because the page faulted has never been referenced or logically contains all zeros.

Volmap_seg
: is the number of page faults taken on free storage maps; also expressed as the average time between faults.

Zero pages
: is the number of page writes that were avoided because the page to be written contained all zeros. In general, pages of zeros are not written to disk but are converted to null addresses (an indication in the file map for the segment that the page is logically zero). Also expressed as the average time between zero pages.

Segment State Change
: is a measure of suspected covert channel activity.

  audited
  : is the number of times all processes performed suspected covert channel segment state changes at a rate execeeding the system-defined limit.

  delayed
  : is the number of times all processes had to be delayed in order to keep the rate of suspected covert channel segment state changes within the system-defined limit.

  avg.
  : is the average time processes were delayed.

Laps
: is the number of times the used pointer has gone around the main memory used list in the search for a usable block of main memory.

+ Steps
: is the number of steps taken around the main memory used list. A step consists in moving the used pointer to the next entry on the list.

+ Skip
: is the number of times a page was skipped; also expressed as a percentage of Steps.

+ wired
: is the number of times a page was skipped while searching the main memory used list because it was wired down; also expressed as a percentage of Skip.

+ used
: is the number of times a page was skipped because it was used in the last lap; also expressed as a percentage of Skip.

+ mod
: is the number of times a page was skipped because it had been modified; also expressed as a percentage of Skip.

+ fc pin
: is the number of times a page was skipped by find_core because it was pinned; also expressed as a percentage of Skip.

+ cl pin
: is the number of times a page was skipped by claim_mode_core because it was pinned; also expressed as a percentage of Skip.

pages
: is the number of pages available in the system. This is the total main memory minus the permanently wired down supervisor.

wired
: is the number of pages temporarily wired down. This includes descriptor segments and process data segments (PDS) for loaded processes.

Average steps
: is the average number of steps taken around the main memory used list to find a usable frame of main memory.

EXAMPLES

The following is an example of the information printed when the file_system_meters command is invoked with no control arguments. (Appendix A contains a representation of the configuration deck used to create the system from which the metering samples were taken.)

```
! file_system_meters

  Total metering time          0:15:23

                         #          ATB

Activations            2227       0.415 sec.
  segfault             1786       0.517 sec.  80.198% of all
  makeknown             184       5.021 sec.   8.262% of all
  backup                257       3.595 sec.  11.540% of all
  directories           242       3.818 sec.  10.867% of all
  synch                   2     461.961 sec.   0.090% of all
Deactivations          2225       0.415 sec.
Demand deactivate
  attempts                6     153.987 sec.
Seg Faults            10571       0.087 sec.
  fault                8972       0.103 sec.  84.874% of Seg Faults
  call                 1599       0.578 sec.  15.126% of Seg Faults
  activations          1786       0.517 sec.  16.895% of Seg Faults
Bound Faults            434       2.129 sec.
Setfaults             10325      89.484 msec.
  access                 32      28.873 sec.   0.310% of setfaults
ASTE Trickle            212       4.358 sec.
Steps                  9283      99.528 msec.
Skips                  6568       0.141 sec.  70.753% of Steps
  ehs                   756       1.222 sec.  11.510% of Skips
  mem                  1930       0.479 sec.  29.385% of Skips
  init                 3882       0.238 sec.  59.105% of Skips
Searches                  0       0.000 sec.
Cleanups               2279       0.405 sec.   0.7 % of real time
Force writes           3631       0.254 sec.
  pages written        6101       0.151 sec.
  force updatev          17      54.348 sec.
Lock AST              44010       0.021 sec.

                       AVE/lock       %
```

```
AST locked          5.871 msec.  28.0
AST lock waiting    9.579 msec.  45.6

AST Sizes              4        16        64       256
Number              3500      1500       750       365
Need                1935       565       179        36
Steps               6259      2122       793       109
Ave Steps            3.2       3.8       4.4       3.0
Lap Time(sec)      516.7     653.1     873.8    3093.9

                     #           ATB

Needc          119620       7.724 msec.
Ring 0 faults              15.446 %
PDIR faults                50.083 %
Level 2 faults             11.880 %
DIR faults                  9.162 %
New Pages                  12.417 %
Volmap_seg          0       0.000 msec.
Zero pages       1264     730.951 msec.
Seg state chg.
   delayed          2     461.961 sec.    0.2 sec. avg. delay
   audited          4     230.981 sec.
Laps               74      12.485 sec.
Steps          654084       1.413 msec.
Skip           579739       1.594 msec.  88.634% of Steps
   wired         9589      96.352 msec.   1.654% of Skip
   used        190026       4.862 msec.  32.778% of Skip
   mod         269202       3.432 msec.  46.435% of Skip
   fc pin       65647      14.074 msec.  11.324% of Skip
   cl pin       45275      20.407 msec.   7.810% of Skip

8826 pages, 203 wired.
Average steps       5.468
```

Name: fim_meters

SYNTAX AS A COMMAND

```
fim_meters {fault_name} {-control_args}
```

FUNCTION

interprets and prints per-system metering information on central processor faults.

ARGUMENTS

fault_name
: is the name of a single fault type (valid fault types are listed under "Notes" below). Only the information for that fault type is printed. If fault_name is not specified, then information for all fault types is printed. No control argument can be given if a fault_name is specified.

CONTROL ARGUMENTS

-cpu {tag1...tagN}
: displays fault counts for those processors specified by tagi. Tag may be one of the letters a through h or A through H. If tag is omitted, all processors are selected.

-long, -lg
: prints information on software-interpreted subordinate faults with each associated hardware fault.

-report_reset, -rr
: generates a full report and then performs a reset operation.

-reset, -rs
: resets the metering interval for the invoking process so that the interval begins at the last call with -reset specified. If -reset has never been given in a process, it is equivalent to having been specified at system initialization time. The metering interval is reset for ALL processors.

-sort {STR}
: sorts the output as specified by STR, which can be either "count" or "number". If STR is not specified, the output is sorted by count. If this control argument is not specified, the output is sorted by hardware fault number.

-total, -tt
: displays total fault counts for all processors. This is the default.

ACCESS REQUIRED

This command requires access to phcs_ or metering_gate_.

NOTES

If both -long and -sort are used, subordinate faults are sorted within associated hardware fault.

The following is a brief description of the variables printed by the fim_meters command.

type
: is the name of the hardware fault.

count
: is the total number of times the fault occurred on any central processor configured.

The following are the valid names of hardware faults. These names are printed as "type" and can be used as the fault_name argument. Full descriptions of these faults can be found in the *Multics Processor Manual*, Order No. AL39.

```
access_violation, acv
command, cmd
connect, con
derail, drl
directed_fault_2, df2
directed_fault_3, df3
divide_check, div
execute, exf
fault_tag_1, ft1
fault_tag_3, ft3
illegal_procedure, ipr
linkage_fault, ft2
lockup, luf
mme1, mme
mme2
mme3
mme4
op_not_complete, onc
overflow, ovf
page_fault, df1
parity, par
segment_fault, df0
shutdown, sdf
startup, suf
store, str
timer_ranout, tro
trouble, trb
```

Subordinate faults are recursive calls to the main fault processor that can be interpreted as subcases of hardware faults. For example, the hardware generated "command" fault may be interpreted as an isot_fault, a lot_fault, or a command_fault, depending on other conditions at the time of the fault. A complete description of subordinate faults would require a description of Multics fault processing, and is beyond the scope of this manual.

EXAMPLES

The following is an example of the information printed when the fim_meters command is invoked with no control arguments. (Appendix A contains a representation of the configuration deck used to create the system from which the metering sample was taken.)

```
fim_meters

Total Metering Time:           56:14:39

Fault Type                    Total Fault Count
mme1                                      23149
fault_tag_1                                  14
timer_runout                            4026546
command                                     302
derail                                      272
connect                                32565031
illegal_procedure                            29
overflow                                    246
divide_check                                651
segment_fault                            591202
page_fault                              5919620
access_violation                         452383
mme2                                         12
linkage_fault                           1380667
```

Name: fix_quota_used

SYNTAX AS A COMMAND

```
fix_quota_used path
```

FUNCTION

repairs inconsistencies in storage system quota used for a directory.

ARGUMENTS

path
    is the pathname of the directory for which quota is to be made consistent.

ACCESS REQUIRED

Access to the hphcs_ gate is required.

NOTES

The normal use of this command is from the fix_quota_used.ec exec_com, or by the "x repair" operator command. When a quota (segment quota or directory quota) is found inconsistent and corrected, a message is printed. If the correction causes a directory to have greater quota used than allocated, another message is printed.

Name: flush

SYNTAX AS A COMMAND

```
flush {-control_arg}
```

FUNCTION

causes excessive paging activity in order to time the system. It is not to be used casually as it impairs service to all users of the system.

CONTROL ARGUMENTS

-temp_dir path, -td path
: specifies that temporary segments used for flushing main memory are to be created in the directory identified by path (the default is to create them in the process directory).

NOTES

In order for all pages in main memory to be flushed, the directory used for temporary segments must have sufficient quota for as many pages as there are in main memory.

Name: fnp_data_summary

SYNTAX AS A COMMAND

```
fnp_data_summary {fnp_names} {-control_args}
```

FUNCTION

reports error statistics recorded in the syserr log by the poll_fnp command. The statistics reported include parity errors for all channels and various counters for synchronous channels whose interpretation depends on the line type of the channel. Only nonzero statistics are reported.

ARGUMENTS

fnp_names
: are the names of FNPs for which statistics are to be reported. If no fnp_names are specified, statistics are reported for all FNPs for which any nonzero statistics are available.

CONTROL ARGUMENTS

-expand
displays in expanded form every entry in the syserr log containing statistics for the specified FNP(s).

-extend, -ext
appends the output of the command to the end of the output_file if it already exists. This control argument can be specified only if -output_file (below) is specified.

-from DATE_TIME, -fm DATE_TIME
starts scanning the syserr log from the time specified by DATE_TIME, which must be a character string acceptable to convert_date_to_binary_ (described in the *Multics Subroutines and I/O Modules* manual, Order No. AG93). The default is to start at the beginning of the log.

-for TIME
restricts the scan of the syserr log to an interval of length TIME, where TIME is a character string representation of a time interval in a form acceptable to convert_date_to_binary_ (described in the *Multics Subroutines and I/O Modules* manual, Order No. AG93). This control argument is incompatible with -to (described below). The default is to continue the scan up to the end of the log.

-output_file {PATH}, -of {PATH}
directs output to a file. If PATH is specified, it is the pathname of the output file; otherwise, output is sent to fnp_data_summary.output in the current working directory. If -output_file is not specified, the default is to direct output to your terminal.

-to DATE_TIME
ends the scan of the syserr log at the time specified by DATE_TIME, which must be a character string acceptable to convert_date_to_binary_. This control argument is incompatible with -for (above). The default is to continue the scan to the end of the log.

ACCESS REQUIRED

Read permission is required on the log segments themselves and status permission is required on their containing directories.

Name: fnp_throughput

SYNTAX AS A COMMAND

```
fnp_throughput {fnp_id} {-control_arg}
```

FUNCTION

reports character throughput for an FNP or all FNPs and optionally allows the resetting of the metering interval.

ARGUMENTS

fnp_id
: is the name of an FNP or "*", which means all currently running FNPs. This argument must be specified unless the -reset control argument is specified, in which case fnp_id must not be specified.

CONTROL ARGUMENTS

may be either, but not both, of the following. If neither is specified, information is printed and the metering interval is not reset.

-report_reset, rr
: causes statistics to be printed and the metering interval to be reset. If this control argument is specified, fnp_id must be specified.

-reset, -rs
: causes the metering interval to be reset without printing any statistics. If this control argument is specified, fnp_id must be omitted.

NOTES

The start of the metering interval in effect is measured from the time the FNP was last booted, or from the time the interval was last reset, whichever was the most recent event.

The reset action of the -report_reset control argument applies to the metering interval for all FNPs, even though the command invocation is specific to the statistics for a particular FNP.

EXAMPLES

To report on throughput statistics for FNP "a", while resetting the metering interval, specify:

```
   fnp_throughput a -report_reset
```

The output would appear as follows:

```
                                 input              output
    Characters transmitted    71,966,400          94,934,400
    Characters per second          4,700               6,200
```

Name: format_disk_pack, fdp

SYNTAX AS A COMMAND

```
fdp operation -control_args
```

FUNCTION

provides a means of formatting 451 disk packs online under Multics. This command cannot be used to format fixed media devices (MSU0500 and MSU0501 disks). An I/O disk drive must be available in the configuration to allow this command to operate.

In general, all disk packs should be formatted during Multics operation by using the online T&D tool, MTR, under TOLTS and MOLTS. Procedures for formatting both MSU0451 and MSU0500/MSU0501 disk packs with MTR are described in the *Multics Online Test and Diagnostics Reference Manual*, Order No. AU77.

ARGUMENTS

operation
    is one of the following operations to be performed on the specified disk pack:

read_pack
    each track header of the disk pack is read and data is accumulated on all tracks that are formatted "defective." After all disk pack headers are read, a summary of the defective tracks is displayed on the user's terminal.

format_pack
    each track header is read and the track condition is interrogated. A format_track command is then issued, using the information from the previous track condition in the following manner:

If the -nodef control argument is not specified, all the tracks are reformatted according to the current format condition.

If the -nodef control argument is specified, all tracks are reformatted as "good" regardless of the previous track condition.

Data on the original track condition (as determined by reading the track header) is retained and displayed, after the entire disk pack has been formatted. Tracks are formatted as defective only if they are marked as defective in the track header information of each track.

read_track
: a request loop is entered (see "Notes" below) and the user is asked to type in the cylinder and head address of a particular track. A read track header is then issued to the requested track and the track condition is displayed on the user's terminal. The program remains in the request loop until the user issues the quit request.

format_track
: a request loop is entered (see "Notes" below) and the user is asked to enter the track address and the desired track condition; a format track is then issued to the requested track address, formatting the track with the user specified track condition. If the requested track condition is defective with alternate assigned, the program searches for the next available alternate track, formats this track as alternate and displays the alternate track address to the user. The program remains in the request loop until the user issues the quit request.

read_label
: the format label record is read and its contents displayed in an interpretive format.

CONTROL ARGUMENTS

-hbypass
: (header bypass) sets the format track control bits so that no check of the track header matching the seek address is made by the hardware. Indiscriminate use of this control argument is not advised. It is recommended that the -hbypass control argument be used only when formatting packs that have been degaussed or otherwise erased.

-hold
: allows the user to go into an option loop after the current operation is complete so that he may select another operation without releasing the disk pack. The user may exit from the option loop by issuing the quit request instead of the operations described above.

-model ID
: specifies that ID is the disk pack model (e.g., m400, m451). Default is the m400 model.

-nodef
: forces all tracks to be formatted as "good" regardless of the previous track condition. This control argument is valid only for the format_pack operation.

-system
: is required for the mounting and formatting of Multics storage system volumes. In the case of a new pack, if the system can't read the pack's label, it will assume that the pack is a storage system pack. Thus, the -system control argument should always be used with a new pack, to avoid any possibility of problems. In fact, the only time -system should not be used is when the volume has been previously formatted and is not a storage system volume. Access to the privileged gate rcp_sys_ is needed for use of this control argument.

-volume ID
: specifies that ID is the pack label name. This control argument is always required, so the system can tell you the name of the disk to be mounted.

NOTES

Input to the read_track request loop is of the form:

```
ccc,hn
```

or:

```
quit
```

where:

ccc
: is the cylinder number.

hh
: is the head number.

quit
: is the exit from the request loop.

The track's condition is then displayed on the user's terminal.

Input to the format_track loop is of the form:

```
ccc,hh,key
```

or:

```
quit
```

where:

ccc
: is the cylinder number.

hh
: is the head number.

key
: is any of the following:

  good
  : formats the track as good.

  defa
  : formats the track as defective and assigns an alternate.

  def
  : formats the track as defective.

quit
: is the exit from the request loop.

The track's new condition is displayed on your terminal.

DEFECTIVE DISK TRACK HANDLING

The last three cylinders of a disk pack can be reserved for use as an alternate track area. The tracks of these cylinders may be pointed to by the track headers of defective tracks. The disk controller automatically references an alternate track when it encounters a reference to a track marked as defective with an assigned alternate. Such a reference requires at least two additional rotations of the disk.

This command allows you to format a disk pack under Multics operation, and to redirect defective tracks to the alternate track area of a disk pack.

You can't normally reference directly a disk track defined as alternate. You can usually reference it by making a reference to the defective track redirected to that alternate. Therefore, when a disk pack contains alternate tracks, you must carry out certain procedures for successful BCE/Multics operation. These are described in the * following paragraphs.

BCE/MULTICS OPERATION WITH ALTERNATE TRACKS

If alternate tracks are to be defined on a new disk pack, you must define a special partition when that pack is initialized as a Multics storage system volume. You use the init_vol command for this purpose. Always give -special with init_vol if the pack being initialized contains alternate tracks. The first partition defined must be the partition containing the alternate track area, which you accomplish by typing

```
    part alt high 141
```

which defines the ALT partition to ensure that Multics makes no other use of the disk pack area reserved for alternate tracks.

If you define no ALT partition, Multics may attempt, at some future time, to utilize the ALT track area of the disk as part of the storage system hierarchy, with disastrous results.

When a storage system volume is copied and rebuilt onto a new volume, you must define the ALT partition as above in the invocation of the rebuild_disk command.

*

Name: generate_mst, gm

SYNTAX AS A COMMAND

```
gm path reel_id {-control_args}
```

FUNCTION

generates a BCE/Multics system tape that can later be "bootloaded" by BCE as the first step in bringing up a Multics system.

ARGUMENTS

path
: is the pathname of the header segment without the header suffix.

reel_id
: is the reel identification number of the tape from which information is to be copied. This reel identification number, which is site dependent, can be up to 32 characters long, and can include a density specification to indicate the density of the tape being written, as in "060341,den=1600".

CONTROL ARGUMENTS

-directory, -dr
: puts a search rule segment in your working directory. The name of the search rule segment is path.search, where path is the entryname portion of the pathname given.

-file, -fl
: directs output to a file in the storage system rather than to a tape. The file name (which can specify a multisegment file) has the same name as the reel_id argument.

-hold
: does not detach the tape when generation is completed. You can then perform a checker run on the same tape without remounting the reel.

-notape
: does not generate a tape. You can use it to check the consistency of the header segment and produce an output listing without actually generating a tape.

-sys_id STR, -sysid STR
: sets the system identifier to STR, which can be up to eight characters long. If you omit it, the first eight characters of the entryname portion of the pathname given are used by default.

-vers_id STR, -versid STR
: sets the version identifier to STR, which can be up to eight characters long. If you omit it, the first eight characters of the entryname portion of the pathname given are used by default.

FORMAT OF A SYSTEM TAPE HEADER

A system tape header is an ASCII file (in free format) consisting of keywords followed by optional control arguments. You can place comments anywhere in the header, except within a keyword name or control argument, and can separate them by "/*" and "*/".

There are two levels of keywords: major and minor. The fabricate, first_name, name, object, and text keywords are initial keywords and indicate the start of a description of control arguments for a single segment to be placed on the system tape. The linkage keyword is only valid if found in a segment description list (SDL). The end keyword indicates the end of an SDL. The collection keyword, which cannot occur in an SDL, instructs the generator to write a collection mark on the system tape. The fini keyword, which cannot occur within an SDL, instructs the generator to close out the tape by writing an EOF and dismounting it.

The syntax of the header consists of some SDLs, occasionally separated by collection keywords and ending with a fini keyword.

Keywords having no arguments are followed immediately by semicolons; those having arguments are followed immediately by a colon, which is followed by arguments, separated by commas. The arguments end with a semicolon.

LIST OF MAJOR KEYWORDS

add_segnames
: adds the segnames defined in an object segment to the list of names for that segment, as if they had appeared in the list following an "object" or a "name" statement. All names that appear as segname definitions in the object segment are added to the list of names for this segment. You can only use this keyword in the SDL for a bound object segment immediately after the keyword that begins the SDL. You can usually use it to replace the list of names associated with a bound segment.

boot_program
: begins the definition of a segment that is placed in the bootload portion of the system tape label. The bootload_program portion of the system tape label is executed when the initialize/bootload sequence is executed by the IOM switch or OC command sequence. Only the text section of the program is placed on the tape, and it must be less than 1500 (octal) words long; if shorter, it is padded to 1500 words with NOP instructions. Put this keyword first in the header file. It is incompatible with the first_name keyword.

collection
: writes a collection mark indicated by N on the tape containing the collection number that follows the collection keyword. Put this keyword between segments, not in a segment definition.

data names
: begins a list of names associated with the segment. This keyword places the complete named segment on the tape, preceded by a preface area containing all the information specified in the SDL. The data keyword is used only for segments that are not Multics standard object segments, such as ASCII files. The data and linkage keywords are incompatible.

delete_name names
: removes extra names from the list of names for the current segment that were added with the add_segnames statement but that should not appear on the segment. Like add_segnames, you can usually use it to replace the list of names associated with a bound segment. It must appear after add_segnames in an SDL.

end
: specifies the end of a segment definition. This keyword must conclude every use of an object, name, first_name, fabricate, or text keyword.

fabricate names
: makes an all-zero segment and places it on the tape; names is a list of names associated with the segment. The attributes for the segment are derived from the SDL. The fabricate and linkage keywords are incompatible.

fini
: specifies the end of a system tape header. Any keywords appearing in the header after the first fini keyword are ignored.

first_name name
: indicates that the named segment associated with this SDL is the first segment on the tape and is specially processed; i.e., the first 32 decimal words of the segment are overwritten with tape header information when the tape is bootloaded.

linkage
: places the linkage and definitions sections of an object segment on the tape, following the object segment itself (if you used the object keyword to define it) or the text section (if you used the name or text keywords). The linkage keyword must appear in an object definition between the object, text, or name keyword for the segment and the end keyword. Any minor keywords following a linkage keyword (e.g., wired) are applied to the linkage section rather than to the text section: you can use this to direct the linkage section into a different supervisor-combined linkage segment than would be used by default. You must supply the linkage keyword to include definitions on the tape and copy them into the supervisor definitions segment, even if the segment has no linkage section. This is often true for object segments created with create_data_segment. If an object segment is used by the supervisor, place its definitions sections on the tape by specifying the linkage keyword, even if the segment is started with the object statement, so that the definitions section is included along with the text section.

name names
: places the named segment on tape preceded by a preface area for the segment containing all the information specified in the SDL. If the linkage keyword is found in the SDL, the generator splits apart the object segment named and places only the text on the tape. Then the linkage section by itself (preceded by a preface area for the linkage section) follows the text and definitions section (preceded by its preface) on the tape. Otherwise the entire object segment is placed on the tape. Use this keyword for nonobject segments. For a BCE/Multics system tape, the names specified in the header for a segment are the only ones by which you can reference the segment. Extra names on the segment itself are ignored. When adding a new program to an existing bound segment, update the system tape header, as well as the bindfile, before adding the name of the new program to the list of names for the bound segment.

object names
: behaves exactly as the name keyword except that the entire object segment is placed on tape rather than just the text section. It is also followed by the (redundant) linkage and definition sections if you use the linkage keyword.

text names
: places the text section alone on tape. Use this keyword if you want only the text part of an object segment.

LIST OF MINOR KEYWORDS

abs_seg
: is either yes or no. Indicates whether or not to suppress creation of a segment when current length/maximum length is not zero.

access
: is the SDW access mode for the segment in the supervisor's address space. The list can contain any combination of read, write, execute, and privileged.

acl
: is an ACL entry placed in the branch of the segment. Only segments placed in the hierarchy (via "path_name") can have ACL entries. The format of the acl arguments is "<access> Person_id.Project_id.tag", where Person.Project.tag must include all three components.

bit_count
: is a number specifying a bit count to be associated with the segment.

cache
: is either yes or no. It indicates whether or not to override the default encacheability of the segment. If you don't give this keyword, the following defaults are used: if you specify the per_process keyword as yes, then cache is yes; if you specify the init_seg or temp_seg keywords as yes or specify write access under the access keyword, then cache is no; otherwise cache is yes.

cur_length
: is a number specifying the number of words to be allocated to the segment (for unpaged segments and segments loaded in collection1). If this segment is a collection1 segment that is to be paged, cur_length is its length while unpaged.

delete_at_shutdown
: is either yes or no. It indicates whether or not to return the pages of the segment to the appropriate free pool at shutdown time.

init_seg
: is either yes or no. It indicates whether or not to delete the segment at the end of initialization.

link_sect_wired
: is either yes or no. It indicates whether or not the linkage for the segment is to be combined in the supervisor's wired linkage section even though the segment itself might not be wired.

max_length
: is a number specifying the number of pages to be allocated to this segment (for paged segments). The greater of max_length and cur_length (converted to pages) determines the size of the page table and the segment bound.

paged
: is either yes or no. It indicates whether or not the segment is to be constructed as a paged segment.

path_name
: specifies that the segment is to be placed in the hierarchy. The value of the argument is the pathname of the directory in which the segment is placed. This keyword is required for segments in collection3. If you choose this keyword, all names listed for the segment are added to the version in the hierarchy. If an object segment is to be placed in the hierarchy, define it with the object keyword, so that the whole segment appears rather than just the text section.

per_process
: either yes or no. Indicates whether or not to suppress copying of the SDW for this segment at process-creation time.

ringbrack
: is 1, 2, or 3 numbers, separated by commas, to be interpreted as the ring brackets to be placed in the branch for segments that are to go in the hierarchy. Default ring brackets are (0,0,0). Rules for assigning ring brackets are described in the set_ring_brackets command.

sys_id
: specifies an external name in this segment identifying a location that is set to the eight-character system identifier (see -sys_id). This normally appears only for BCE/Multics system tapes and identifies the symbol active_all_rings_data$system_id.

temp_seg
: either yes or no. Indicates whether or not to delete the segment at the end of the collection in which it was loaded.

vers_id
: specifies an external name in this segment identifying a location that is set to the eight-character version identifier (see -vers_id). This normally appears only for BCE/Multics system tapes and identifies the symbol active_all_rings_data$version_id.

wired
: either yes or no. Indicates whether or not the pages of the segment are to be wired.

OPERATION

The generate_mst command works by reading the header segments and performing one of the following:

1. If the word found is an initial keyword, the information about the specified segment (i.e., all information up to the next end keyword) is gathered together and written on the system tape followed by the data for the segment itself.

2. If the keyword is collection, a special mark is written on the tape indicating the end of the specified collection.

3. If the keyword is fini, the tape is closed out and dismounted.

For segments that are placed on tape (i.e., segments specified with an initial keyword), the first argument to the initial keyword is the name used when searching for the actual segment to be placed on tape. All subsequent arguments are treated as secondary names, and although they are placed on the tape in the preface area for each segment they are not used by the generator.

HARDCORE PROFILING

If hardcore programs are compiled with the -profile or -long_profile options, it is possible to profile the behavior of the supervisor (see the -hardcore control argument to the profile command).

There are several common pitfalls encountered in hardcore profiling. The size of the supervisor linkage segments must be increased to contain the additional static data generated by the profiling code. You can determine the required sizes from the loading summary information following collection two in the output file from check_mst. The supervisor linkage segments are as_linkage ("active supervisor"), ai_linkage ("active initialization"), ws_linkage ("wired supervisor"), and wi_linkage ("wired initialization"). They are defined near the beginning of the standard header. Unless you remove the init_seg and temp_seg keywords from initialization programs and their linkage sections, it is not possible to profile supervisor initialization programs (because the profiling information would otherwise be discarded as the system finished initialization), but this is rarely a problem.

If wired code is to be profiled and you use -long_profile, the hcs_ gate and its linkage section must be wired because they are referenced by the virtual CPU time and paging calculation operators; this is not necessary if you use only -profile. If profiling a procedure that is specified as wired in the header but whose linkage section is specified as unwired, change the linkage section to be wired.

Interrupt side code can be meaningfully profiled only with -profile, not with -long_profile, because interrupt code is not run in any particular process, and therefore the virtual CPU time calculation (which is per process) returns random results. This may lead to overflow faults while running on the PRDS. Because -profile does not require these calculations, you can use it with interrupt code.

NOTES

The procedures that generate the system tape must first find the necessary segments to place on the system tape and put them there in a manner that can later be read by BCE and the initializing programs themselves. The system tape generating procedures find this information by scanning a header segment that contains names of programs and databases to be placed on the tape, along with other control information about the segments. *

There is a set of search rules specifying which directories are to be searched and the search order when looking for the specified segments. These rules can be contained in a segment, or you can use default rules. If you use no search segment, only the directory >ldd>hardcore>execution is searched for the programs to be placed on the tape.

The standard system tape header used to generate the BCE/Multics system tape is * located in the segment >ldd>hardcore>info>hardcore.header. The standard headers contain many examples of valid header syntax. When you modify a header, first, if possible, locate an example of the modification elsewhere in the header since the semantics of the header are complicated.

This command assumes the name of the header segment is path.header, where path is the path given. The output listing is placed in a segment path.list in your working directory.

The search file must contain a list of directories to be searched, one directory name per line. A blank line signifies your working directory.

Name: get_flagbox

SYNTAX AS A COMMAND

```
get_flagbox flagbox_variable
```

SYNTAX AS AN ACTIVE FUNCTION

```
[get_flagbox flagbox_variable]
```

FUNCTION

returns the value of a specified flag in the BCE/Multics communication area.

ARGUMENTS

flagbox_variable
  is one of the valid flagbox variables listed below.

  bce_command command
    a command to be invoked by BCE when it reaches a command level. This command is set so that if the system crashes, the command is automatically executed. See the *Multics System Maintenance Procedures* manual (AM81).

  keyword
    can be either a number from 1 to 36 or the corresponding name of one of the flagbox flags as given below. The names of the flagbox flags are:

1	auto_reboot
2	booting
3	bit3
4	rebooted
5	unattended
6	bit6
7	bit7
8	bit8
9	bit9
10	bit10
11	bit11
12	bit12
13	bit13
14	bit14
15	bit15
16	bit16
17	bit17
18	bit18
19	bit19
20	bit20
21	bit21
22	bit22
23	bit23
24	bit24
25	bit25
26	bit26
27	bit27
28	bit28
29	bit29
30	bit30
31	bit31
32	bit32
33	bit33
34	bit34
35	bit35
36	bit36

ACCESS REQUIRED

Privileged access to phcs_ is required to use this command.

Name: get_uid_with_lastname

SYNTAX AS A COMMAND

```
get_uid_with_lastname name
```

FUNCTION

is used by the ison command in the master.ec. It searches the URF for all users with a given last name, and prints their full names and their assigned Person_ids. This information is useful in suggesting, or in predicting what the system will suggest as, a Person_id for a new user who has the same last name as one or more users already registered.

ARGUMENTS

name
: is the last name of a prospective new user.

NOTES

This command prints a series of lines of the form:

```
User_id for "Lastname, Firstname I." is "Person_id"
```

followed by:

```
Number of users with lastname "Lastname" is N.
```

ACCESS REQUIRED

Use of this command requires read access on the system URF located at >udd>sa>a>urf.

Name: hc_pf_meters

SYNTAX AS A COMMAND

```
hc_pf_meters {-control_args}
```

FUNCTION

prints system-wide statistics concerning page faults taken on hardcore segments, including those taken on ring-0 stacks.

CONTROL ARGUMENTS

-first N, -ft N
: causes the output to be sorted in descending order by number of page faults. Only the first N segments in this sequence are printed.

-report
: prints data accumulated since the last invocation of the command with -reset or -report_reset, or since the last bootload (if the command has not been invoked with -reset or -report_reset). This is the default.

-report_reset, -rr
: generates a report and then performs the reset operation.

-reset, -rs
: resets the metering interval for the invoking process. Does not print the report unless -report is specified.

-sort
: causes the output to be sorted in descending order by number of page faults. If not given, the output is sorted in ascending order by primary segment name.

ACCESS REQUIRED

This command requires access to phcs_.

EXAMPLES

See Appendix A for a representation of the configuration deck used to create the system from which the sample was taken.

```
        hc_pf_meters -first 20

    Total metering time    0:15:23

    Segment                                 Page Faults  % Total

    dbm_seg                                    173        9.54
    str_seg                                    149        8.21
    tty_area                                   127        7.00
    definitions_                                99        5.46
    hcs_                                        75        4.13
    bound_file_system                           62        3.42
    stack_0.024                                 55        3.03
    stack_0.023                                 49        2.70
    hphcs_                                      48        2.65
    stack_0.013                                 44        2.43
    stack_0.028                                 44        2.43
    stack_0.016                                 43        2.37
    stack_0.025                                 41        2.26
    stack_0.010                                 40        2.21
    stack_0.026                                 40        2.21
    stack_0.011                                 38        2.09
    stack_0.014                                 38        2.09
    stack_0.015                                 36        1.98
    stack_0.007                                 35        1.93
    stack_0.022                                 34        1.87

    Total Hardcore Page Faults                1814
```

Name: hp_delete, hpdl

SYNTAX AS A COMMAND

```
hpdl path
```

FUNCTION

deletes segments or directories (including their inferior directories, segments, and links (if accessible)) which cannot be deleted by the delete or delete_dir commands due to connection failures or other problems resulting from volume deregistration or the failure of emergency shutdown.

ARGUMENTS

path
: is the pathname of the segment or directory to be deleted. The star convention may not be used.

ACCESS REQUIRED

The hp_delete command requires access to the highly-privileged hphcs_ gate. Access to the system_privilege_ gate is required as well. All discretionary (ACL) and nondiscretionary (AIM) access control is observed by this command. The segment or directory to be deleted may, however, have a lower ring bracket than the user's current validation level.

NOTES

The deletion is not logged in the syserr log.

The user is queried once before performing the deletion.

Out-of-service directories may be deleted.

The segment safety switch is ignored.

Name: hp_delete_acl, hpda

SYNTAX AS A COMMAND

```
hpda {path} {User_ids} {-control_args}
```

FUNCTION

removes entries from the access control lists (ACLs) of segments, multisegment files, directories, and gates. This command operates on objects whose ring brackets prevent them from being operated on by the delete_acl command.

ARGUMENTS

path
: is the pathname of a segment, multisegment file, directory, or gate. If it is -wd, -working_dir, or omitted, the working directory is assumed. If path is omitted, no User_id can be specified. The star convention can be used.

User_ids
: are access control names that must be of the form Person_id.Project_id.tag. All ACL entries with matching names are deleted. (For a description of the matching strategy, refer to the *Multics Programmer's Reference Manual*, Order No. AG91.) If no User_id is given, the user's Person_id and current Project_id are assumed.

CONTROL ARGUMENTS

-all, -a
: causes the entire ACL to be deleted with the exception of an entry for *.SysDaemon.*.

-brief, -bf
: suppresses the message "User name not on ACL."

-directory, -dr
: specifies that only directories are affected. The default is segments, multisegment files, directories, and gates.

-segment, -sm
: specifies that only segments, multisegment files, and gates are affected.

ACCESS REQUIRED

The user needs modify permission on the containing directory. The system administrator needs access to the highly privileged gate hphcs_.

NOTES

If the hp_delete_acl command is invoked with no arguments, it deletes the entry for the user's Person_id and current Project_id on the ACL of the working directory.

An ACL entry for *.SysDaemon can be deleted by specifying either *.SysDaemon.* or *.SysDaemon. The user should be aware that in deleting access to the SysDaemon project he prevents Backup.SysDaemon.* from saving the segment or directory (including the hierarchy inferior to the directory) on tape, Dumper.SysDaemon.* from reloading it, and Retriever.SysDaemon.* from retrieving it.

EXAMPLES

```
   hp_delete_acl news .Faculty. Jones
```

deletes from the ACL of news all entries with Project_id Faculty and the entry for Jones.*.*.

```
   hpda beta.** ..
```

deletes from the ACL of every segment, multisegment file, directory, and gates (in the working directory) whose entryname has a first component of beta all entries except the one for *.SysDaemon.*.

```
   hpda beta.** .. -sm
```

deletes from the ACL of only all segments, and multisegment files, and gates (in the working directory) whose entryname has a first component of beta all entries except the one for *.SysDaemon.*.

Name: hp_delete_vtoce

SYNTAX AS A COMMAND

```
hp_delete_vtoce pvname vtoc_index {-control_args}
```

FUNCTION

deletes a specified Volume Table of Contents Entry (VTOCE). You can use it when cleaning up after a sweep_pv to get rid of orphans, or whenever you want a forward connection failure.

ARGUMENTS

pvname
    is the name of the physical volume on which the VTOCE to be expunged exists.

vtoc_index
    is the index (in octal) of the VTOCE to be expunged.

CONTROL ARGUMENTS

-brief, -bf
: suppresses the message announcing the deletion of the VTOCE, which is only printed if no questions are asked.

-clear
: uses the privileged entry that sets an entire VTOCE to zero, rather than deleting it. Give -clear only when a VTOCE contains invalid information that may cause problems (reused addresses, crashes, etc.) if you delete it as usual, since it leaves the volume on which the VTOCE existed in an inconsistent state. Salvage such volume with the volume salvager after you have deleted all the seriously inconsistent VTOCEs. Don't use this control argument to delete a reverse connection failure VTOCE (an ordinary orphan).

-force, -fc
: forces the deletion of the VTOCE if it is an orphan, with no intervening questions; if it is not, you must supply also -no_check to suppress all questions.

-no_check, -nch
: suppresses the check made to see whether or not the VTOCE is an orphan; if it is not, deleting it causes a forward connection failure in its parent directory.

-query, -qy
: always queries whether or not to delete the VTOCE, even if it is an orphan.

ACCESS REQUIRED

You require access to the phcs_ and hphcs_ gates.

NOTES

You cannot use this command to delete the VTOCE of an active segment. The default is to check whether the VTOCE is an orphan, and delete it if it is, or ask whether to delete it if is not.

Name: hp_set_acl, hpsa

SYNTAX AS A COMMAND

```
hpsa path mode1 User_id1 ...  modeN {User_idN} {-control_args}
```

FUNCTION

manipulates the access control lists (ACLs) of segments, multisegment files, directories, and gates. This command operates on objects whose ring brackets prevent them from being operated on by the set_acl command.

ARGUMENTS

path
: is the pathname of a segment, multisegment file, directory, or gate. If it is -wd or -working_dir, the working directory is assumed. The star convention can be used and applies to either segments and multisegment files or directories, depending on the type of mode specified in mode1.

modei
: is a valid access mode. For segments, multisegment files, or gates, any or all of the letters rew; for directories, any or all of the letters sma with the requirement that if modify is present, status must also be present. Use null, "n" or "" to specify null access.

User_idi
: is an access control name that must be of the form Person_id.Project_id.tag. All ACL entries with matching names receive the mode modei. (For a description of the matching strategy, see the *Multics Programmer's Reference Manual*, Order No. AG91.) If no match is found and all three components are present, an entry is added to the ACL. If the last mode_i has no User_id following it, the Person_id of the user and current Project_id are assumed.

CONTROL ARGUMENTS

(either control argument is used to resolve an ambiguous choice between segments and directories that occurs only when modei is null and the star convention is used in path):

-directory, -dr
: specifies that only directories are affected.

-segment, -sm
: specifies that only segments and multisegment files are affected. This is the default.

ACCESS REQUIRED

To use this command, a system administrator must have access to the highly privileged gate hphcs_.

EXAMPLES

```
   hp_set_acl *.pll rew *
```

adds to the ACL of every segment in the working directory that has a two-component name with a second component of pll an entry with mode rew to *.*.* (everyone) if that entry does not exist; otherwise it changes the mode of the *.*.* entry to rew.

```
   hpsa -wd sm Jones.Faculty
```

adds to the ACL of the working directory an entry with mode sm for Jones.Faculty.* if that entry does not exist; otherwise it changes the mode of the Jones.Faculty.* entry to sm.

```
   hpsa alpha.basic rew .Faculty. r Jones.Faculty.
```

changes the mode of every entry on the ACL of alpha.basic with a middle component of Faculty to rew, then changes the mode of every entry that starts with Jones.Faculty to read.

```
   hpsa >sl1>hphcs_ re Jones.Work
```

adds to the ACL of hphcs_ an entry with mode re for Jones.Work.

Name: hpset_dir_ring_brackets, hpdsrb

SYNTAX AS A COMMAND

```
hpdrsb path {optional_args}
```

FUNCTION

modifies the ring brackets of any directory, including directories whose ring brackets are 0,0. This command operates on directories whose ring brackets prevent them from being operated on by the set_dir_ring_brackets command.

ARGUMENTS

path
: is the relative or absolute pathname of the directory whose ring brackets are to be modified.

optional_args

rb1
: is the number to be used as the first ring bracket of the directory. See "Notes" below.

rb2
: is the number to be used as the second ring bracket of the directory. See "Notes" below.

ACCESS REQUIRED

To use this command, a system administrator must have access to the highly privileged gate hphcs_.

NOTES

If rb2 is omitted, the ring bracket is set to rb1. If rb1 and rb2 are omitted, they are set to the user's current validation level. The ring brackets must be in the allowable range 0 through 7 and must have the ordering:

```
rb1 is less than or equal to rb2
```

Name: hpset_ring_brackets, hpsrb

SYNTAX AS A COMMAND

```
hpsrb path {optional_args}
```

FUNCTION

modifies the ring brackets of any segment, including segments whose ring brackets are 0,0,0. This command operates on objects whose ring brackets prevent them from being operated on by the set_ring_brackets command.

ARGUMENTS

path
: is the relative or absolute pathname of the segment whose ring brackets are to be modified.

rb1
: is the number to be used as the first ring bracket of the segment. See "Notes" below.

rb2
: is the number to be used as the second ring bracket of the segment. See "Notes" below.

rb3
: is the number to be used as the third ring bracket of the segment. See "Notes" below.

ACCESS REQUIRED

To use this command, a system administrator must have access to the highyl privileged gate hphcs_.

NOTES

If rb3 is omitted, the third ring bracket is set to rb2. If rb2 and rb3 are omitted, the ring brackets are set to rb1. If rb1, rb2, and rb3 are omitted, they are set to the user's current validation level. The ring brackets must be in the allowable range 0 through 7 and must have the ordering:

```
rb1 is less than or equal to rb2 is less than or equal to rb3
```

Name: idump

SYNTAX AS A COMMAND

```
idump path {-control_args}
```

FUNCTION

is the same as the backup_dump command, except that it assumes it is called as a standalone command instead of being invoked by another hierarchy dumping command. Refer to the description of the backup_dump command for details.

Name: inhibit_pv

SYNTAX AS A COMMAND

```
inhibit_pv pvname {-control_arg}
```

FUNCTION

sets up for the evacuation of a physical volume and inhibits further segment creation on that volume.

ARGUMENTS

pvname
    is the name of a physical volume.

CONTROL ARGUMENTS

-off
    allows segment creation again on the named physical volume.

ACCESS REQUIRED

You need access to the hphcs_ gate.

NOTES

Attempting to inhibit a volume already inhibited or to disinhibit one not inhibited produces a diagnostic.

Name: install

SYNTAX AS A COMMAND

```
install path {-control_args}
```

FUNCTION

requests installation of a system control table.

ARGUMENTS

path
    is the relative or absolute pathname of the table to be installed. You must give the appropriate suffix (e.g., pdt).

CONTROL ARGUMENTS

-all, -a
    installs all attributes.

-attributes, -attr
    installs only nonsecurity-related attributes. (Default: if you specify no control arguments)

-authorization, -auth
    installs only security-related attributes.

NOTES

The request is transmitted to the system control process. This process validates the request, attempts to perform the installation, and sends a message indicating the success or failure of the installation.

This command reports PDT parameters that exceed limits specified for the project in the SAT, but it allows the PDT to be installed. If the SAT limits are not subsequently raised, they are enforced at login time and a message indicating that is logged. This is done for the initial ring, max ring, grace time, and pdir quota parameters.

A project administrator can install a PDT only; a system administrator can also install a number of additional tables.

EXAMPLES

```
install Alpha.pdt -auth
```

installs the security-related attributes in the PDT named alpha.pdt found in the project administrator's working directory.

```
install >udd>Demo>Demo.pdt
```

installs the PDT named demo.pdt found in the Demo project directory.

A message like this is sent to the project administrator when the installation has been completed:

```
From Initializer.SysDaemon.z (install) 1321.0:
installed Demo.pdt for Renoir.Demo.a.
```

Name: instr_speed

SYNTAX AS A COMMAND

```
instr_speed
```

FUNCTION

measures the speed of a processor by timing a series of code sequences.

NOTES

The code sequences are chosen to test the more common and important sequences of instruction as well as to get a general idea of the limiting speed that can be expected. It is better if you select a processor with the set_proc_required command before running timing tests.

Code sequences that take longer than expected, probably due to interrupt or fault action, are factored out and not counted.

Name: interrupt_meters, intm

SYNTAX AS A COMMAND

```
intm {-control_args}
```

FUNCTION

prints out metering information for input/output multiplexer (IOM) channel interrupts.

CONTROL ARGUMENTS

-channel N
: prints out only interrupt metering information for IOM channel N.

-iom N
: prints out only interrupt metering information for those channels on IOM N.

-report_reset, -rr
: generates a full report and then performs the reset operation.

-reset, -rs
: resets the metering interval for the invoking process so that the interval begins at the last call with -rs specified. Having never given -rs in a process is equivalent to having specified it at system initialization time.

-total, -tt
: prints out only the total IOM and non-IOM interrupt metering information.

ACCESS REQUIRED

You need access to phcs_ or metering_gate_.

NOTES

If you give intm without control arguments, it prints a full report.

The following are brief descriptions of the metering information printed out by intm.

Int
: is the number of interrupts that occurred.

Avg Time
: is the average time (in milliseconds) needed to handle each interrupt.

% CPU
: is the percentage of total CPU time needed to handle the interrupts.

Name
: is the name of the device on the channel.

The following are descriptions of the totals printed by intm.

Chan
: is the total of all IOM channel interrupts. The times printed are based on the total time spent in the per-channel interrupt handlers.

Other
: is the total of all IOM interrupts. Each IOM interrupt may cause the handling of several channel interrupts. The times printed include only that time in the common interrupt path and exclude time spent in the per-channel handlers.

Total
: is the total of all interrupts handled by the system.

EXAMPLES

The following is an example of the information printed when you invoke intm without control arguments. (See Appendix A for a representation of the configuration deck used to create the system from which the samples were taken.)

```
Total metering time    0:15:25

 IOM Ch       Int  Avg Time  % CPU  Name

 A    6.        8  2.838     0.00  IOM A special
 A    8.        1  1.088     0.00  prtd
 A   10.       21  1.312     0.00  prtb
 A   13.     6867  3.120     0.46  fnp c
 A   14.       26  0.601     0.00  opca
 A   16.    11320  1.156     0.28  tapa
 A   18.    19254  5.083     2.11  fnp b
 A   20.    16536  0.822     0.29  dska
 A   21.     1708  0.769     0.03  dska
 A   22.     4310  0.785     0.07  dskb
 A   23.       15  0.610     0.00  dskb
 A   24.    12212  0.786     0.21  dskb
 A   25.      614  0.747     0.01  dskb
 A   26.     7678  0.777     0.13  dska
 A   27.      121  0.777     0.00  dska
 A   28.    13796  0.797     0.24  dskc
 A   29.      342  0.870     0.01  dskc
 A   30.     6063  0.754     0.10  dskd
 A   31.      251  0.740     0.00  dskd
 A   36.    14815  0.800     0.26  dskf
 A   37.     1150  0.783     0.02  dskf
 A   38.     4062  0.774     0.07  dskg
 A   39.        2  0.836     0.00  dskg
 B   15.    11050  2.930     0.70  fnp f
 B   18.     1534  2.772     0.09  fnp a
 B   19.    12053  3.277     0.85  fnp e
 B   20.    16474  0.806     0.29  dskb
 B   21.     2084  0.788     0.04  dskb
 B   22.     3934  0.756     0.06  dska
 B   23.       20  0.584     0.00  dska
 B   24.    12312  0.789     0.21  dska
 B   25.      540  0.792     0.01  dska
 B   26.     7872  0.778     0.13  dskb
 B   27.      134  0.738     0.00  dskb
 B   28.    14295  0.756     0.23  dskd
 B   29.     1676  0.736     0.03  dskd
 B   30.     3240  0.754     0.05  dskc
 B   32.        2  1.288     0.00  dske
 B   36.    13097  0.822     0.23  dskg
 B   37.      278  0.749     0.00  dskg
 B   38.     5828  0.787     0.10  dskf

Chan       227595  1.490     7.33
Ovhd       224799  0.226     1.10
Total      224799  1.735     8.43
```

Name: io_error_summary

SYNTAX AS A COMMAND

```
io_error_summary {-control_args}
```

FUNCTION

scans the syserr log and summarizes I/O errors in a brief report.

CONTROL ARGUMENTS

-cylinders, -cyl
: separates the disk device error by cylinder and head. Only disk control syserr messages can be separated.

-detailed_status, -dtst
: displays detailed status if available.

-device STRs, -dv STRs
: reports information for the device(s) named, where STRs are device types ("prt") or device names ("prtb").

-for T
: computes the ending time from the starting time, where T is a relative time (such as "1 hour").

-from DT, -fm DT
: starts scanning the log at the date/time given.

-hex_detailed_status, -hxdtst
: displays detailed status in hexidecimal if available.

io_command, -ioc
: displays the I/O command that was being executed when the abnormal status occurred. The command is displayed in octal, in parenthesis, prior to the interpreted status.

-status status_list, -st status_list
: reports information for the IOM status listed, where status_list is the IOM major and substatus ("0310" or "4310").

-tape_data_bit_in_error, -tdbie
: displays the data bit(s) in the detailed status that were in error.

-to DT
: stops scanning the log at the date/time given.

ACCESS REQUIRED

You need read permission on the log segments and status permission on their containing directories.

NOTES

If you give no -from, the scan starts with the earliest message in the syserr log. You can specify the ending time by using -for or -to, but not both. If you omit both, the scan terminates with the last message in the log. All dates and times must be in a format acceptable to convert_date_to_binary_ (see the *Multics Subroutines and I/O Modules* manual, AG93).

Name: iod_command

SYNTAX AS A COMMAND

```
iod_command io_daemon_command {args}
```

FUNCTION

permits execution of I/O daemon commands from within admin exec_coms invoked by the I/O daemon x command.

ARGUMENTS

io_daemon_command
    is the I/O daemon command to be executed.

args
    are any arguments needed to implement the specified command.

NOTES

You can't issue the go command using iod_command.

EXAMPLES

```
   iod_command defer_time pica_10 30
```

can be used within an I/O daemon admin exec_com to set the auto defer time of the pica_10 minor device of the current I/O daemon driver to 30 minutes.

Name: iod_tables_compiler

SYNTAX AS A COMMAND

```
iod_tables_compiler path
```

FUNCTION

is the translator for the I/O daemon tables source language. Source segments to be translated by iod_tables_compiler must have a name ending with the suffix iodt. The name of an object segment produced by iod_tables_compiler is the same as that of the corresponding source segment with the iodt suffix removed. The object segment is placed in your working directory.

ARGUMENTS

path
: is the relative or absolute pathname of the source segment to be translated.

Name: iod_val

SYNTAX AS AN ACTIVE FUNCTION

```
[iod_val key]
```

FUNCTION

supplies several preset driver parameters to be used in driver admin exec_coms. Site administrators use the iod_val active function in conjunction with the driver x command to set up and modify these exec_coms.

ARGUMENTS

key
: is a character string parameter name associated with the value to be returned. The key, defined during initialization of the given driver, may be one of the following:

For all standard drivers:

device
: the name of the major device that the driver is running.

station_id
: the name of the station_id that the driver is running (equivalent to the major device). The default is the name of the major device if the station is not a remote device.

request_type
: the name of the request type that is being run on the driver.

channel
: the name of the iom or tty channel of the driver.

<minor device>
: the name of the request type that is being processed on the minor device.

rqt_string
: a string of request type names, separated by spaces, of all (printer, punch, etc) request types the driver can process. This key is equivalent to the request_type key if the driver is running only one minor device.

For remote drivers:

request_type
: the request type for a single printer device, if present.

pun_rqt
: the request type for a single punch device, if present.

NOTES

If a key is given that has not been defined, the string "undefined " is returned.

Name: is_he_user

SYNTAX AS AN ACTIVE FUNCTION

```
[is_he_user Person_id]
```

FUNCTION

function returns "true" if its argument is the name of a user in the URF hash table (urf.ht) in the current working directory; it returns "false" otherwise.

NOTES

The function is used in the master.ec segment to detect errors in the input arguments.

Name: is_legal_proj

SYNTAX AS AN ACTIVE FUNCTION

```
[is_legal_proj Project_id]
```

FUNCTION

returns "true" if its argument is listed in the current SAT copy (smf.cur.sat) in the user's working directory, or it returns "false" otherwise.

NOTES

This function is used in the master.ec segment to detect errors in the input arguments.

Name: link_meters

SYNTAX AS A COMMAND

```
link_meters {-control_arg}
```

FUNCTION

prints out per-process information regarding use of the Multics linker. The statistics are obtained from the Process Descriptor Segment (PDS) of the process. System-wide linkage information can be obtained with the system_link_meters command (described later in this section).

CONTROL ARGUMENTS

-report_reset, -rr
: generates a full report and then performs the reset operation.

-reset, -rs
: resets the metering interval for the invoking process so that the interval begins at the last call with -reset specified. If -reset has never been given in a process, it is equivalent to having been specified at process initialization time.

ACCESS REQUIRED

This command requires access to phcs_ or metering_gate_.

NOTES

If the link_meters command is given with no control argument, it prints a full report.

The following are brief descriptions of the variables printed by the link_meters command.

slot
: is a time slot into which the calls to the linker are broken down. The four slots are for calls completed in less than 25 milliseconds, calls completed in between 25 and 50 ms, calls completed in between 50 and 75 ms, and calls completed in more than 75 ms.

calls
: is the number of calls to the linker that are completed in each time slot and the total number of calls made to the linker by the process.

avg time
: is the average time (in milliseconds) to completion for a call in each slot and the average time to completion for all calls to the linker made by the process.

avg pf
: is the average number of page faults for a call in each slot and the average number of page faults for all calls made by the process.

tot time
: is the total virtual time (in seconds) taken by calls in each slot and the total virtual time spent in the linker by the process. It equals calls times average time.

% time
: is the percentage of total linker time for the process that was taken by calls in each slot and the percentage taken by all calls.

EXAMPLES

The following is an example of the information printed when the link_meters command is invoked with no control argument. (See Appendix A for a description of the configuration deck used to create the system from which the metering sample was taken.)

```
Linkage Meters:

slot     calls  avg time   avg pf   tot time  % time

<25       320     6.673     0.3      2.135    76.1
25-50      14    28.432     2.1      0.398    14.2
50-75       0     0.000     0.0      0.000     0.0
>75         1   272.756    10.0      0.273     9.7
   -----  ---------  -----  -------
Total     335     8.377     0.4      2.806
```

Name: list_as_requests

SYNTAX AS A COMMAND

```
list_as_requests
```

FUNCTION

displays any answering service requests currently pending in the answering service request queue message segment (>sc1>as_request.ms).

ACCESS REQUIRED

You must have o access to as_request.ms to display your own requests. You must have r access to as_request.ms to display requests entered by others.

Name: list_delegated_projects

SYNTAX AS A COMMAND

```
list_delegated_projects sat_path {Project_id}
```

FUNCTION

is used by the who_delg command in master.ec to list the administrator(s) of a given delegated project, or to list all delegated projects, with their administrators.

ARGUMENTS

sat_path
: is the pathname of the SAT to be searched for delegated projects. If the entryname portion of sat_path is given without a suffix of sat, the suffix is added (unless the entryname portion is simply "sat").

Project_id
: is the Project_id of the delegated project whose administrators are to be listed. If this argument is not given, all delegated projects are listed.

NOTES

The output from this command is of the form

```
Project      Administrator
Project_id1    Admin1
               Admin2
               Admin3
Project_id2    Admin1
               Admin2
```

Name: list_extra_personids

SYNTAX AS A COMMAND

```
list_extra_personids
```

FUNCTION

lists Person_ids that are registered in the PNT but do not exist in any PDT. Person_ids in a PDT but not in the PNT are also listed.

*

NOTES

This command is useful for checking the consistency of system tables and listing those Person_ids that can be removed from the PNT to compress it. (See remove_user for more information.)

The command also references the SAT in >sc1 and the PDTs for all valid projects in >sc1>pdt.

First, the PDTs are scanned, and a list of users missing from the PNT is printed; this list is probably due to errors by project or accounting administrators. Then, a list of * Person_ids in the PNT but not in the PDTs is printed.

This command is expensive. Therefore run it on absentee or at least using file_output (see the *Multics Commands and Active Functions* manual, AG92) so that the results can be saved.

Name: list_mst

SYNTAX AS A COMMAND

```
list_mst reel_id {names}
```

FUNCTION

lists segments on a system tape (a BCE/Multics bootload tape).

ARGUMENTS

reel_id
: is the reel identification number of the tape to be listed. This reel identification number, which is site dependent, can be up to 32 characters long, and can include a density specification to indicate the density of the tape being written, as in "060341,den=1600".

names
: are names of segments to be listed. If you give no names, all the segments on the tape are listed. You can use the star convention.

NOTES

A summary line, giving the length of each segment and its primary name, is printed out for each segment listed. A special name is printed out for segments written with the first_name keyword of the generate_mst command because the proper name of such segments does not appear on the tape. A diagnostic is issued if you provide names that match no segments on the tape.

Name: list_partitions

SYNTAX AS A COMMAND

```
list_partitions pvname
```

FUNCTION

lists the locations and sizes of all the partitions on a specified physical volume (see clear_partition and dump_partition).

ARGUMENTS

pvname
: is the name of the physical volume whose partitions are to be listed.

NOTES ON OUTPUT FORMAT

The output consists of a header, which lists the physical and logical volume names, the PVID and LVID, the size of the volume in pages, the size of the VTOC in both pages and VTOCEs, the size of the paging region, and the number of partitions. It is followed by a table listing the name, first record, and size of all partitions and other regions on the volume. All numbers in the table are given in both decimal and octal (in parentheses), and all other numbers in the output are decimal.

Name: list_proc_required

SYNTAX AS A COMMAND

```
list_proc_required {-control_arg}
```

SYNTAX AS AN ACTIVE FUNCTION

```
[list_proc_required {control_arg}]
```

FUNCTION

determines the group of CPUs on which the invoking process can be run or the default group of CPUs for all processes that have not requested specific CPUs.

CONTROL ARGUMENTS

-priv
: indicates that this command applies to the default group of CPUs for processes that have not requested specific CPUs. If omitted, this command applies to the group of CPUs for the invoking process only.

ACCESS REQUIRED

This command requires access to phcs_ or metering_gate_.

NOTES

When invoked as a command without the -priv control argument, list_proc_required indicates that the set of CPUs needed for this process is the system default by printing "(default)" following the list of CPUs. This information is not provided when list_proc_required is invoked as an active function. If invoked as an active function, it returns a string of CPU tags that represent the group of CPUs requested (e.g., "ABCF").

This command prints the list of CPUs required as an uppercase string. If invoked as an active function, it prints the returned list of CPU tags in uppercase.

EXAMPLES

This command is most useful when used in conjunction with the set_proc_required command to verify that the restriction specified in an earlier invocation of set_proc_required is still in operation. The effect of set_proc_required can be canceled by the system because of dynamic reconfiguration without notification to the process affected. If the following set of commands are input:

```
set_proc_required A

{other commands}

list_proc_required
```

an output of "A" from list_proc_required indicates that all commands between the set_proc_required and the list_proc_required were run entirely on CPU a. Any other output indicates that the effect of the set_proc_required has been canceled due to dynamic reconfiguration.

Name: list_vols

SYNTAX AS A COMMAND

```
list_vols {lv_name} {-control_args}
```

SYNTAX AS AN ACTIVE FUNCTION

```
[list_vols {lv_name} {-control_args}]
```

FUNCTION

prints information about currently-mounted physical or logical volumes. Several of the items printed by list_vols can also be obtained as return values by invoking list_vols as an active function.

ARGUMENTS

lv_names
: prints information about the logical volume lv_name. This is the default if a name is given without -lv or -pv preceding it.

CONTROL ARGUMENTS

-grand_total, -gtt
: prints a single number about the total records left or records used on the system. This is used in conjunction with the -rec_used control argument, or if no other control argument is given, -rec_left is assumed. This allows a command line equivalent of active function usage as: [list_vols -rec_left] or [list_vols -rec_used] where -rec_left is also the default.

-lv name(s)
: prints information about the logical volume(s) named. A single name or several names, separated by space, can be given.

-pv name
: prints information about only the physical volume named.

-records, -rec
: prints only the number of records on the specified volume(s), exclusive of records occupied by partitions and the volume table of contents (VTOC). This is one of the items that can be obtained as an active function return value.

-records_left, -rec_left
: prints only the number of records on the specified volume(s) that are currently unused and are available to hold the pages of segments and directories. This is one of the items that can be obtained as an active function return value. (This is the default.)

-records_used, -rec_used
: prints the number of records on the specified volume(s) that are used. This can be obtained as an active function value.

-totals, -tt
: does not print information for individual physical volumes but rather totals and prints for each logical volume.

ACCESS REQUIRED

This command requires access to phcs_ or metering_gate_ as well as mdc_.

NOTES

If no volume name is given, the list_vols command prints information about all mounted logical volumes.

If physical volume information is being printed, there may be up to two flags printed immediately to the right of the drive name. The flags are "X" (if a drive is inoperative) and "I" (if a drive is inhibited for segment creation).

If a physical volume information is being printed, the average segment size, to the nearest record, is printed per physical volume.

If either logical volume or physical volume information is being printed, the percentages used or left for Records and VTOCEs is printed.

If list_vols is used as an active function, either the -records or the -records_left control argument must be given.

If the -totals argument is given together with the name of a logical volume, a single line containing totals information for that logical volume is printed.

If physical volume information is being printed (-totals not given), the output lines contain the following items:

```
   Drive  flag  Records  Left  VTOCEs  Left  PV Name  PB/PD LV Name
```

If logical volume information is being printed (-totals given), the output lines contain the following items:

```
   Records   Left   VTOCEs   Left PB/PD Lv Name
```

The following are brief descriptions of the above variables.

Drive
: is the name of the drive on which the physical volume is mounted.

flag
: is the letter "X" if the drive is inoperative.

Records
: is the number of records not occupied by partitions or the VTOC, and therefore usable for the pages of segments and entries.

Left
: is the number of records currently unused and therefore available for the pages of segments and directories.

VTOCEs
: is the number of VTOC entries.

Left
    is the number of unused VTOC directories.

PV Name
    is the name of the physical volume.

PB/PD
    contains "pb" if the logical volume is public, and "pd" if it has been designated as being available to hold the segments in process directories.

LV Name
    is the name of the logical volume.

EXAMPLES

The active function:

```
[list_vols -records_left root]
```

returns the number of records left on the root logical volume.

The command:

```
list_vols -tt root
```

prints the following totals for the root logical volume.

```
Records   Left  VTOCEs   Left PB/PD LV Name
  50267 4026 20185 7874 pb root
```

The following is an example of the information printed when the list_vols command is invoked with no control arguments. (See Appendix A for a representation of the configuration deck used to create the system from which the metering samples was taken.)

```
Drive     Records    Left  %  VTOCEs Left  %  Avg   PV       PB/PD  LV Name
                                              Size Name

dskd_17     64504   47202  73 13440  9664  72   4  alpha01   pb pd  Alpha
dskd_18     64504   48364  75 13440  9633  72   4  alpha02   pb pd  Alpha
dska_05     36428    4042  11  8400  2470  29   5  mul03     pb     Multics_Pubs
dska_06     36428    3006   8  8400  2557  30   5  mul01     pb     Multics_Pubs
dskb_19     36428    3323   9  8400  2690  32   5  mul02     pb     Multics_Pubs
dskb_26     36429    4963  14  8400  2689  32   5  mul05     pb     Multics_Pubs
dskb_27     36429    4655  13  8400  2353  28   5  mul04     pb     Multics_Pubs
dska_01     36308    2241   6  9000   620   7   4  pub01     pb     Public
dska_03     36268    1751   5  9200   924  10   4  pub07     pb     Public
dska_04     36268    1574   4  9200  1107  12   4  pub04     pb     Public
dska_09     36268    2166   6  9200   725   8   4  pub02     pb     Public
dskb_17     36269    2046   6  9200  1255  14   4  pub05     pb     Public
```

```
dskb_18    36268   2336   6   9200    641    7    3  pub08     pb      Public
dskb_21    36268   1304   4   9200   1021   11    4  pub06     pb      Public
dskb_22    36268   2145   6   9200    982   11    4  pub03     pb      Public
dskf_13    64504  14177  22  13440   7274   54    8  rel01             Release
dskf_14    64504  15569  24  13440   6964   52    7  rel02             Release
dskg_19    64504  53171  82  13440  11860   88    7  rel03             Release
dskg_20    64504  54967  85  13440  11925   89    6  rel04             Release
dskc_01    64503  22432  35  13440   8473   63    8  xpub01    pb  pd  Xpublic
dskc_02    64503  24790  38  13440   8418   63    7  xpub02    pb  pd  Xpublic
dskf_03    64503  23042  36  13440   8369   62    8  xpub03    pb  pd  Xpublic
dskf_04    64503  21783  34  13440   8294   62    8  xpub04    pb  pd  Xpublic
dskg_29    64504  29269  45  13440   9780   73    9  xpub05    pb  pd  Xpublic
dskg_30    64504  26593  41  13440   9934   74   10  xpub06    pb  pd  Xpublic
dskd_27    64504  20398  32  13440   5469   41    5  ypub03    pb  pd  Ypublic
dskd_28    64504  20797  32  13440   5246   39    5  ypub04    pb  pd  Ypublic
dskf_09    64503  21896  34  13440   5327   40    5  ypub05    pb  pd  Ypublic
dskf_10    64503  22001  34  13440   5325   40    5  ypub06    pb  pd  Ypublic
dskg_21    64503  22327  35  13440   5518   41    5  ypub01    pb  pd  Ypublic
dskg_22    64503  19898  31  13440   5449   41    5  ypub02    pb  pd  Ypublic
dskc_07    64504  14528  23  13440   6461   48    7  zpub01    pb      Zpublic
dskc_08    64504  14702  23  13440   6580   49    7  zpub02    pb      Zpublic
dskd_23    64504  16920  26  13440   6044   45    6  zpub03    pb      Zpublic
dskd_24    64504  14026  22  13440   6393   48    7  zpub04    pb      Zpublic
dskg_25    64504  21269  33  13440   7208   54    6  zpub05    pb      Zpublic
dskg_26    64504  20487  32  13440   7245   54    7  zpub06    pb      Zpublic
dske_06    37089  13407  36   5100   4121   81   24  list01    pb      list_1
dska_12    37561   4428  12   2735    942   34   18  list02    pb      list_2
dska_02    37309   2673   7   4000   1929   48   16  list03    pb      list_3
dskc_11    64504  63894  99  13440  13381  100   10  pdir01    pb  pd  pdir_1
dskc_12    64504  63546  99  13440  13380  100   15  pdir02    pb  pd  pdir_1
dskd_21    64504  63070  98  13440  13283   99    9  pdir05    pb  pd  pdir_3
dskd_22    64504  63299  98  13440  13292   99    8  pdir06    pb  pd  pdir_3
dskd_31    64504  63957  99  13440  13397  100   12  pdir07    pb  pd  pdir_4
dskd_32    64504  63946  99  13440  13404  100   15  pdir08    pb  pd  pdir_4
dskf_05    64504  63504  98  13440  13345   99   10  pdir11    pb  pd  pdir_6
dskf_06    64504  63756  99  13440  13346   99    7  pdir12    pb  pd  pdir_6
dska_07    36208   4999  14   7000     81    1    4  root5     pb      root
dska_11    36209   5392  15   7000   2137   31    6  root6     pb      root
dska_16    31283   2798   9   9000   2979   33    4  rpv       pb      root
dskb_23    36208   4609  13   7000    123    2    4  root3     pb      root
dskb_24    36108   4811  13   7500    408    5    4  root4     pb      root
dskb_25    36108   3852  11   7500    301    4    4  root2     pb      root
```

Name: list_volume_registration, lvr

SYNTAX AS A COMMAND

```
lvr control args
```

FUNCTION

displays the registration information for a physical volume or a logical volume with all its physical volumes.

CONTROL ARGUMENTS

The "-physical_volume" or the "-logical_volume" control argument is required and must immediately follow the command name. (Only one may be supplied.)

-physical_volume PV_NAME, -pv PV_NAME
    specifies that the registration information for the given physical volume is to be displayed.

-logical_volume LV_NAME, -lv LV_NAME
    specifies that the registration information for the given logical volume is to be displayed.

-brief, -bf
    if given with the "-logical_volume" control argument, will specify that only the names of the physical volumes are to be displayed. No other information is given. This control argument is ignored when "-physical_volume" is specified.

ACCESS REQUIRED

The user of this command is required to have "re" access to the mdc_priv_ gate and "s" to >lv.

EXAMPLES

```
-> lvr -lv root
 lvname:   root
 lvid:               516404202641 (!gwBcBjbbBBBBBB)
 public:             yes
 owner:              Initializer.SysDaemon
 min_access_class:   0:000000 ()
 max_access_class:   7:777777 (system_high)
 acs_path:           >lv>root.acs

npv:                 2

 pvname:   rpv
 pvid:               516404202602 (!gwBcBjCBBBBBB)
 serial:             3M - 345666
 model:              451
 location:           dska 34
 date registered:    02/05/84 1552.7 edt Sun

 pvname:   root2
 pvid:               520225300221 (!hBdhbDLbBBBBBB)
 serial:             unspecified
 model:              501
 location:           uninitialized
 date registered:    02/05/84 1554.1 edt Sun

-> lvr -lv root -bf

rpv
root3
```

Name: load_mpc

SYNTAX AS A COMMAND

```
load_mpc {mpc_name} {-control_args}
```

FUNCTION

loads ITRs or application firmware or both into MPCs.

ARGUMENTS

mpc_name
: is the name of the MPC to be tested or reloaded or both. This name must appear on an mpc card in the config deck. If this argument is omitted, the -channel control argument must be given.

CONTROL ARGUMENTS

-channel channel_name, chn channel_name
: specifies a channel name, where channel_name is of the form {iomtag}channel_no (for example, a14). The iomtag field must be the tag of a configured IOM and is required on multiple IOM systems. The channel_no field is an octal channel number. If this control argument is used, the mpc_name argument is optional. If both are used, the channel must be connected to the mpc specified.

-itr
: loads only the ITRs; the standard firmware is not reloaded.

-firm
: loads ony the standard firmware; ITRs are not run.

-revision RV, -rev RV
: specifies which revision of the firmware is to be loaded, where RV is a 2-character firmware revision code. If multiple revisions exist and this argument is omitted, you are queried as to which revision to load.

-time, -tm
: prints timings for each program loaded into the MPC.

-brief, -bf
: withholds printing of the names of the programs as they are run.

ACCESS REQUIRED

To use load_mpc, you must have access to the phcs gate.

NOTES

By default, this command suspends I/O on all devices connected to the selected MPC, resets the controller, runs all the known ITRs, reloads the standard firmware (including device routines for urmpc), and restores I/O on all devices connected to the controller.

If any abnormal conditions occur, the program displays the status that occurred, and stops. I/O is left in a suspended state, because the MPC has been left in an unusable state. In order to return the controller to operation, it is necessary to restore the firmware, using either this command or TOLTS (documented in the *Multics Online Test and Diagnostics Manual*, Order No. AU77).

You can use load_mpc on disk MPCs only if they are fully cross barred.

Firmware and ITR modules are found in the test and diagnostics (T&D) deckfile created by the load_tandd_library command (see the *Multics Online Test and Diagnostics Reference Manual* (AU77).

This command informs the operator, through the main initializer console, of major events during command execution; the events include suspending I/O, running ITRs, loading firmware, and resuming I/O (or leaving I/O suspended).

Name: load_ctl_status

SYNTAX AS A COMMAND

```
load_ctl_status {group} {-control_args}
```

FUNCTION

prints the current status of the system load control groups, i.e., prints selected items from the system copy of the master group table (MGT).

ARGUMENTS

group
: prints only the header and the line for the group named; otherwise, prints one line for each group in the MGT. Each line gives the maximum number of primary load units; the current number of primary load units; the current number of secondary load units; the group's current load, as a percentage of the total allowable system load; and, if the group has an absolute maximum, the total and maximum number of units.

CONTROL ARGUMENTS

-long, -lg
: requests a long format header.

-total, -tt
: requests that only a header be printed.

NOTES

If the priority scheduler is enabled, then each line gives, in addition, the interactive and absentee work class of the group, and the header contains two additional lines giving the defined work classes and their percentages, for the current shift.

Name: lock_mca

SYNTAX AS A COMMAND

```
lock_mca
```

FUNCTION

locks (disables) input to all maintenance channel adapters (MCAs) from the console.

Name: make_commands

SYNTAX AS A COMMAND

```
make_commands path_name
```

FUNCTION

creates a Limited Service Subsystem (LSS) control segment from an ASCII input segment. The control segment is referenced by the Limited Service System (LSS) when it is limiting the commands and percentage of CPU time of a user (see also the enter_lss command).

The input segment consists of a series of statements. Each statement is composed of two parts. The first part is the name of the command to be transformed; i.e., the command that is to be typed by the user in a limited system. If there is more than one name for the command, they should all be enclosed in parentheses and separated from each other by one or more blanks. The name field is terminated by a colon preceded by any number of blanks.

The second part of each statement is the pathname (which may be a relative pathname) of the command to be executed when the user types one of the names in the first part. If a relative pathname is used, it is relative to the current working directory. If only an entryname is given, the standard system search rules are applied. It is followed by any number of blanks and terminated by a semicolon. If the pathname is omitted (semicolon still required), it is assumed to be the same as the last name in the name field.

The first and second parts of each statement may be separated from each other by any number of blanks or tabs. Newlines are ignored and are allowed anywhere. Comments enclosed between "/*" and "*/" are allowed and are treated as blanks.

If the first two statements have as their first part the names "ratio" and "interval", respectively, the second parts of the two statements are assumed to be decimal integers to be assigned to the ratio and interval_length variables of the LSS control segment. Otherwise, the two variables are set to zero.

The ratio and interval variables control the amount of CPU time used by the process. The LSS forces the process to use no more than (interval/ratio) virtual CPU seconds in each (interval) real second(s) (see "Examples" below). If it attempts to do so, the process is rendered inactive for the remainder of the interval.

ARGUMENTS

path_name
: is the pathname of an ASCII input segment that has the name path_name.ct. (The .ct suffix is assumed if it is not included.) The output segment has the same entryname as the input segment with the .ct suffix removed, and is placed in the working directory.

This page intentionally left blank.

EXAMPLES

A project administrator wants to create a limited command environment for a project consisting of student users. The first step is to create the segment >udd>Students>student_commands.ct, with the following contents:

```
/* set the ratio and interval */

ratio:    60;
interval: 120;

/* define the commands */

(list ls):          >abc>special$list;
logout:   ;
edit:     bsys;
start:    ;
hold:     ;
(pr print):         ;
```

Then, to convert this segment to an LSS control segment, type:

```
make_commands student_commands.ct
```

The command then creates an output segment named student_commands, which can be put to use with the enter_lss command.

In the above example, the ratio is set to 60, and the interval is set to 120, indicating that no more than 2 virtual CPU seconds are allowed per 120 real seconds.

NOTES

See the enter_lss command for additional information.

Name: make_volume_labels

SYNTAX AS A COMMAND

```
make_volume_labels volume_spec {-control_args}
```

FUNCTION

generates data for producing stickers to be attached to volumes (e.g., tape reels and disk packs). For each volume name specified, two labels are prepared. The first label contains the name of the resource, and the second label contains a three-character authentication code for the volume. Any number of volume names may be specified, including volume name series.

ARGUMENTS

volume_spec
: is either a single volume name, or the sequence "volume_name1 -to volume_name2". When the second sequence is specified, both volume_name1 and volume_name2 must end with one or more digits. Label data will be produced for each volume in the sequence specified. Each volume name may be as long as 32 characters. A volume name may not contain spaces.

CONTROL ARGUMENTS

-auth_only
: specifies that labels containing the volume names are not to be generated. If this argument is specified, only authorization code labels are produced. The default is to produce both types of labels.

-height N
: specifies that the labels on the stock to be used are N lines in height, measured from the first line of one label to the first line of the next label. The default for N is 9. The data produced assumes printing at six lines per inch.

-name_only
: specifies that labels containing authorization codes are not to be generated. If this argument is specified, only volume name labels are produced. The default is to produce both types of labels.

-output_switch switchname, -osw switchname
: specifies the name of the I/O switch over which the label data is to be written. If this control argument is omitted, the data will be written over the I/O switch named "label_stream".

-width N
: specifies that the labels on the stock to be used are N characters across. The default for N is 50.

NOTES

Each sticker will contain data in a 5x5 format. Uppercase characters are distinguished from lowercase characters by additional underscoring. If the data in 5x5 format cannot be accommodated on the size sticker specified, a warning is printed, the data is truncated, and an indicator is appended to the label to show that the name has been truncated. However, all stickers always contain a line printed in normal characters, containing both the complete volume name and the authentication.

If the sequence "volume_name1 -to volume_name2" is specified, both volume names must contain numeric portions, and the rightmost numeric portion of volume-name2 must not be numerically less than that of volume_name1. Labels are generated for each volume in the sequence. Only the rightmost numeric portion of the resource name is incremented. Leading zeroes are not generated in the significant numeric portion unless they are explicitly specified as part of volume_name1.

The I/O switch on which the labels are to be written is assumed to be attached and opened for stream output before the command is invoked. (See the description of the io_call command in the *Multics Commands and Active Functions* manual, Order No. AG92, for information on attaching and detaching I/O switches.) *

Name: map355

SYNTAX AS A COMMAND

```
map355 pathname {-control_args}
```

FUNCTION

is used to assemble a program written in the FNP assembler language, map355. The command does not assemble the program directly. Instead, it prepares a GCOS job deck to perform the assembly and calls the GCOS Environment Simulator to do the work.

ARGUMENTS

pathname
: is the pathname of the source program to be assembled. The suffix of map355 need not be given by the user; it is assumed.

CONTROL ARGUMENTS

-argument list, -ag list
: specifies a list of arguments to be passed to the GCOS Environment Simulator.

-check
specifies that the compiler only perform a syntax check of the source. No object segment is created.

-comdk
creates a GCOS comdk segment. This segment contains a BCD version of the source program. It is created in the working directory with a suffix of comdk.

-gcos_list, -gcls
creates a GCOS listing segment in the working directory. This is a BCD version of the listing segment. It has a suffix of glist.

-list, -ls
creates a listing segment that documents the compilation. The listing is created in the working directory, and has a suffix of list.

-macro_file path
specifies the pathname of the macro file to be used for the assembly. If omitted, >ldd>comm>fnp>info>355_macros is used.

-noconvert
specifies that the input segment is a GCOS comdk, rather than an ASCII segment. If this control argument is used, the source segment must have a suffix of comdk.

NOTES

This command creates a series of segments for use by the GCOS simulator. Some are created in the working directory, some are created in the process directory, and some through links in the working directory to the process directory. These segments and links are normally deleted when the command terminates, leaving just the object segment, which has a suffix of objdk.

Refer to the *GCOS Environment Simulator* manual, Order No. AN05, for more information on the use of the GCOS Environment Simulator.

WARNING

The map355 command creates links in the working directory to segments to be placed in the process directory. If the process terminates in the middle of a compilation (new_proc or a crash), these links will remain. This means that the next time the command is invoked, it will fail because the links point to a nonexistent directory. Even though the command fails, the bad links will be unlinked and subsequent invocations will work correctly.

LIST OF SEVERITY VALUES

The map355 command sets the following severity values to be returned by the severity active function when the "map355" keyword is used:

Value	Meaning
0	No error (or command not yet used)
1	The assembly produced warning flags
2	The objdk segment could not be created

Name: mc_trace, mct

SYNTAX AS A COMMAND

```
mct path {-control_args}
```

FUNCTION

gives a snapshot of machine conditions and history registers (resulting from hardware faults and interrupts) incurred while executing another Multics command or subroutine.

ARGUMENTS

path
: is the absolute or relative pathname of the segment that is to be traced.

CONTROL ARGUMENTS

-all
: captures machine conditions and history registers for every fault and interrupt that occurs in your process. This control argument cannot be used with -hc or the path argument.

-brief, -bf
: suppresses printing the "-->" prompt.

-buffer N, -buf N
: sets the machine condition trace buffer size to N, where N is a decimal integer value from 1 to 16, and represents the buffer size in units of 1024 words (1K). The default buffer size is 5K words.

-hc SEG
: captures machine conditions and history registers for faults and interrupts that occur in the hardcore segment SEG while your process is in execution. SEG can be a hardcore segment name or number. This control argument cannot be used with -all or the path argument.

NOTES

This command initiates the segment specified by the path argument, and creates the machine condition trace buffer in your process directory. The number of machine conditions and history register sets that can be stored is directly related to the size of the trace buffer. There is an approximate 8 to 1 ratio of machine conditions to history registers (e.g., in a 5K buffer there would be storage for 79 sets of machine conditions and 10 sets of history registers, allowing room for a trace buffer header). The trace buffer is temporarily "wired" (i.e., the segment remains in main storage and is not subject to removal by the dynamic paging mechanism). The hardcore snapshot or trace mechanism is then enabled and mc_trace goes into a request loop after printing "-->" as your prompt on the error_output switch. The valid user reponses while in this request loop are as follows:

. prints out the command name "mc_trace" on the user_output switch.

.q turns the hardcore snapshot mechanism off, unwires the machine condition buffer, and returns to Multics command level.

..<command> calls the Multics command processor and executes <command> as a Multics command (e.g., ..who).

.rpt n <command> calls the Multics command processor to loop n times, executing the specified Multics command <command>; n is an integer from 1 to 99999999 (e.g., .rpt 10 who).

.pmc m n displays machine conditions in octal starting with machine condition set m for n sets. The integer m therefore represents a negative index from the last set of machine conditions stored (e.g., the request ".pmc 8 2" would be interpreted to mean, "display two sets of machine conditions starting from the last machine conditions stored at position 8"). If n is not specified, then all machine conditions starting at m to the last machine conditions stored are displayed. If neither m nor n are specified, all sets of machine conditions are displayed.

.pmci m n same as .pmc above except that the machine conditions are displayed in interpreted format.

.pscu m n same as .pmc above except that only the System Control Unit (SCU) data for the specified number of machine conditions is printed, displayed in interpreted format.

.hr m n displays history registers in octal, starting with history register set m for n sets. The variables m and n are defined as in .pmc above.

.hrou m n same as .hr above except that only the Operations Unit (OU) history register is displayed in octal.

.hrcu m n same as .hr above except that only the Control Unit (CU) history registers are displayed in octal.

.hrdu m n same as .hr above except that only the Decimal Unit (DU) history registers are displayed in octal.

.hrau m n same as .hr above except that only the Appending Unit (AU) history registers are displayed in octal.

.hranl m n same as .hr above except that the specified number of history registers are displayed in interpreted format.

.hrlgd produces a list of abbreviations used with the .hranl request above.

The mc_trace command invokes a condition handler for the "any_other" condition. When any unusual system condition is encountered, a message indicating the condition that was raised is displayed on the error_output I/O switch, and control is passed to the request loop. At this time, any of the valid requests described above can be entered. For further information on system conditions, refer to the *Multics Programmer's Reference Manual*, Order No. AG91).

ACCESS REQUIRED

To use mc_trace, you must have re access to phcs_.

EXAMPLES

Assume that you have written a program that generates an op_not_complete fault while executing a csl (combine bit strings left) EIS instruction in a particular sequence (e.g., descriptors fall on page boundaries). This is clearly a hardware problem, but because it only occurs when a particular set of events take place, it is very difficult for the CSR to trouble-shoot. For simplicity, call this program onc_csl. To run this program under control of mc_trace, execute the following sequence of commands:

```
mc_trace onc_csl
     start trace with default buffer size of 5K words.
--> ..set_proc_required a
     run with only one processor, in this case processor a.
--> ..onc_csl
     execute the program onc_csl
op_not_complete condition raised, enter command.
```

At this point, the op_not_complete condition has occurred, and the machine condition history for the last 103 machine conditions should be preserved in the machine condition buffer. You can now selectively display these machine conditions.

```
--> ..set_proc_required
    reset set_proc_required (run on any processor now)

--> ..file_output onc_trace
    direct the output from the user_output I/O switch to the file named onc_trace
    created in your working directory.

--> .pmc
    display the entire machine condition buffer. In this case, the output goes to the
    onc_trace file.

--> .pmci
    additionally display the machine conditions in interpretive format.

--> .pscu
    also display only SCU data in interpretive format.

--> .hr 1
    display history registers from last fault (in this case, the op_not_complete fault)
    in octal format.

--> .hranl 1
    display a composite analysis of the last set of history registers.

--> .hranl 2 1
    display a composite analysis of the next to last set of history registers.
```

--> ..revert_output
direct the output from the user_output I/O switch from the onc_trace file back to your terminal.

--> ..dprint onc_trace
print the onc_trace file on a remote printer.

--> .q
return to Multics command level.

If the op_not_complete fault does not occur on a consistent basis and you suspect that it only occurs randomly when a particular sequence of page faults and interrupts occurs, you can use another program called "flush", which generates heavy paging activity, and can loop on these commands several times by executing the following line in place of the onc_csl command line:

```
--> .rpt 99999999 flush;onc_csl
op_not_complete condition raised, enter command
-->
```

At this point, you can proceed as in the above example and print the machine conditions to a file or display them on the terminal.

OUTPUT PRODUCED WITH THE .PMC, .PMCI, AND .PSCU REQUESTS

.pmc REQUEST

The .pmc request produces an octal dump of the machine conditions, separated by the logical data in the machine conditions (e.g., pointer registers, processor registers). The format is dependent on the state of the user_output I/O switch. If the user_output I/O switch is attached to a file or a terminal with a line length greater than or equal to 104 characters, then the output is formatted in lines of eight octal words per line. If the user_output I/O switch is attached to a terminal with a line length less than 104 characters per line, then the output is formatted in lines of four octal words per line.

```
*****Machine Conditions at mc_trace_buffer|2410*****

Pointer Registers
000035000043 004646000000 000017000043 000000000000
000062000043 005362000000 000135000043 000000000000
000014000043 006712000000 000062000043 000000000000
000062000043 004060000000 000356400043 000000000000

Processor Registers
011127005716 000000000031 060614000030 000015000720
000000000012 000000000004 000000000000 001714442000

SCU Data
000113050202 000000000043 400356000004 000000540000
044571000200 000000000400 700000100440 000140100540

Software Data
121040000012 420040000006 000000000000 000000000000
000000000000 000000000000 000000104422 532243555724

EIS Pointers and Lengths
000400000000 000400000000 000000000070 004077777774
000102000030 000000000000 030356000004 000000000077
```

.pmci REQUEST

The .pmci request displays the machine conditions in an interpreted format, as shown below.

```
*****Machine Conditions at mc_trace_buffer|2410*****

pr0 (ap) 35|4646 bound_sss_wired_|4646
pr1 (ab) 17|0    sst_seg|0
pr2 (bp) 62|5362 pds|5362
pr3 (bb) 135|0   dirlockt_seg|0
pr4 (lp) 14|6712 as_linkage|6712
pr5 (lb) 62|0    pds|0
pr6 (sp) 62|4060 pds|4060
pr7 (sb) 356|0   complex_decimal_op_|0 (bound_pl1_runtime_|0)

x0  11127   x1   5716   x2      0   x3     31
x4  60614   x5     30   x6     15   x7    720
a 000000000012  q 000000000004  e 0
Timer reg - 1714442, Ring alarm reg - 0

SCU Data:

   240    000113050202 000000000043 400356000004 000000540000
          044571000200 000000000400 700000100440 000140100540

(DF1) Page Fault (43)
By: 113|44571 bound_file_system|44571
Referencing: 356|0 complex_decimal_op_|0  (bound_pl1_runtime_|0)
On: cpu a (#0)
Indicators:  ^bar
APU Status:  sd-on, pt-on, ptw
CU Status:   rfi
instructions:
  246  700 000 100 440  mlr    (rl),(),fill(700)
  247  000 140 100 540  mlr    (pr,rl),(pr,rl),fill(000)

Time stored: 05/31/77  1355.3 mst Tue (104422532243555724)
Ring:          0

EIS Pointers and Lengths:

   260    000400000000 000400000000 000000000070 004077777774
          000102000030 000000000000 030356000004 000000000077
```

.pscu REQUEST

The .pscu request displays only the SCU data from the machine conditions in an interpreted fashion, as shown below.

```
*****Machine Conditions at mc_trace_buf|2410*****

SCU data at mc_trace_buf|240

   240     000113050202 000000000043 400356000004 000000540000
           044571000200 000000000400 700000100440 000140100540

(DF1) Page Fault (43)
By: 113|44571 bound_file_system|44571
Referencing: 356|0 complex_decimal_op_|0  (bound_pl1_runtime_|0)
On: cpu a (#0)
Indicators:  ^bar
APU Status:  sd-on, pt-on, ptw
CU Status:   rfi
Instructions:
  246  700 000 100 440  mlr    (rl),(),fill(700)
  247  000 140 100 540  mlr    (pr,rl),(pr,rl),fill(000)
```

OUTPUT PRODUCED BY THE .HR REQUEST

The .hr request produces an octal dump of the history registers. The output is separated by the history register type being dumped. The format is dependent on the state of the user_output I/O switch. If the user_output I/O switch is attached to a file or a terminal with a line length greater than or equal to 104 characters, then the output is formatted in lines of eight octal words per line. If the user_output I/O switch is attached to a terminal with a line length less than 104 characters per line, then the output is formatted in lines of four octal words per line. If the .hrou, .hrcu, .hrdu, or .hrau requests are selected, only the requested history register type is dumped.

```
*****History Registers at mc_trace_buf|210*****

Operations Unit (OU) History Registers
315000315100 137777012705 450000450300 177777012707
255000255300 177777012710 221000221100 136777012711
220000221340 114777012717 220000220100 135777012720
740000740300 175777012721 741000741300 176777012722
450000450300 177777012725 446000446300 177757012726
720000720500 135777012727 440000440300 175777012730
450000450300 177777012731 621000621500 122777000051
431210431100 123777000052 431210431000 037777000054

Control Unit (CU) History Registers
700137352120 000046050200 300117352120 027342050020
200117352000 027220050024 200017252100 027574242100
700137352120 000050050200 300137352120 027344050021
200117352000 027224050024 200017252100 027576242100
600137621100 000052050200 200117621100 027566050014
200017431007 020000042014 700137370120 000054042200
300137370120 027524050021 200117370000 034304050024
300017352120 034314042020 300005352046 034314402002

Decimal Unit (DU) History Registers
737757037737 744243410017 737757037737 744243410017
777757037737 744243410017 737757037717 744243410017
737757037737 744243410017 777757037737 744243410017
737757037737 744243410017 737757037717 744243410017
777757037737 744243410017 737757037737 744243410017
737757037737 744243410017 777757037717 744243410017
737757037737 744243410017 777757037737 744243410017
737757037737 744243410017 777757037717 744243410017

Appending Unit (AU) History Registers
002342006066 027555724775 003576007262 024520464775
002342006066 027553424775 002346001420 000272204775
002342006066 027555744775 003576007262 024520504775
002342006066 027553444775 002346001420 000272244775
002342006066 027555764775 003576007262 024520524775
003576001420 000275664775 003576001420 000200004775
003576007262 024520544775 002342006066 027555244775
002556001420 000343044775 002552007370 026403144000
```

OUTPUT PRODUCED BY THE .HRANL REQUEST

The .hranl request produces a composite analysis of the history registers in the trace buffer. The output produced is dependent on the state of the user_output I/O switch. If the user_output I/O switch is attached to a terminal with a line length less than 104 characters, the output appears as below. If the user_output I/O switch is attached to a file or a terminal with a line length greater than or equal to 104 characters, then the octal representation of the history registers is displayed in addition to the sample below:

```
*****History Registers at mc_trace_buf|210*****

Composite Analysis of History Registers

 HR                           c
id##   IC_____  opcd__  tag_  y  seg#_  offset__  mc  flags
CU 1            epp2    n*    i   415         46   4  pi pa ri ic wi it
AU 2                               7  6   336046   4  ap sm pm
CU 2             "      n*    n   234      27342   4  pa ri ic it cl
AU 3                             12  5    533342   4  ap sm pm
CU 3             "            d   234      27220   4  pa ic cl dr
AU 4                                       27220   4
CU 4        46  spri2         o   234      27574  24  pa it cs
AU 5                             12  5    533574   4  ap sm pm
CU 5        47  epp2    n*    i   415         50   4  pi pa ri ic wi it
AU 6                               7  6   336050   4  ap sm pm
CU 6             "      n*    n   234      27344   4  pa ri ic wi it cl pb
AU 7                             12  5    533344   4  ap sm pm
CU 7             "            d   234      27224   4  pa ic cl dr
AU10                                       27224   4
CU10        50  spri2         o   234      27576  24  pa it cs
AU11                             12  5    533576   4  ap sm pm
CU11        51  eaxl          i   415         52   4  pi pa ic wi it
AU12                               7  6   336052   4  ap sm pm
CU12             "            d   415      27566   4  pa ic it ol dr
AU13                                       27566   4
OU16                                                  rb rs of -d ar qr xl
CU13        52  fld     dl    d   415      20000   4  pa ol dr
AU14                                       20000   4
OU17                                                  dl rs of -d ar qr
CU14        53  epp4    n*    i   415         54   4  pi pa ri ic wi it
AU15                               7  6   336054   4  ap sm pm
CU15             "      n*    n   234      27524   4  pa ri ic wi it cl pb
AU16                             12  5    533524   4  ap sm pm
CU16             "            d   255      43742   4  pa ic cl dr
AU17                                       43742   4
CU17        54  epp2    n*    n   255      43752   4  pa ri it cl
AU20                               4  4  3225752   4  ap sm pm
```

OUTPUT PRODUCED BY THE .HRLGD REQUEST

The .hrlgd request produces a list of the abbreviations used with the .hranl request above.

```
Abbreviations used in History Register Analysis

____________CU Legend_____________     ____________OU Legend_____________
cy = cycle type (d = direct operand)   >>flags<<<
(i=instr. fetch,o=operand,F=fault)     9b = 9-bit byte (IT modifier only)
(n=indirect,x=xec,*=nop,e=EIS)         ar = A-register in use
mc = memory command                    d1 = first divide cycle
(00=rrs,sp; 04=rrs,dp; 10=rcl,sp)      d2 = second divide cycle
(12=rmsk,sp; 16=rmsk,dp; 20=cwr,sp)    dl = direct lower operand
(24=cwr,dp; 32=smsk,sp; 36=smsk,dp)    du = direct upper operand
(40=rd/lck; 54=rgr; 56=sgr)            in = first ou cycle
(60=wrt/ulck; 62=con; 66=xec; 72=sxc)  it = IT character modifier
>>>flags<<<                            oa = mantissa alignment cycle
-y = memory address invalid            oe = exponent compare cycle
br = BAR mode                          of = final OU cycle
cl = control unit load                 om = general OU cycle
cs = control unit store                on = normalize cycle
dr = direct operand                    os = second cycle of multiple ops
fa = prepare fault address             qr = Q-register in use
ic = IC value is odd                   rb = opcode buffer loaded
it = AR/PR reference                   rp = primary register loaded
in = inhibited instruction             rs = secondary register loaded
ol = operations unit load              sd = store data available
os = operations unit store             -d = data not available
pa = prepare operand address           x0 = index 0 in use
pb = port busy or data from cache      x1 = index 1 in use
pi = prepare instruction address       x2 = index 2 in use
pl = port select logic not busy        x3 = index 3 in use
pn = prepare final indirect address    x4 = index 4 in use
pt = prepare operand tally             x5 = index 5 in use
ra = request alter word                x6 = index 6 in use
ri = request indirect word             x7 = index 7 in use
rp = executing repeat
sa = store alter word
si = store indirect word
tr = transfer condition met
wi = request instruction fetch
xa = prepare execute interrupt address
xe = execute double from even ICT
xi = execute interrupt present
xo = execute double from odd ICT
```

```
____________DU Legend____________  ____________APU Legend___________
mc = data mode (b,4,6,9,w)         seg# = SDWAMR and PTWAMR numbers if
offset = descriptor counter        corresponding MATCH bits are set.
>>>flags<<<                        offset = final store address
()a = prepare alignment count for  mc = ring number (TSR.TRR)
      numeric operand (1,2)
a() = load alpha operand (1,2)     >>>flags<<<
al = adjust length                 an = final address, non-paged
as = alpha store                   ap = final address, paged
bd = binary-decimal execution      f  = access violation or directed
                                        fault
bg = blanking gate                 fd = fetch descriptor segment PTW
c0 = force stc0                    fh = fault waiting
cg = character operation           fs = fetch SDW
d() = descriptor active (1,2,3)    md = modify descriptor segment PTW
da = data available                mp = modify PTW
db = decimal-binary execution      p1 = fetch PTW
dd = decimal unit idle             p2 = fetch PTW+1
di = decimal unit interrupted      pm = MATCH in PTWAM
dl = decimal unit load             sm = MATCH in SDWAM
ds = decimal unit store
ei = mid-instruction interrupt enabled
en = end instruction
es = end sequence
ff = floating result
fl = first data buffer load
fp = first pointer preparation
fs = end sequence
l() = load descriptor (1,2,3)
ld = length = direct
lf = end first pointer preparation
lv = level < word size
lx = length exhaust
l< = length < 128
mp = executing MOPs
n() = load numeric operand (1,2)
nd = need descriptor
ns = numeric store
op = operand available
pc = alpha packing cycle
pl = prepare operand length
pp = prepare operand pointer
r() = load rewrite register (1,2)
re = write-back partial word
rf = rounding                      -------DU Legend-------
rl = rewrite register 1 loaded     xg = exponent network
rw = du=rd+wt control interlock    xm = extended al,ql modifier
sa = select address register       +g = add-substract execution
sg = shift procedure               *g = multiply-divide execution
```

Name: mcs_version

SYNTAX AS A COMMAND

```
mcs_version {fnp_tag}
```

FUNCTION

prints the version of the core image most recently loaded into the specified FNP.

ARGUMENTS

fnp_tag
: where fnp_tag is the identifier of the FNP whose version is to be printed. It can be a, b, c, d, e, f, g, or h. If it is omitted, a is assumed.

Name: meter_fnp_idle

SYNTAX AS A COMMAND

```
meter_fnp_idle fnp_name {-control-args}
```

FUNCTION

reads FNP idle metering information at specified intervals and stores the information in a segment for later viewing through the display_fnp_idle command.

ARGUMENTS

fnp_name
: identifies the FNP for which idle time is to be recorded.

CONTROL ARGUMENTS

-directory path, -dr path
: specifies that the segments in which idle time information is to be stored are to be created in the directory with pathname path. The default is the user's working directory. Segment names are of the form fnp_idle_data.F.MMDDYY.HHMMSS.I where F is fnp_name, MMDDYY.HHMMSS is the starting date and time of recording, and I is the specified interval in minutes.

-interval N
: specifies that metering information is to be sampled every N minutes. The default is 1.

-stop, -sp
: terminates meter reading on this FNP. No other control argument may be specified with -stop.

ACCESS REQUIRED

Use of this command requires access either to the metering_gate gate or the phcs_ gate.

NOTES

Information is appended to the idle time segment until the -stop control argument is encountered or the process terminates. Maximum length of an idle time segment is 64K (to avoid exhausting a 256K ASTE). If a segment becomes full, a new one is created. On average, taking readings at one-minute intervals would fill a 64K segment in a little over three weeks (if a single process in which the command was invoked were to run that long).

Each invocation of the command creates a new segment.

EXAMPLES

The command

```
   meter_fnp_idle a -interval 5
```

creates a segment in the working directory into which will be stored idle time meter readings taken on FNP "a" at five-minute intervals.

Name: meter_gate, mg

SYNTAX AS A COMMAND

```
mg STR {entry_nm} {-control_arg}
```

FUNCTION

is used to interpret and print per-system metering information for entries in specified hardcore gates.

ARGUMENTS

STR
: is the name of the gate segment to be examined; i.e., hcs_, phcs_, hphcs_, ioi_, hc_backup_, etc.

entry_nm
: is the name of a single entry in the specified gate. Only the information for that entry is printed. If entry_nm is not specified, information for all entries is printed. No control argument can be given if an entry_nm is specified.

CONTROL ARGUMENTS

-average, -av
: sorts the output on the average time spent in each entry.

-call, -cl
: sorts the output on total calls to each entry.

-page, -pg
: sorts the output on the average number of page faults in each entry.

-reset, -rs
: resets the metering interval for the invoking process so that the interval begins at the last call with -reset specified. If -reset has never been given in a process, it is equivalent to having been specified at system initialization time.

-time, -tm
: sorts the output on the total time spent in each entry.

NOTES

If the meter_gate command is given with no control argument, it does not sort the output.

The output header consists of the time the system was brought up, the current time, and the total charge time (which equals total_cpu_time minus idle_time). Also printed is the total number of calls to the gate, the amount of time spent in the entries that were called, and the percentage of total charged time that was spent in the entries that were called.

Metering information is collected only for gate segments defined with the "hgate" macro, and only for those entries in the segment defined with the "gate" macro (refer to the gate_macros.incl.alm include file for these macros, and refer to the source listing of a particular gate to apply this principle). For example, some hardcore gate entries are defined with the "fgate" macro for efficiency or because ring 0 stack history is abandoned during the call (e.g., hcs_$block); such gate entries are not metered.

The following is a brief description of the variables printed out by the meter_gate command.

calls
    is the total number of times the gate entry point was called.

pcnt
    is the percentage of total charge time spent in the called segment.

avg
    is the average virtual time in milliseconds spent in the called segment.

pfault
    is the average number of page faults incurred during a call to a segment through the specified entry.

entry name
    is the name of an entry point to the gate.

EXAMPLES

The following is an example of the information printed when the meter_gate command is invoked with hcs_ as the name of the gate segment to be examined. (See Appendix A for a representation of the configuration deck used to create the system from which the metering sample was taken.)

```
Metering since 12/12/83  1247.2 mst Mon.
Total non-idle time at 12/12/83  1302.6 mst
Mon = 1 hr. 12 min. 4 sec.

Gate meters for hcs_:                  total calls = 136778.
0 hr. 12 min. 44 sec. or 17.675% spent in calls through gate.

 calls    pcnt     avg      pwait    entry name

   17     0.01     15.00    0.06     add_acl_entries
    1     0.00     10.03    0.00     add_dir_inacl_entries
    1     0.00     17.32    0.00     add_inacl_entries
   79     0.06     30.75    0.29     append_branch
    6     0.00      9.70    0.17     append_link
   42     0.00      0.32    0.00     assign_channel
    6     0.00      0.55    0.00     assign_linkage
```

```
  76    0.01     7.63    0.00    chname_file
 164    0.04    11.23    0.00    chname_seg
   4    0.00     5.96    2.00    combine_linkage
  90    0.06    31.01    0.34    create_branch_
 103    0.09    38.56    0.21    delentry_file
  18    0.01    33.10    0.06    delentry_seg
   8    0.00    13.39    0.13    delete_acl_entries
  37    0.00     0.25    0.00    delete_channel
2369    0.32     5.85    0.00    flush_consecutive_pages
1182    0.19     7.01    0.00    flush_pages
  28    0.00     6.56    0.11    force_write
 154    0.00     0.42    0.00    fs_get_access_modes
1610    0.13     3.55    0.00    fs_get_brackets
 983    0.05     2.34    0.04    fs_get_mode
2181    0.20     3.98    0.05    fs_get_path_name
 525    0.01     0.51    0.04    fs_get_seg_ptr
1718    0.03     0.81    0.14    fs_search_get_wdir
  74    0.01     6.13    0.42    fs_search_set_wdir
2118    0.84    17.09    1.09    get_access_class
 262    0.05     8.65    0.02    get_access_class_seg
   1    0.00     1.26    0.00    get_alarm_timer
3714    0.01     0.11    0.00    get_authorization
  47    0.01     5.74    0.02    get_defname_
   3    0.00     7.12    0.00    get_dir_ring_brackets
 127    0.00     0.10    0.00    get_initial_ring
 903    0.21     9.87    0.70    get_link_target
  43    0.00     0.26    0.00    get_lp
 230    0.06    11.98    0.01    get_max_length
 141    0.03     9.70    0.01    get_max_length_seg
 366    0.00     0.28    0.00    get_process_usage
 117    0.03    11.77    0.42    get_ring_brackets
  74    0.01     6.77    0.00    get_safety_sw
  10    0.00     5.53    0.00    get_safety_sw_seg
  54    0.01     6.11    0.09    get_search_rules
  52    0.00     1.27    0.00    get_system_search_rules
 334    0.08    10.33    0.49    get_uid_file
6546    0.04     0.25    0.01    get_uid_seg
  60    0.02    15.87    1.35    get_user_access_modes
 370    0.08     9.78    0.36    get_user_effmode
 753    0.00     0.12    0.01    high_low_seg_count
   1    0.00     0.16    1.00    history_regs_set
4575    0.87     8.24    0.12    initiate
1400    0.25     7.60    0.52    initiate_count
  55    0.03    25.83    0.51    initiate_search_rules
 230    0.06    10.75    0.01    list_acl
  12    0.06   204.02    1.25    list_dir
  11    0.00    11.11    0.00    list_dir_acl
   4    0.00     5.30    0.00    list_dir_inacl
  13    0.00    14.78    0.69    list_inacl
1649    0.88    23.15    0.33    make_entry
```

```
   6914      1.59       9.96     0.38      make_ptr
    478      0.35      31.72     0.49      make_seg
     73      0.00       0.14     0.00      proc_info
     62      0.03      21.54     1.47      quota_read
   8030      0.13       0.72     0.00      read_events
    217      0.10      19.52     0.01      replace_acl
      9      0.00      19.63     0.00      replace_dir_acl
      3      0.00       8.45     0.00      replace_dir_inacl
     12      0.00      17.46     0.00      replace_inacl
     25      0.03      45.14     0.00      set_256K_switch
   6953      0.27       1.67     0.07      set_alarm_timer
     45      0.01       9.88     0.44      set_bc
   5583      0.98       7.57     0.03      set_bc_seg
      2      0.00      10.74     0.00      set_copysw
     37      0.00       0.09     0.00      set_cpu_timer
   1328      0.39      12.82     0.05      set_max_length_seg
     75      0.02      10.08     0.00      set_safety_sw
    157      0.03       8.80     0.04      set_safety_sw_seg
     52      0.13     109.05     2.83      star_
    186      0.42      98.13     2.92      star_dir_list_
    856      0.32      16.36     0.46      status_
     13      0.00       9.75     0.00      status_for_backup
   3030      0.86      12.23     0.12      status_long
   2980      0.53       7.66     0.20      status_minf
   1031      0.12       4.84     0.04      status_mins
     25      0.00       0.63     0.08      stop_process
     13      0.00       5.65     0.00      terminate_file
     52      0.00       1.80     0.02      terminate_name
   5062      0.23       1.97     0.08      terminate_noname
     45      0.00       3.54     0.07      terminate_seg
     26      0.01      18.98     0.00      truncate_file
   2581      0.77      12.86     0.09      truncate_seg
    257      0.01       2.04     0.00      try_to_unlock_lock
    116      0.01       3.65     0.07      tty_abort
     50      0.03      22.60     0.60      tty_attach
     35      0.00       0.96     0.00      tty_detach
     31      0.00       1.19     0.00      tty_detach_new_proc
   4233      0.26       2.62     0.04      tty_get_line
    129      0.00       0.99     0.00      tty_event
      9      0.00       1.08     0.11      tty_get_name
      5      0.00       7.37     0.40      tty_index
   8709      0.81       4.00     0.03      tty_order
   7338      0.51       3.01     0.01      tty_read
   5262      0.20       1.62     0.00      tty_read_with_mark
   3528      0.25       3.09     0.02      tty_read_echoed
   1049      0.02       0.96     0.00      tty_state
  17123      2.74       6.91     0.02      tty_write
   4228      0.55       5.65     0.02      tty_write_whole_string
      7      0.00       0.24     0.00      validate_processid
   2963      0.06       0.83     0.09      wakeup
```

Name: meter_rcp

SYNTAX AS A COMMAND

```
meter_rcp {-control_args}
```

FUNCTION

prints information about devices controlled by the resource control package (RCP).

CONTROL ARGUMENTS

-all, -a
: displays meters for both locks and devices.

-device STR, -dv STR
: displays meters only for the device named STR. This control argument cannot be used with the -lock or -type control arguments.

-lock
: displays only meters for locks. The default is to display meter information only for devices and not for locks.

-long, -lg
: displays additional information about devices/locks (as selected by the other control arguments) that is not otherwise printed (e.g., for -type, an assignment histogram; for -lock, four lines of totals).

-report_reset, -rr
: generates a full report and then performs the reset operation.

-reset, -rs
: resets the metering interval for the invoking process so that the interval begins at the last call with -reset specified. If -reset has never been given in a process, it is equivalent to having been specified at system initialization time.

-type STR, -tp STR
: displays meters only for devices of the type specified by STR. This control argument cannot be used with the -lock or -device control arguments. STR can be tape, disk, console, printer, punch, reader, or special.

NOTES

If the meter_rcp command is given with no control arguments, it prints information for all devices only (no locks).

The following is a brief description of the variables printed if the -lock control argument is specified.

% time locked
: is the percentage of time spent locked.

% time waiting
: is the percentage of time waiting for locks.

% number of waits
: is the percentage of the number of attempts to lock that required a wait.

If the -type control argument is specified, the following variables are printed for that type only. This same information is printed for all device types if no control argument is given, and for the named device if the -device control argument is given.

Total assignments
: is the number of times the device has been assigned during the metering interval.

Total errors
: is the number of I/O transfer errors for the device during the metering interval.

Total time assigned
: is the time (in hours, minutes, and seconds) the device has been assigned during the metering interval.

% time assigned
: is the percentage of the metering interval that the device has been assigned.

EXAMPLES

The following is an example of the information printed if the meter_rcp command is specified with the -lock control argument.

```
Total time metered  =   1 hours, 29 minutes, 34 seconds

Lock meters for rcp_com_seg:
% time locked       =   0 %
% time waiting      =   0 %
% number of waits   =   0 %

Lock meters for rcp_data:
% time locked       =   0 %
% time waiting      =   0 %
% number of waits   =   2 %
```

The following shows a sample result of invoking the meter_rcp command with the -type control argument specifying tape.

```
Total time metered  =    0 hours, 15 minutes, 29 seconds

Meters for device type tape_drive:
Meters for tapa_06
          Total assignments   =    1
          Total errors        =    0
          Total time assigned =    0 hours, 13 minutes, 53 seconds
          % time assigned     =   89 %
Meters for tapa_09
          Total assignments   =    1
          Total errors        =    9
          Total time assigned =    0 hours,  1 minutes, 32 seconds
          % time assigned     =    9 %
```

Finally, if the command:

```
meter_rcp -device dska_05
```

is given, the following information is printed. Giving the command with no control arguments produces a report that would include both this and the above example's information, as well as the same type of information for all other RCP devices.

```
Total time metered  =    0 hours, 5 minutes, 37 seconds
Meters for dska_05
     Total assignments   =   1
     Total errors        =   0
     Total time assigned =   0 hours, 5 minutes, 37 seconds
     % time assigned     =  99 %
```

Name: meter_signal

SYNTAX AS A COMMAND

```
meter_signal {-control_args}
```

FUNCTION

measures the performance of the Multics signalling mechanism. It sets up an environment of condition handlers and stack frames, and then causes a specified number of zerodivide faults to occur. The calendar clock is read before each fault and again as the first operation in the zerodivide condition handler. The difference between these values is recorded and printed on the terminal. The mean and minimum values for all zerodivide faults caused in an invocation are computed.

CONTROL ARGUMENTS

Each control argument must include a decimal value (N) and can be specified in any order.

-nfaults N
: specifies how many zerodivide faults to cause. One zerodivide fault is the default.

-nframes N
: specifies the number of stack frames to be established between the frame containing the zerodivide handler and the frame that causes the fault. The fault occurs in the same frame that established the handler if the value is one; this is the default.

-nhandlers N
: specifies the number of handlers for dummy conditions to be established in each stack frame. Handlers are established for the conditions meter_signal1 through meter_signalN where N is the value specified. The default is that no dummy interrupt handler is established.

-unclaimed N
: specifies that an unclaimed_signal handler should be established instead of the zerodivide handler. The unclaimed_signal handler is established in the Nth frame where N is the value specified. Stack frames are numbered from 1 to p, where p is the number in the -nframe control argument. The default is that no unclaimed_signal handler is established.

EXAMPLES

The command line:

```
meter_signal -nfaults 25 -nframes 5 -nhandlers 2 -unclaimed 3
```

causes 25 faults. Five stack frames are laid down before any faults are caused. Each frame contains handlers for meter_signal_1 and meter_signal_2. An unclaimed_signal handler appears in the third frame.

Name: monitor_cache

SYNTAX AS A COMMAND

```
monitor_cache {cpu_tag} {-control_args}
```

FUNCTION

initiates and controls automatic monitoring of cache memory error data saved during normal fault processing. This command constantly monitors the cache memory error data to provide a warning when error rates become excessive. To compare error rate threshold values, use a per process threshold or the -priv control argument described below. For information pertaining to changing the per-system defaults and setting up per-process values that are different from the per-system defaults, see "Changing the Threshold Values" below. To display the current threshold values, use the display_cache_threshold command.

ARGUMENTS

cpu_tag
: identifies the tag(s) of the CPU(s) whose cache error rates should be monitored. If no CPUs are identified, all possible CPUs will be monitored.

CONTROL ARGUMENTS

-brief, -bf
: suppresses the "CPU below cache error threshold" message when the error rate is within the specified threshold limits. This does not suppress the warning when the error rate is above the threshold values. This is the default.

-long, -lg
: emits a "CPU below cache error threshold" message on the user_output I/O switch.

-cpu cpu_list
: an alternate method of specifying the list of the tags of the CPUs whose cache error rates should be monitored.

-priv
: causes warning messages of "Cache above error threshold for CPU" to be written into the syserr_log and the bootload console with an audible alarm. This control argument sends the polling message described under -long above to the syserr_log instead of the user_output I/O switch. You must have re access to the hphcs_ gate to use this control argument.

-stop, -sp
: stops monitoring for the CPUs specified in the cpu_tag argument or the -cpu control argument. If no CPU tags have been specified, then monitoring for all CPUs is terminated.

-start, -sr
    resumes monitoring for the CPUs specified in the cpu_tag argument or the -cpu control argument. Monitoring will continue with the next scheduled cycle.

-time N, -tm N
    specifies the monitoring interval in minutes, where N is a decimal integer. The default time is 15 minutes.

ACCESS REQUIRED

You must have re access to the phcs_ gate to use this command.

CHANGING THE THRESHOLD VALUES

The per-system and per-process default threshold values are defined in >system_library_tools>cache_threshold_defaults. This segment is created by a cds source segment of the same name. To change the system default values, you must change this source segment, recompile it, and install the generated object in >tools. A per-process threshold that is different from the per-system values may be created by performing the same operations, except that the per-process threshold should be found in the process' object search rules before >tools or the threshold may be specifically initiated.

NOTES

The monitor interval is closely associated with the threshold values in that the threshold is specified in terms of an acceptable error rate N in X time. The default threshold values are expressed as a per_hour_error rate.

Name: monitor_sys_log, msl

SYNTAX AS A COMMAND

```
msl {log_selector} {-control_args}
```

FUNCTION

monitors activity in system logs. Contents of the selected logs are periodically examined, and any new messages are printed on the terminal or passed as arguments to a specified command line. Control arguments determine what action the command performs, and also supply message selection and formatting options.

There are three types of control arguments: log selection, action, and monitoring control arguments. Only one action control argument may be specified in any given invocation of the command. For all actions (except -status), a log must be specified; for some actions (-update, -on, -off, -remove, -status), -all may be specified to select the log, indicating that the action applies to all logs currently being monitored. All other control arguments specify monitoring operations: message selection, message processing, message format. A separate set of monitoring options is kept for each log being monitored. A specific log may be monitored more than once, by specifying -add to establish separate monitors; this is useful when different sets of monitoring options are desired to process messages from the same log differently.

ARGUMENTS

log_selector
: is the pathname of a log to be monitored. The pathname must specify the first segment in a log. This argument is incompatible with any of the log selection control arguments.

  If a log is selected, either by pathname or one of the control arguments below, and no "action" is specified, monitoring is started or updated, as appropriate (see -update, below). To replace the monitor information for a log, or start an independent monitor with different parameters, -replace or -add must be specified. For all actions except -status, either a log pathname or a log selection control argument must be specified.

CONTROL ARGUMENTS FOR LOG SELECTION

-admin
: specifies that the admin log is to be examined. The admin log is called "admin_log", and is located in the >sc1>as_logs directory. This argument is incompatible with any of the other log selection control arguments, or an explicit log pathname.

-all, -a
: specifies that the monitor information for all logs is to be changed. This argument is incompatible with any of the other log selection control arguments, or an explicit log pathname.

-answering_service, -as
: specifies that the answering service log is to be examined. The answering service log family is called "log", and is located in the >sc1>as_logs directory. This argument is incompatible with any of the other log selection control arguments, or an explicit log pathname.

-dm_system, -dms

specifies that the data management system log for the process's current AIM authorization is to be examined. The data management log family is called "dm_system_log", and its location depends on the AIM access class of the log. This argument is incompatible with any of the other log selection control arguments, or an explicit log pathname. Reading the log requires access to the dm_admin_ gate.

-mc_log log_name, -mcl log_name

specifies that the message coordinator (daemon) log named LOG_NAME is to be examined. All message coordinator logs are located in the >sc1>as_logs directory; their names depend on the daemon to which they belong. This argument is incompatible with any of the other log selection control arguments, or an explicit log pathname.

-number N, -nb N

specifies that the monitor information for log number N is to be changed. The log numbers are listed in the output from "monitor_sys_log -status". This argument is incompatible with any of the other log selection control arguments, -all, or an explicit log pathname.

-syserr

specifies that the syserr log is to be examined. The syserr log family is named "syserr_log". The first segment in the family is >sl1>syserr_log; there may be a history segment in >sl1, and older history segments are in the directory >sc1>syserr_log. This argument is incompatible with any of the other log selection control arguments, or an explicit log pathname.

CONTROL ARGUMENTS FOR ACTION TO BE PERFORMED

-add

adds the specified log to the list being monitored, with the parameters specified by the other control arguments. This is incompatible with -all and -number, and any other "action" control argument.

-off

turns monitoring off for the specified log(s), remembering the monitoring parameters. Monitoring must already have been started on the specified log. This is incompatible with the other "action" control arguments. No monitoring parameters may be specified with this control argument.

-on

turns monitoring back on for the specified log(s), reversing the effect of -off. Monitoring must already have been started on the specified log. This is incompatible with the other "action" control arguments. No monitoring parameters may be specified with this control argument.

-remove, -rm
: removes the specified log(s) from the list being monitored. Monitoring must already have been started on the specified log. This is incompatible with the other "action" control arguments. No monitoring parameters may be specified with this control argument.

-replace, -rp
: replaces all the monitoring parameters for the specified log(s) with those specified by the other control arguments. The log(s) must already be being monitored; it is an error to specify a log pathname or selection control argument identifying a log not being monitored already. This is incompatible with -all and any other "action" control argument.

-status, -st
: displays the status of monitoring for the specified log(s), or for all logs currently being monitored if none is specified explicitly. This is incompatible with the other "action" control arguments. No monitoring parameters may be specified with this control argument.

-update, -ud
: if the specified log is not already being monitored, monitoring is started for the specified log. Otherwise, the monitoring parameters for the specified log are updated by the other control arguments specified, as if those control arguments had been specified at the end of the command line that initially started the monitoring. This is incompatible with the other "action" control arguments. This is the default.

CONTROL ARGUMENTS FOR MESSAGE PROCESSING

-call COMMAND, -cl COMMAND
: specifies a command line to be executed each time a message is selected from the specified log(s). If the command is the null string (""), command line processing is turned off, and new entries are printed on the specified switch, instead. The command line receives the following parameters:

```
1) log prefix string
2) date-time of message
3) sequence number of message
4) severity of message
5) text of message
6) data class of message
7) expanded text of message
```

Parameters 6 and 7 are only supplied if the message has binary data, and -expand was specified.

-time N, -tm N
: specifies the monitoring interval, in seconds; the specified log(s) will be sampled once every monitoring interval. If the specified interval is zero, periodic monitoring is turned off. The default is ten seconds.

CONTROL ARGUMENTS FOR MESSAGE SELECTION

-all_data_classes, -adc
: specifies that all messages, regardless of data class, are to be processed. This control argument cancels the effect of any preceding -match_data_class or exclude_data_class control argument. This is the default.

-all_severities, -asv
: specifies that messages of all severities are printed. This cancels the effect of any previous -severity control arguments. This is the default.

-all_text, atxt
: specifies that all messages, regardless of text contents, are to be processed. This control argument cancels the effect of any preceding -match or -exclude control argument. This is the default.

-exclude_data_class CLASS, -exdc CLASS
: specifies that no messages with the specified data class are to be processed.

-exclude STR-1 ... STR-n, -ex STR-1 ... STR-n
: specifies that no message whose text contains one of the specified strings (STR-1 to STR-n) is processed. A string is interpreted either as a text string, which must be an exact substring of the message text, or, if surrounded by slashes, as a regular to match against the message text. See the "Notes on String Matching" section below for details.

-match_data_class CLASS, -mdc CLASS
: specifies that only messages with the specified data class are processed. See "Notes on Data Classes," below.

-match STR-1 ... STR-n
: specifies that only messages whose text contains one of the specified strings (STR-1 to STR-n) are processed. The strings are processed as for -exclude.

-severity SEV-1 ... SEV-n, -sv SEV-1 ... SEV-n
: specifies that only messages of the specified severity (severities) are processed. The severity values (SEV-1 to SEV-n) may either be decimal integers, or ranges consisting of a pair decimal integers separated by a colon ("20:29"). If multiple severities are specified, or the -severity control argument is specified more than once, all messages with any of those severities are printed. A severity value must be between -100 and 100. See the "Notes on Severity Values" below for details.

CONTROL ARGUMENTS FOR MESSAGE EXPANSION

-all_data, -ad
: specifies that all messages with binary data are to be processed. This control argument cancels the effect of any preceding -match_data or -exclude_data control argument. This is the default.

-exclude_data STR-1 ... STR-n, -exd STR-1 ... STR-n
: no messages whose interpreted expanded data contains one of the specified strings (STR-1 to STR-n) is processed. The strings are processed as for -exclude. Note: this control argument merely matches against the textual interpretation of the expanded data; if this interpretation is to be displayed as well, the -interpret control argument must also be specified.

-expand {CLASS-1 ... CLASS-n}
: specifies that binary data is to be expanded and displayed along with the message text for the selected messages. If a data class value (CLASS-1 to CLASS-n is specified, only binary data of the specified classes will be expanded; otherwise, all selected messages with binary data will be expanded. The type of expansion depends on whether the -octal or -interpret control arguments are also specified. See the "Notes on Data Classes" section below for details. By default, no messages are expanded.

-interpret, -int
: specifies that the binary data in expanded messages is to be displayed as interpreted text, by calling the appropriate expand_XXX_msg_ program for the data class of the message. If the -octal control argument is also specified, the binary data is displayed both in interpreted form and as octal data. (Default)

-match_data STR-1 ... STR-n, -md STR-1 ... STR-n
: specifies that only messages whose interpreted expanded data contains one of the specified strings (STR-1 to STR-n) are processed. The strings are processed as for -exclude. Note: this control argument merely matches against the textual interpretation of the expanded data; if this interpretation is to be displayed as well, the -interpret control argument must also be specified.

-no_expand, -nex
: specifies that no messages are to be displayed with binary data expanded. This cancels the effect of any previous -expand control arguments. By default, no messages are expanded.

-octal, -oc
: specifies that the binary data in expanded messages is to be displayed in octal, rather than, or in addition to, the interpreted representation. If both octal and interpreted representations are desired, both the -octal and -interpret control arguments must be supplied.

CONTROL ARGUMENTS FOR MESSAGE FORMAT

-continuation_indent N, -ci N
: specifies that all messages are to be formatted for printing with continuation lines prefixed by N spaces, or, if the keyword "standard" or "std" is used in place of a number, with the continuation lines indented sufficiently to line up under the first character of the text of the message. The value of N must be between zero and fifty. By default, continuation lines are indented three spaces.

-date_format FORMAT_STRING, -dfmt FORMAT_STRING
    specifies a date/time format string (see time_format.gi.info or the *Multics Programmer's Reference Manual*, Order No. AG91) to be used when formatting the date when successive messages are printed with different dates. The date string is printed on a line entirely by itself, preceded by a blank line. If the date format string is blank, no date separators will be printed; this should be used if a -time_format string is specified that includes the date as well. The default date string is "^9999yc-^my-^dm ^da ^za", which prints as (for example) "1984-10-31 Wed est".

    By specifying null strings for date, time, and number formats, the log can be printed and saved, so that it can be compared to another log script later, without spurious mis-compares because the times and sequence numbers do not match.

-duplicates, -dup
    inhibits the printing of "=" messages for messages whose text is the same as the previous message printed. All messages are printed exactly as they appear in the log.

-indent N, -ind N
    specifies that all messages are to be formatted for printing prefixed with N spaces. The value of N must be between zero and fifty. The indentation is printed before any data associated with the message, including the message prefix. By default, there is no indentation.

-line_length N, -ll N
    specifies the line length used when formatting message text and data for printing. The value (N) must be between 25 and 500. By default, it is the line length associated with the user_output I/O switch, or, if none (as for an absentee), it is 132 (for line printer output).

-no_duplicates, -ndup
    prints "==" for messages whose text is the same as the previous message printed. This is the default.

-number_format IOA_STRING, -nfmt IOA_STRING
    specifies an ioa_ string to be used when printing the sequence number for the message. If the string is null, no sequence number is printed with the message. The default is "^7d". (See the *Multics Subroutines and I/O Modules* manual, Order No. AG93 for a description of ioa_ control strings.)

-prefix STRING, -pfx STRING
specifies that all messages are to be formatted with the specified string as a prefix. This prefix appears after the indentation (if any was specified). The prefix must explicitly include trailing spaces, if any are desired to separate the prefix from the message text. The default prefix for a log is the entryname of the log except for logs specified by the log selection control arguments. The default prefix for a log selected by the log selection control arguments are listed below:

Control Argument	prefix
-syserr	"SYSERR: "
-answering_service	"AS: "
-admin	"ADMIN: "
-dm_system	"DMS: "

-time_format FORMAT_STRING, -tfmt FORMAT_STRING
specifies a date/time format string (see time_format.gi.info or the *Multics Programmer's Reference Manual*, Order No. AG91) to be used when formatting the message time portion of the message. If the string is null, no time is printed with the messages. The default time format is "iso_time", which prints as (for example) "23:21:59".

MISCELLANEOUS CONTROL ARGUMENTS

-data_class, -dc
specifies that the data class of each message is to be printed.

-header, -he
specifies that a header is to be printed giving the times and sequence numbers of the first and last messages processed. This is the default.

-no_data_class, -ndc
specifies that the data class of each message is not to be printed. This is the default.

-no_header, -nhe
specifies that no header is to be printed.

-no_process_id, npid
specifies that the process_id stored with each message is not to be printed. This is the default.

-output_switch NAME, -osw NAME
specifies that the messages are to be written on the named I/O switch. The default is user_output.

-procedure NAME, -proc NAME
    specifies that entrypoints in the procedure called NAME are to be used instead of entrypoints in log_read_ to read the log. This is used to read logs protected by inner-ring subsystems, where the inner-ring subsystem provides a replacement log-reading procedure. See "Access Required", below.

-process_id, -pid
    specifies that the process_id stored with each message is to be printed.

NOTES ON STRING MATCHING

The strings specified by -match and -exclude, or by -match_data and -exclude_data, are processed in sequence. An arbitrary number of strings may follow any of those control arguments, and each string will be treated as if it was preceded by another instance of the control argument, except that any string beginning with a hyphen and not immediately following one of the match/exclude control arguments is treated as a new control argument, and no more strings are picked up until the next match/exclude argument.

A string may be either a text string, in which case it is tested simply to see whether it is a substring in the message, or it may be a regular expression, which is matched against the message. A string will be interpreted as a regular expression if it begins and ends with "/" characters.

Each log message is processed against the set of strings, matching its text (or data) to see if it contains the string. There are two simple cases: only match strings, and only exclude strings. In the case of only match strings, any log message that matches any of the strings will be printed. In the case of only exclude strings, a log message will be printed only if it matches none of the strings.

The more complicated case where match and exclude strings are mixed is handled as follows: test the message against each string in turn. If the message matches, and the string is a "-match" string, the "print-this-message" flag is set on. If the message matches, and the string is a "-exclude" string, the flag is set off. Otherwise, the flag is unaffected. The flag's initial value is on if the first string was a "-exclude" string, and off if the first string was a "-match" string.

NOTES ON DATA CLASSES

A data class is a short string (1 to 16 characters) stored with any message that contains binary data, and is used to identify the expander procedure used to expand the data into its interpreted textual form. The data class is specified when the message is placed into the log.

The data class "syserr" identifies an old-style syserr log message. The "syserr binary" code (see syserr_binary_defs.incl.pl1 for a list) is the first word of the data in the message; the remaining words of data are the real binary syserr data.

ACCESS REQUIRED

For all except inner-ring logs, read permission is required on the log segments themselves, and status permission is required on their containing directories. If an access error is encountered searching for older history logs, the search is stopped at that point, and no further history will be available. For the logs selected by control arguments, the control argument descriptions list the standard history directories for the logs.

For inner-ring logs (the data management system log is the only standard inner-ring log), access to the logs is required, as is access to the gate used by the log-reading procedure (see -procedure).

NOTES ON MESSAGE SELECTION

Messages are selected for printing in a series of steps, each of which filters out certain messages according to the control arguments specified. The set of messages at each step is any that were left after the previous step. If a control argument was not specified, then its corresponding step eliminates no messages. Note that the -expand control arguments do NOT select messages, but only affect how their contents are displayed

```
1) -severity
2) -exclude        (eliminate matching messages)
3) -match          (eliminate non-matching messages)
4) -exclude_data   (eliminate matching messages)
5) -match_data     (eliminate non-matching messages)
```

NOTES ON SEVERITY VALUES

Severity values in log messages are used to indicate the importance of the message being logged, in a general way. Most logs use increasing severity to indicate increasing importance, but the actual meaning depends on the log. For the Answering Service and Message Coordinator logs, the severities have the following meanings:

```
0 => Message just logged
1 => Message logged and printed on a console
2 => Message logged and printed on a console with bells
3 => Message logged, printed, and the system crashed
```

For the syserr log, the severities have different meanings:

```
0 => Message logged and printed on syserr console
1 => Message logged, printed, and the system crashed
2 => Message logged, printed, and the process writing the
     message is terminated.
3 => Message logged and printed, and console alarm sounded
4 => Message just logged, or printed if logging mechanism is
     inoperable
5 => Message just logged, or discarded if it can't be logged
```

The severities 20 to 25 are handled just like 0 to 5, but are different to indicate that the originating program was writing an access audit message, rather than an informative message.

NOTES ON INNER-RING LOGS

Some applications create logs in an inner ring that must be read using a special interface. The only standard log to do this is the data management system log, and it is read when you specify -dms, which supplies both the pathname and the procedure name (dm_log_read_). Other applications can provide their own special procedures for log reading, in which case you must give the procedure name, using -proc. A log read using a reader procedure may enforce additional access requirements as well as requiring access to the log itself. In particular, you must have access to the reader procedure.

Name: mos_edac_summary

SYNTAX AS A COMMAND

```
mos_edac_summary {-control_args}
```

FUNCTION

scans the syserr log and summarizes MOS EDAC activity in a brief report.

CONTROL ARGUMENTS

-day_limit N
: sets a threshold of N days that a memory chip can fail before including it in the summary. The maximum value for N is 16.

-for T
: specifies a relative time (such as "1 hour") used to compute the ending time from the starting time.

-from DT, -fm DT
    specifies the date/time to start scanning the log.

-limit N
    sets a threshold of N EDAC errors for a memory chip before including it in the summary.

-mem list
    specifies a list of memories for which information is required (i.e., mem a b c).

-to D
    specifies the date/time to stop scanning the log.

ACCESS REQUIRED

Read permission is required on the log segments themselves and status permission is required on their containing directories.

NOTES

If -from DT is not specified, the scan starts with the earliest message in the syserr log. The ending time may be specified by using -for or -to, but not both. If both are omitted, the scan terminates with the last message in the log. All dates and times must be in a format acceptable to the convert_date_to_binary_ subroutine, described in the *Multics Subroutines and I/O Modules* manual, Order No. AG93.

Name: move_dir_quota

SYNTAX AS A COMMAND

```
move_dir_quota path1 quota_change1 ...  {pathN quota_changeN}
```

FUNCTION

allows a user to move records of directory quota between two directories, one immediately inferior to the other.

ARGUMENTS

pathi
    is the pathname of a directory. The quota change takes place between this directory and its containing directory. A pathi of -wd or -wdir specifies the working directory. The star convention cannot be used.

quota_changei
: is the number of records to be subtracted from the containing directory's directory quota and added to the directory quota on pathi. If this number is negative, records are added to the containing directory's directory quota and subtracted from the directory quota on pathi.

ACCESS REQUIRED

The user must have modify permission on both the directory specified by pathi and its containing directory.

NOTES

After the change, the directory quota must be greater than or equal to the number of records used by directories in pathi unless the change would make the quota zero.

If the change would make the directory quota on pathi zero, there must be no immediately inferior directory with nonzero quota. When the directory quota is changed to zero, the records used and the time-record product for pathi is reflected up to the superior directory.

EXAMPLES

```
   move_dir_quota >udd>Demo>Smith>1_dir 1000
   >udd>Demo>Smith>1_dir>2_dir -50
```

adds 1000 records to the directory quota on >udd>Demo>Smith>1_dir and tracts 1000 records from the directory quota on >udd>Demo>Smith. It then tracts 50 records from the directory quota on >udd>Demo>Smith>1_dir>2_dir and adds 50 records to the directory quota on >udd>Demo>Smith>1_dir.

Name: move_log_segments

SYNTAX AS A COMMAND

```
move_log_segments log-name from-dir to-dir cutoff
```

FUNCTION

moves log segments from one directory to another, updating the previous log information in the log segment headers so that the log tools (print_sys_log, summarize_sys_log) will continue to find the moved log segments. Because it is necessary to update log segment headers, this command must be used to move log segments; using the move command will cause the segments to become unlocatable.

ARGUMENTS

log-name
: is the name of the log whose segments are being moved. This should be the entryname of the first segment in the log (e.g. log, admin_log, dm_system_log, syserr_log); it must not be a pathname.

from-dir
: is the directory in which to look for segments to move.

to-dir
: is the directory to which to move the segments.

cutoff
: is a time value specifying which logs are to be moved. All log segments whose messages are all dated before the cutoff are moved. See the *Multics Programmer's Reference Manual*, Order No. AG91, for a description of valid time string values.

NOTES

Old log segments are identified by a timestamp suffix (YYYYMMDD.HHMMSS), which identifies the time of the latest message in the log segment. This suffix is tested against the cutoff time, and if it is earlier than the cutoff, the log segment is moved.

A history directory must contain at least one segment; even if the one remaining segment is earlier than the cutoff, it will not be moved, in order to keep the chain correct. Thus, a cutoff of the current time can be used to move all but the most recent log segment into another directory.

NOTES ON HISTORY DIRECTORIES

An arbitrary number of history directories can be maintained; for instance, one for the current day's logs, one for the previous week's, and one for the previous month's. This allows logs to be distributed onto various storage media, perhaps placing older logs on a demountable volume, or even on tape.

Because log segments are self-identifying (from the suffix), they can be placed on tape and later retrieved selectively into a history directory; the print_sys_log command will still be able to find them if that history directory is in the chain.

EXAMPLES

To move the answering service logs more than a day old into the standard history directory, use the following command:

```
   move_log_segments log >sc1>as_logs >udd>sa>a>history -1day
```

To trim a log, the date_deleter command can be used on the history directory. Since, however, the DTCM of the log segments in the history directory is the time they were moved there, rather than the actual latest message time in the log, the date_deleter cutoff must be selected to acscount for this.

Name: mpc_data_summary

SYNTAX AS A COMMAND

```
mpc_data_summary {list} {-control_args}
```

FUNCTION

scans the syserr log and summarizes the MPC statistics placed there by poll_mpc.

ARGUMENTS

list
: is a list of MPC controller names for which the data is to be summarized (i.e., mspa mtpb urpa). The MPC controller names must be four characters long, and the first three characters must be msp, mtp, or urp. The default list is of all MPCs found in the log.

CONTROL ARGUMENTS

-all
: reports all MPCs found in the syserr log.

-brief, -bf
: reports only nonzero device statistics.

-expand
: expands each syserr log entry that is used for the summary. This may cause much output.

-extend, -ext
    extends the output file if it exists. The default is to overwrite the file.

-for T
    computes the ending time from the starting time, where T is a relative time (such as 1hour or 1day).

-from DT, fm DT
    starts scanning the log at the date/time given.

-long, lg
    reports all device statistics. This is the default.

-mpc list
    displays MPC error data only.

-output_file {path}, -of {path}
    directs output to the segment specified by path. If path is not given, a default segment is used in the working directory and named mpc_data_summary.output.

-short
    formats output for devices with fewer than 132 columns. The default is based on output file type and can be used to override the file output default.

-to DT
    stops scanning the log at the date/time given.

ACCESS REQUIRED

Read permission is required on the log segments themselves, and status permission is required on their containing directories.

Name: ms_create, mscr

SYNTAX AS A COMMAND

```
mscr paths
```

FUNCTION

creates a queue message segment with a specified name in a specified directory.

ARGUMENTS

paths
    are the pathnames of queues to be created.

ACCESS REQUIRED

The user must have modify and append permission on the directory in which he is creating a queue.

NOTES

If pathi does not have the ms suffix, it is assumed.

If the creation of a queue would introduce a duplication of names within the directory, and if the old queue has only one name, the user is interrogated as to whether he wishes the old queue to be deleted. If the user answers "no", no action is taken. If the old queue has multiple names, the conflicting name is removed and a message to that effect is issued to the user.

The extended access placed on a new queue message segment is:

```
    adros     user who created the queue
    ao        *.SysDaemon.*
```

For more information on extended access, see the ms_set_acl command in this document.

EXAMPLES

```
   mscr special_3 special_2 >daemon_dir_dir>carry_dir>RES.carry
```

creates the queues special_3.ms and special_2.ms in the working directory and creates the queue RES.carry.ms in the directory >daemon_dir_dir>carry_dir.

Name: new_user

SYNTAX AS A COMMAND

```
new_user$new_user
or
new_user$nu
or
new_user$nua
or
new_user$change Person_id item newvalue
or
new_user$cg Person_id item newvalue
or
new_user$cga Person_id item newvalue
```

FUNCTION

adds or modifies entries in the URF and PNT. It is called by master.ec to implement the accounting administrator commands that deal with user registration (e.g., register, change, chalias).

Entry: new_user

FUNCTION

This entry point adds a new person. The dialogue exchanged between the command and the user of the command is detailed in the register command.

Entry: new_user$nu

FUNCTION

This entry point adds a new person but is less verbose in its prompting.

Entry: new_user$nua

FUNCTION

This entry is similar to the new_user$nu entry point but also allows the system administrator to specify an alias, password flags, and AIM attributes for the user.

The dialogue for new_user, new_user$nu, and new_user$nua obtains and checks the following items for user registration:

```
Full name (Last, First I.: title)
Mailing address
Programmer number
Default project
Password
Card input Password
If new_user$nua is called:
   Alias
   Password flags
   AIM authorization
   Default AIM authorization
   Audit flags
```

The commands attempt to generate a site-unique Person_id from the last name, or the administrator may specify the Person_id.

The user is then registered in the URF and PNT and the administrator is asked if there are any more users to be added.

Typing "stop" at any time aborts the registration of the current user.

Entry: new__user$change

FUNCTION

This entry point supports editing of user registration.

Entry: new__user$cg

FUNCTION

This entry point is similar to new_user$change but is less verbose.

Entry: new__user$cga

FUNCTION

This entry is similar to new_user$cg but also allows the changing of user aliases, password flags, and AIM attributes.

ARGUMENTS
For new_user$change, new_user$cg, and new user_$cga.

Person_id
is a Person_id of a registered user. If not specified, the command asks for one.

item
may be any one of the following keywords:

NOTE: The following items marked with an plus-sign (+) can only be changed with the new_user$cga entry point.

addr
User's mailing address

+ alias
User's login alias. An alias can be deleted by using a period (.) as the new value.

+ audit
AIM audit selectivity flags. This keyword is a character string of the form:

```
name1{,name2,...,namen}
```

where namei is the name of an audit flag. The names and their meanings are listed below.

priv_op
: controls auditing of privileged operations performed by the process. A privileged operation is one performed through use of a privileged gate or under previously set AIM privileges. It is recommended that sites interested in auditing turn this flag on for all processes except perhaps the system daemons.

admin_op
: controls auditing of administrative operations performed by the process. This includes such operations as registration of new users or projects. It is recommended that sites interested in auditing should turn this flag on for all processes.

fault
: controls auditing of illegal procedure and access violation faults that can indicate an attempt to access protected data.

small_cc
: controls auditing of covert channel activity that takes place over channels with a potential bandwidth of 1-10 bps.

moderate_cc
: controls auditing of covert channel activity that takes place over channels with a potential bandwidth of 10-100 bps.

<object_type>=<grant_level>/<deny_level>
: controls the auditing of specified operations on specified system objects. The values of <object_type> can be one of the following:

  fsobj
  : specifies that operations to file system objects are to be audited.

  fsattr
  : specifies that operations to file system attributes are to be audited.

  rcp
  : specifies that operations to objects controlled by the Resource Control Package are to be audited.

  admin
  : specifies that operations to administrative objects (e.g., the PNT) are to be audited.

  special
  : specifies that operations to special objects are to be audited. (Currently, the only special objects are processes.)

  other
  : specifies that operations to objects (e.g., mailboxes) controlled by ring 1 security related subsystems are to be audited.

The values that can be assigned to <grant_level> and <deny_level> are listed below.

N
: specifies that no auditing is to take place.

MA
: specifies that"modify access" operations are to be audited. Operations are audited that attempt to change the access attributes of the object.

M
: specifies that "modify" operations are to be audited. Operations are audited that attempt to change the object or the attributes of the object. This level of auditing includes the "modify access" operations.

R
: specifies that "read" operations are to be audited. Operations are audited that return information about the contents of the object or its attributes/properties. This level of auditing includes the "modify" and "modify access" operations.

The <grant_type>/<deny_type> values are a matched pair. The <grant_type> value specifies auditing of successful operations. The <deny_type> value specifies auditing of unsuccessful operations. For example, the audit flag "fsobj=N/M" specifies that there is to be no monitoring of successful operations on file system objects; however, all unsuccessful modify operations on file system objects will be audited.

Please note that modify access operations cannot be associated with file system objects (fsobj). Instead, modify access operations can be specified for file system attributes (fsattr).

Additional information on auditing, including a more detailed description of the operations that are audited on each object type, can be found in the *Multics System Administration Procedures* manual, Order No. AK50.

+ auth
: AIM authorization is the authorization to be assigned to Person_id. The value for auth can be a range of values in the format "min_auth:max_auth," in which case the new user is eligible to use any of the authorizations within the specified range. Alternatively, the value for auth can be specified as a single value. In this case, the system interprets the specified value as a maximum authorization value and the minimum authorization value is assumed to be system_low. Use the print_auth_names command for a list of valid authorization values.

+ dfauth
: default AIM authorization

+ flags
    The password flags are:

    password
        user has a login password

    network_pw
        user has a network password

    trap
        attempts to log in will be logged

    lock
        attempts to log in will be refused

    change
        user can change passwords, default authorization, and default project

    must_change
        user must change login password before logging in

    generate
        user must use -generate_password to change password.

    time_lock=TIME
        password is locked until TIME.

    operator
        user can use the sign_on command to sign on as an operator.

name
    full name (Last First I.: title)

notes
    a field for administrator comments.

npass (Formerly cpass)
    network password (also used as card input password)

proj
    default project

pass
    login password

progn
    programmer number

newvalue
    is the new value as a single argument (i.e., enclosed in quotes if it contains

blanks). This argument can only be given if item is given. If not specified, the command prompts with the old value and waits for a response. If the new value is an empty line, the old value remains unchanged. (The argument may not be specified at command level when changing a user's password.)

NOTES

Changes are made to both the URF and PNT.

A password may consist of from one through eight ASCII printing characters including backspace, but excluding space and semicolon. "HELP", "help", "quit", and "?" are interpreted uniquely by the password processor and are therefore unacceptable as password specifications for an interactive login. Entering "quit" terminates the login attempt, while "HELP", "help", or "?" results in an explanatory message and repeat of the password prompt.

Name: patch_firmware

SYNTAX AS A COMMAND

```
patch_firmware path mem addr word1...word2...wordi
```

FUNCTION

patches a segment containing an image of a firmware module for an MPC.

ARGUMENTS

path
    is the pathname of the segment containing the firmware.

mem
    is the memory overlay to patch. This argument can be cs to patch the control store overlay, or rw to patch the read/write memory overlay.

addr
    is the starting address to patch, in hexadecimal.

wordi
    is a new MPC word, in hexadecimal. All wordi arguments must be in the range 0-FFFF. At least one wordi argument must be specified. Up to 16 words can be patched with one patch_firmware command.

NOTES

This command displays the old and new contents of each firmware word patched, as well as the checksum, before the patch is made. You are then asked whether the patch is correct. The patch is not made unless you answer yes.

You can retrieve firmware modules from the IFAD tape using the load_tandd_library command (described in the *Multics Online Test and Diagnostics Manual*, Order No. AU77). Normally, firmware modules are kept in the sequential file >system_library_tandd>tandd_deckfile.

If the firmware segment to be dumped is for an MSP800 device adapter unit (DAU), you must still specify the "cs" or "rw", which does not affect the output because the DAU is defined as having only one memory and no control store). In addition, A DAU word is defined as containing one eight-bit byte as opposed to the MPC's two eight-bit byte format. Even though the DAU memory is one contiguous memory, the firmware is still broken into two halves: control store (the first half) and read/write (the second half).

Name: pdt_copy

SYNTAX AS A COMMAND

```
pdt_copy sat_dir pdt_dir admin_dir
```

FUNCTION

is used by the crank in master.ec to copy the system copies of the SAT and all the PDTs into an administrative directory.

ARGUMENTS

sat_dir
: is the directory containing the system copy of the SAT (normally >scl).

pdt_dir
: is the directory containing the system copies of the PDTs (normally >scl>pdt).

admin_dir
: is the administrative directory into which the SAT and PDTs should be copied (normally >udd>SysAdmin>admin>safe_pdts).

NOTES

The copies serve as both a backup of the system copies and a summary of the continually changing usage figures, from which accounting programs can compute month-to-date charges.

-stop, -sp
: stops polling for the FNPs specified with the fnp_list argument. If no FNPs have been specified, polling of all FNPs is stopped. Polling continues to be scheduled periodically, even though no FNPs are being polled.

-start, -sr
: resumes polling for the FNPs specified with the fnp_list argument. If no FNPs have been specified, polling of all FNPs is resumed. Note that the next polling does not occur immediately; it is performed during the next scheduled polling cycle.

This page intentionally left blank.

The command attempts to copy the PDT of every active project in the SAT. If unable to copy a particular PDT, it prints an error message and continues.

Name: poll_fnp

SYNTAX AS A COMMAND

```
poll_fnp {fnp_list} {-control_args}
```

FUNCTION

initiates and controls automatic polling of FNPs. Polling consists of reading error statistics from the FNP memory and logging them in either the syserr log or a file. This command sets up timers and event call handlers within the process. Once initiated, FNP polling is performed periodically, independent of whatever else is going on in the process. This command is normally used by the initializer or a daemon.

ARGUMENTS

fnp_list
: is a list of the FNP names to be polled. If no names are listed, all FNPs are polled.

CONTROL ARGUMENTS

-log
: writes statistical information to the syserr log. This is the default. Access to the hphcs_ gate is required.

-output_file path, -of path
: writes statistical information to the segment specified by path. This control argument can be used in conjunction with -log.

-time N, -tm N
: specifies the polling interval in minutes. The default polling interval is 15 minutes.

-debug, db
: prints extra debugging information each time polling is performed.

The following control arguments modify the polling already in process and cannot be used on the initial invocation of the poll_fnp command.

-finish
  schedules the last polling cycle immediately. Once this cycle completes, polling is disabled, and a new poll_fnp command is required to start it again. To stop polling without performing one last cycle, use both -stop and -finish.

NOTES

If polling of an individual FNP fails three consecutive times, polling of that FNP is stopped. If three consecutive scheduled polling cycles are missed because a previous cycle did not complete, an automatic finish operation is performed, and no further cycles are scheduled.

Polling of FNPs has no effect on the users of devices connected to the FNP.

Name: poll_mos_memory

SYNTAX AS A COMMAND

```
poll_mos_memory
```

FUNCTION

reads the maintenance register of each memory on the system and prints information about these registers on your terminal. In addition, if the maintenance register indicates that an EDAC error has occurred, it is logged in the syserr log.

ACCESS REQUIRED

You must have re access to phcs_ to use this command.

NOTES

This command should be used with care on systems that have core memories. Unless the TEST/NORMAL switch on the maintenance panel of the memory (not controller) is set to TEST, the result of reading the maintenance register is undefined, and spurious errors may be logged.

Name: poll_mpc

SYNTAX AS A COMMAND

```
poll_mpc {mpc_list} {-control_args}
```

FUNCTION

initiates and controls automatic polling of MPCs. Polling consists of reading statistics on device usage and errors from the MPC memory and logging it in either the syserr log or a file. If an error condition is detected, a message is entered in the syserr_log with a code of 3. This sounds the bootload console alarm and prints the message in the bootload console log on a normally configured system. This command sets up timers and event call handlers within the process. Once initiated, MPC polling is performed periodically, independent of whatever else is going on in the process. This command is used by the initializer or a daemon; Utility.SysDaemon is recommended.

ARGUMENTS

-debug, -db
prints extra debugging information each time polling is performed.

mpc_list
is a list of the names of the tape or disk MPCs to be polled. If no names are listed, all tape and disk controllers are polled.

CONTROL ARGUMENTS

-log
writes statistical information to the syserr log. Access to the hphcs_ gate is required. This is the default.

-output_file path, -of path
writes statistical information to the segment specified by path. This report is the same as the one generated by the -stat control argument of the dump_mpc command. This control argument can be used in conjunction with -log.

-time N, -tm N
specifies the polling interval in minutes. The default polling interval is 15 minutes.

The following control arguments modify the polling already in process and cannot be used on the initial invocation of the poll_mpc command.

-finish
schedules the last polling cycle immediately. Once this cycle completes, polling is disabled, and a new poll_mpc command is required to start it again. To stop polling without performing one last cycle, use both -stop and -finish.

-start, -sr
: resumes polling for the MPCs specified with the mpc_list argument. If no MPCs have been specified, polling of all MPCs is resumed. Note that the next polling does not occur immediately; it is performed during the next scheduled polling cycle.

-stop, -sp
: stops polling for the MPCs specified with the mpc_list argument. If no MPCs have been specified, polling of all MPCs is stopped. Polling continues to be scheduled periodically, even though no MPCs are being polled.

NOTES

If polling of an individual MPC fails three consecutive times, either because it cannot be attached or because of I/O errors, polling of that MPC is stopped. If three consecutive scheduled polling cycles are missed because a previous cycle did not complete, an automatic finish operation is performed, and no further cycles are scheduled.

Polling of MPCs has no effect on the users of devices connected to the MPC.

Name: post_purge_meters, ppm

SYNTAX AS A COMMAND

```
ppm {-control_arg}
```

FUNCTION

displays information collected at post purge time, if post purging is enabled. The print_tuning_parameters and work_class_meters commands (described later in this section) are used to determine which work classes, if any, are being post purged.

CONTROL ARGUMENTS

-reset, -rs
: resets the metering interval for the invoking process so that the interval begins at the last call with -reset specified. If -reset has never been given in a process, it is equivalent to having been specified at system initialization time.

-report_reset, -rr
: generates a full report and then performs the reset operation.

ACCESS REQUIRED

This command requires access to phcs_ or metering_gate_.

NOTES

If the post_purge_meters command is given with no control argument, it prints a full report.

The following is a brief description of each of the variables printed out by the post_purge_meters command.

Post purge time
: is the average CPU time per post purge call.

Ave list size
: is the average number of page fault entries found in the per-process page trace list at post purge time.

Ave working set
: is the average estimated working set. The current estimated working set for each process is computed by the following formula:

```
working set = working_set_factor * raw_working_set
+ working_set_addend
```

The raw working set is estimated by page control at post purge time.

Working set factor
: is the current value of the wsf tuning parameter, and can be changed by the change_tuning_parameters command. Increasing the value tends to reduce page thrashing, but may increase multiprogramming idle. Decreasing the value has the opposite effects.

Working set addend
: is the current value of the wsa tuning parameter, and can be changed by the change_tuning_parameters command. Increasing and decreasing this value has the same effects as noted above.

Thrashing percentage
: is the percentage of page faults that were taken on pages faulted earlier in quantum.

Ave post in main memory
: is the average number of entries in the trace list for which the page was still in main memory at post purge time, and the ratio of incore pages to faulted pages expressed as a percentage.

EXAMPLES

The following is an example of the information printed when the post_purge_meters command is invoked with no control argument.

```
Total metering time      12:43:11

Post purge time            3.14 msec. (0.46% of system)
Ave list size             39.72 entries
Ave working set           15.14 pages
Working set factor         0.50
Working set addend            0
Thrashing percentage      12.07 %
Ave post in core          26.16        (65.85 %)
```

Name: print_apt_entry, pae

SYNTAX AS A COMMAND

```
pae {identifiers} {-control_args}
```

SYNTAX AS AN ACTIVE FUNCTION

```
[pae APTE_item]
```

FUNCTION

prints one or more Active Process Table Entries (APTEs). Each APTE can be printed in octal form, interpreted form, or both. As an active function, it returns individual items from the APTE.

ARGUMENTS

identifiers
: can be User_ids, channel names, or process IDs. The three types of identifier are distinguished from one another by their format (see "Notes" below). They can be preceded by control arguments to eliminate any ambiguity (see "Control Arguments for Entry Selection").

APTE_item
: can be the process directory pathname or process termination event channel.

CONTROL ARGUMENTS FOR ENTRY SELECTION

-absentee, -as
: selects absentee users.

-all, -a
selects all three process types. (Default)

-channel CHN, -chn CHN
selects the user logged in over channel CHN.

-daemon, -dmn
selects daemon users.

-interactive, -ia
selects interactive users.

-process_id PID, -pid PID
selects the specified process.

-user User_id
selects the given user.

CONTROL ARGUMENTS FOR OUTPUT FORMAT

-brief_display
prints the heading and only the first line of the interpretation produced by -display.

-display
prints a header and a four-line interpretation of some of the variables in the APTE (see "Notes on Output Format"). (Default)

-dump
dumps the selected APTE(s) in octal.

-long, -lg
causes octal dumps (when selected) to be eight words per line. (Default)

-no_display
prints the heading, but none of the interpretation.

-no_dump
eliminates octal dump of APTEs. (Default)

-process_dir, -pd
prints or returns the process directory pathname (see "Notes").

-short, -sh
causes octal dumps (when selected) to be four words per line.

-term_channel, -tchn
prints or returns the process termination event channel (see "Notes").

ACCESS REQUIRED

You need read access to the three user tables (absentee_user_table, answer_table, and daemon_user_table) in >sc1, as well as access to the gate metering_ring_zero_peek_.

NOTES

If you give no process selection arguments, the APTE of the current process is printed.

The type of an identifier not preceded by a control argument is determined as follows: if it contains only octal digits, it is a process ID; if it contains any uppercase letters, it is a User_id; otherwise, it is a channel name.

Channel names and User_ids can be star names. User_ids are of the form Person.Project.tag. You can omit any of the three components, along with any trailing periods. Omitted components are treated as if they had been "*". The presence of a tag component restricts the search to the corresponding user table for that user only.

A channel is a communications channel for an interactive process (e.g., a.h017), an absentee slot number for an absentee process (e.g., abs3), or a message coordinator source name for a daemon process (e.g., bk, prta).

If you supply a process ID of six digits or less, it is assumed to be the left half of a process ID, which is the octal offset of the APTE.

When you give mutually exclusive control arguments, the last one on the line from each set is used. This allows you to define your own defaults by an abbreviation and to override them conveniently by using opposing control arguments on a command line. The control arguments -interactive, -absentee, and -daemon are not mutually exclusive, but are mutually exclusive with -all.

NOTES ON OUTPUT FORMAT

This command prints, for each APTE selected, a heading line, an optional interpretation of one to five lines, and an optional octal dump. The contents of the heading and interpretation are described here. Fields enclosed in square brackets ([]) are omitted if they contain null values, such as zero.

The heading

```
Pers.Proj.tag <channel> at <offset> in tc_data >pdd><pdir>
```

gives the User_id, communications channel, octal offset of the APTE, and process directory name. This line is always printed.

Line 1:

```
[F:<flags> ][E:<event> ]PID:<proc_id> TRM:<term channel>
```

gives the flag word (omitted if zero or if line four, containing flag names, is printed), the event word (omitted if zero), the process ID, and the event channel over which this process's termination is signaled. All of these are in octal. This line is printed unless you select -no_display.

The remaining four lines are printed by default, but are suppressed by -brief_display or -no_display.

Line 2:

```
<state> for <interval> (since <time>[<date>]).
   Usage:  cpu <sec>; vcpu <sec>; pf <n>.
```

gives the process state (blocked, running, etc.) and the time interval since state change and the time of state change; the date is printed only if it is different from the current date. These are followed by the total real and virtual CPU time used, in seconds, and the number of page faults.

Line 3:

```
te/s/i/x: <te> <ts> <ti> <timax>.[<ips name> pending.]
  [Flags: <flag names>.]
```

gives the four scheduling parameters (te, ts, ti, and timax) in seconds of CPU time. These parameters are described in the *Multics System Maintenance Procedures* manual, Order No. AM81; briefly, they are time eligible, time since scheduled, min (time since interaction, timax), and the upper limit on ti. Following these parameters, any ips signals pending in the process are printed, as well as the names of any flags that are on (except for the "firstsw" flag, which is only printed if it is off, an indication that the process has never run).

Line 4:

```
[Alarm in <interval> (at <time>[<date>][(<interval> after block)]).
  [CPU monitor in <interval> vcpu sec.]
```

is omitted unless the process has an alarm timer or a CPU monitor set. If an alarm timer is set, its time (and date, if different from the current date) are printed. If the process is blocked, the interval between the time of blocking and the alarm timer is printed. If a CPU monitor is set, the amount of virtual CPU time remaining until it goes off is printed.

Line 5:

```
   IPC R-Offset: <r-offset>, R-Factor <r-factor>
```

gives the R-Offset and R-Factor preprocess values used to create IPC event channel names.

*

Name: print_devices

SYNTAX AS A COMMAND

```
print_devices {-control_args}
```

FUNCTION

prints a list of devices for each request type handled by the I/O daemon. Also, the driver access name and driver authorization (if any) for each request type are printed. An asterisk (*) immediately preceding a device name indicates that the associated request type is the default for the device.

CONTROL ARGUMENTS

-access_name STR, -an STR
: lists only devices for those request types having a driver access name of STR (STR should be of the form Person_id.Project_id).

-brief, -bf
: suppresses printing of a heading line.

-dir path
: specifies the absolute pathname of the directory containing the iod_working_tables segment. If not given, the directory >ddd>idd is assumed.

-request_type STR, -rqt STR
: lists only devices for the request type specified by STR (e.g., printer, punch).

Name: print_disk

SYNTAX AS A COMMAND

```
print_disk
```

FUNCTION

prints a report from the information stored in the projfile by charge_disk showing each project's disk usage. It is called by the master.ec segment.

NOTES

One line is printed for each project.

The segment, projfile, is assumed to be present in the working directory and is an implicit input to print_disk.

Name: print_iod_tables

SYNTAX AS A COMMAND

```
print_iod_tables path
```

FUNCTION

displays the contents of an object segment produced by the iod_tables_compiler command. The format of the output corresponds exactly to the source language accepted by the iod_tables_compiler command. In fact, if the output of the print_iod_tables command is directed to a segment, the resulting segment can be translated by the iod_tables_compiler command.

ARGUMENTS

path
    is the relative or absolute pathname of the object segment to be displayed.

This page intentionally left blank.

This page intentionally left blank.

Name: print_line_ids

SYNTAX AS A COMMAND

```
print_line_ids {-control_args}
```

FUNCTION

prints a list of logical line_ids and their associated communications channel from data in the iod_working_tables segment.

CONTROL ARGUMENTS

-brief, -bf
    suppresses printing of a heading line.

-dir path
    specifies the absolute pathname of the directory containing the iod_working_tables segment. If not given, the directory >ddd>idd is assumed.

Name: print_meters

SYNTAX AS A COMMAND

```
print_meters {path}
```

FUNCTION

prepares a system performance report from a data segment containing system metering data. The metering data is accumulated by as_meter_ in the stat_seg segment; usually this is copied into a temporary segment by copy_as_meters before report preparation.

ARGUMENTS

path
    is the pathname of the data segment.

NOTES

The report contains long lines and is designed for dprinting (use of the file_output command is recommended). The report consists of a paragraph for each bootload. A header giving the system name, bootload time, and time of last shutdown or crash, is followed by one line per sample. Samples are normally taken at each accounting update (every 15 minutes) and at startup and shutdown time. The lines contain the following columns:

Time	time of sample
Units	load units
CP	number of CPUs
Kmem	number of 1024K word main memory pages
MPD	number of million (M) word paging device (PD) blocks
Meter time	time since bootload
CPU time	CPU time available during sample interval
Avg queue	average queue length (number of processes in queue)
Response	average response time (seconds)
Idle	total idle time in sample interval
Zero idle	total zero idle time in sample interval
Avg eligible	average number of eligible processes
K mu	memory units charged in thousands

Name: print_pdt

SYNTAX AS A COMMAND

```
print_pdt path {Person_ids} {-control_args}
```

FUNCTION

The print_pdt command prints a listing of a project definition table (PDT).

ARGUMENTS

path
: is the pathname of the PDT segment to be printed. If the pdt suffix is not given, it is assumed. If the pathname given does not start with a greater-than or less-than character, it is interpreted as a project name and the PDT in the directory containing PDTs (>sc1>pdt) is used.

Person_ids
: are the Person_ids about whom information is desired. If this argument is omitted, information is printed for all users listed in the PDT.

CONTROL ARGUMENTS

-brief, -bf
: prints small amount of information about each user.

-long, -lg
: prints all data items in the PDT.

-no_header, -nhe
: suppresses printing of the header.

-pmf
: prints the PDT in project master file (PMF) format. The file_output command (described in the *Multics Commands and Active Functions* manual, Order No. AG92) can be used to place the printed PDT in a segment for daemon printing or for subsequent use as a PMF (see "Notes" below).

NOTES

If no control arguments are given with this command, all PMF-specifiable attributes and the total amount spent are printed. The user must have read access to the PDT; usually only project administrators have such access. The following command line is recommended to make a PMF from a PDT:

```
  fo Project_id;print_pdt Project_id -pmf;ro
```

See also the proj_usage_report command to get a brief summary of each user's resource consumption and the display_account_status command to obtain the charges accrued to the account.

Name: print_pnt

SYNTAX AS A COMMAND

```
print_pnt {Person_id} {-control_args}
```

FUNCTION

prints the contents of a binary person name table (PNT) in a readable form on the administrator's terminal. Note that, while portions of the user entry in the PNT are stored in encrypted form, any encryption algorithm is susceptible to a sophisticated, computer-assisted code-breaking effort. Therefore the System Administrator should ensure that access to the PNT is as restricted as possible. In general, only the SysAdmin and SysDaemon projects should have access to the PNT.

ARGUMENTS

Person_id
  specifies the person whose PNT entry is to be printed. If this argument is omitted, the entire PNT is printed.

CONTROL ARGUMENTS

-brief
  makes no attempt to access information in the user registration file (URF).

-force_no_password_info
  same as -no_password_info, but additionally causes the rejection of the -password_info control argument for this invocation of print_pnt. This control argument can be used to ensure that limited system administrators cannot obtain password information.

-no_password_info
  suppresses the display of information about good or bad passwords and the date of password change. The rest of the PNT/URF entry is displayed.

-password_info
  displays any information about good or bad passwords and the date of password change with the rest of the PNT/URF entry. (Default)

Name: print_projfile

SYNTAX AS A COMMAND

```
print_projfile {path}
```

FUNCTION

displays the contents of the project registration file.

ARGUMENTS

path
  is the project file to be listed. If omitted, the command prints the segment named projfile in the working directory.

Name: print_reqfile

SYNTAX AS A COMMAND

```
print_reqfile {path}
```

FUNCTION

interprets the requisition file and prints it in a readable form.

ARGUMENTS

path
    is the pathname of the segment to be printed. If omitted, the command prints the named reqfile segment in the working directory.

Name: print_sat

SYNTAX AS A COMMAND

```
print_sat path {Project_id}
```

FUNCTION

prints the contents of a binary system administrator table (SAT) in a readable form on the administrator's terminal.

ARGUMENTS

path
    is the pathname of the sat to be printed. If the entryname portion of path is given without a suffix of sat, the suffix is added (unless the entryname portion is simply "sat").

Project_id
    specifies the project whose SAT entry is to be printed. If this argument is omitted, the entire SAT, including the header, is printed.

EXAMPLES

The command:

```
   print_sat >sc1>sat
```

prints the system copy of the SAT, for example:

```
Maxusers:          75;
Maxunits:          750;
Maxprim:           75;
Uwt:               process_overseer_, 10;
Uwt:               >system_library_tools>iod_overseer_, 10;
Administrator:     *.SysAdmin;
Administrator:     *.SysDaemon;
Attributes:        nopreempt, preempting, brief, vinitproc,
                   vhomedir, nostartup;

projectid:         SysDaemon;
projectdir:        >udd>SysDaemon;
maxprim:           100;
attributes:        primary_line, guaranteed_login, anonymous,
                   nolist, dialok, multip, daemon,
                   v_outer_module;
authorization:     "17,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,
                   c12,c13,c14,c15,c16,c17,c18";
ring:              1,5;
alias:             sd;
group:             System;
grace:             2880
projectid:         SysAdmin;
projectdir:        >udd>SysAdmin
maxprim:           100;
attributes:        primary_line, guaranteed_login,
                   anonymous, dialok, multip, v_outer_module;
audit:             "no_access, ipr_fault, acv_mode, acv_ring,
                   no_wakeup, sys_priv, saa_ops, mseg";
authorization:     "17,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,
                   c12,c13,c14,c15,c16,c17,c18";
ring:              4,5;
alias:             sa;
group:             Admin;
grace:             2880;

projectid:         Operator:
projectdir:        >udd>Operator;
maxprim:           100;
attributes:        primary_line, guaranteed_login,
                   anonymous, dialok, v_outer_module;
ring:              4,5;
alias:             op;
group:             Other;
grace:             2880;

projectid:         HFED;
projectdir:        >udd>HFED;
```

```
maxprim:            100;
attributes:         primary_line, guaranteed_login,
                    anonymous, dialok, v_outer_module;
ring:               4,5;
alias:              hfed;
group:              Other;
grace:              2880;

projectid:          Terminals;
projectdir:         >udd>Terminals;
maxprim:            100;
attributes:         primary_line, guaranteed_login,
                    anonymous, dialok, v_outer_module;
ring:               4,5;
alias:              te;
group:              Other;
grace:              2880;

projectid:          SysMaint;
projectdir:         >udd>SysMaint;
maxprim:            32767;
attributes:         anonymous;
ring:               4,5;
group:              Other;
grace:              2880;

projectid:          Daemon;
projectdir:         >udd>Daemon;
maxprim:            32767;
attributes:         multip, daemon;
authorization:      "17,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,
                    c11,c12,c13,c14,c15,c16,c17,c18";
ring:               4,5;
alias:              d;
group:              System;
grace:              2880;

projectid:          TOLTS;
projectdir:         >udd>TOLTS;
maxprim:            32767;
attributes:         anonymous, multip;
ring:               4,5;
alias:              tolts;
group:              Other;
grace:              2880;

end;
```

Name: print_spooling_tape

SYNTAX AS A COMMAND

```
print_spooling_tape prtdim device {-control_args}
```

FUNCTION

directly attaches a printer and prints the contents of a tape written by the spool driver.

ARGUMENTS

prtdim
: is the literal string "prtdim", which is the name of the standard Multics printer I/O module (DIM).

device
: is the name of the IOM channel for the printer device to use.

CONTROL ARGUMENTS

-number N, -nbr N
: begins printing at N where N is the file number of a file on tape. If it is omitted, printing begins with the first file on the spooling tape.

-debug, -db
: turns on audit trace during printing. The default is debug off.

EXAMPLES

To print a spooling tape, starting with the third file on the tape, using the standard Multics printer I/O module (identified as "prtdim") and the printer (identified as "prta"), the operator types:

```
   print_spooling_tape   prtdim   prta  -nbr 3
```

Then the operator is asked for volume identifiers and spooling limits, as shown below:

Enter volids and optional file limits:

The operator types:

```
   -volid SPOOL1 -files 50
```

giving the volid of the spooling tape to be printed (SPOOL1) and a limit of 50 files to print before printing is stopped. The I/O module, tape_ansi_, determines whether to read the spooling volume at 800 or 1600 bpi density and, from the tape labels, the I/O module determines the tape block size and maximum line-length to be printed.

Next, the operator is requested to mount the first volid and a message is typed on the terminal as follows:

```
   Mounting volume SPOOL1 with no write ring.
   Volume mounted on tape_XX.
```

As each file on the spooling tape is printed, a message appears on the terminal giving the number of the file. This continues until the file limit has been reached or until the entire tape has been processed. The spool driver output looks like the following:

```
    Printing FILE 3
    Printing FILE 4
      .
      .
      .
    Printing FILE 50

    Reached end of data for current fileset.

    Taking current volume down.

    Printer detached.

    Processing of spooling tape ended.

    Spooling file count is 48
    Spooling line count is 1254
```

At this point, printing has finished and the operator can logout the process.

DESCRIPTION OF THE SPOOLING TAPE

The spool driver creates either an 800 or 1600 bpi ANSI standard tape (ASCII) with D-format (variable length) records of a specified printer line length, that are blocked to 8192 characters, unless the interchange option is specified, in which case, the block size is 2048 characters and the density is 800 bpi. Each print request constitutes one ANSI tape file, which is surrounded by ANSI standard tape labels. The exact format of the ANSI tape can be found by referring to Draft Proposed Revision X3L5/419T of the American National Standard Institute's ANSI X3.27-1969, "Magnetic Tape Labels and File Structure for Information Interchange." Each line (logical record) of the request (print file) is preceded by a USA printer carriage control character that directs a printer action before the line is printed. These control characters and the corresponding Multics spool driver slew functions are listed below.

USA Char	Spool Driver Slew_Function	Printer_Action
blank	NL	One line spaced
0	2(NL)	Two lines spaced
-	3(NL)	Three lines spaced
+	CR	Suppress line space
1	FF	Skip to channel 1 (top: line 3, any page)
2	none	Skip to channel 2
3	none	Skip to channel 3
4	none	Skip to channel 4
5	none	Skip to channel 5
6	none	Skip to channel 6
7	bottom inside page	Skip to channel 7 (odd page)
8	bottom inside page	Skip to channel 8 (even page)
9	none	Skip to channel 9
A	none	Skip to channel 10

B	none	Skip to channel 11
C	none	Skip to channel 12

Note: The printer action occurs before a line occurs printed.

Name: print_sys_log, psl

SYNTAX AS A COMMAND

```
psl {log_selector} {-control_args}
```

FUNCTION

prints selected portions of system logs, including the syserr log, answering service log, admin logs, message coordinator (daemon) log, and data management logs. Various control arguments are used to determine which portions of the log are printed, and the format of the output.

ARGUMENTS

log_selector
: is the pathname of a log to be printed, The pathname must specify the first segment in the log. This argument is incompatible with any of the log selection control arguments.

CONTROL ARGUMENTS FOR LOG SELECTION

-admin
: specifies that the admin log is to be printed. The admin commands log is called "admin_log", and is located in the >sc1>as_logs directory. This argument is incompatible with any of the other log selection control arguments, or an explicit log pathname.

-answering_service, -as
: specifies that the answering service log is to be printed. The answering service log is called "log", and is located in the >sc1>as_logs directory. This argument is incompatible with any of the other log selection control arguments, or an explicit log pathname.

-dm_system, -dms
: specifies that the data management system log for the process' current AIM authorization is to be printed. The data management log is called "dm_system_log", and its location depends on the AIM access class of the log. This argument is incompatible with any of the other log selection control arguments, or an explicit log pathname. Reading the log requires access to the dm_admin_ gate.

-mc_log log_name, -mcl log_name
: specifies that the message coordinator (daemon) log named log_name is to be printed. All message coordinator logs are located in the >sc1>as_logs directory; their names depend on the daemon to which they belong. This argument is incompatible with any of the other log selection control arguments, or an explicit log pathname.

-syserr
: specifies that the syserr log is to be printed. The syserr log is named "syserr_log". The first segment in the log is >sl1>syserr_log; there may be a history segment in >sl1, and older history segments are in the directory >sc1>syserr_log. This argument is incompatible with any of the other log selection control arguments, or an explicit log pathname.

CONTROL ARGUMENTS FOR LIMIT SELECTION

-for time, -for number
: specifies a number of messages to print, or a time interval relative to the starting time (specified by -from) in which the messages must be contained. The number of messages is the actual number of messages printed, not the number of messages examined in the log. This is incompatible with -to and -last.

-forward, -fwd
: specifies that the log is to be printed starting with the oldest message selected by other control arguments, and proceed forwards. This is the default.

-from time, -fm time, -from number, -fm number
: specifies that the first message printed is the first message at or after the specified time or sequence number; if -reverse is specified, the first message is the one at or before the specified value. If no -from value is specified, the default is the first message in the log, or the last if -reverse is specified. This is incompatible with -last.

-last number, -lt number, -last time, -lt time
: specifies that only the last number messages, or the messages since time, are to be printed. If a number is specified, it specifies the actual number of messages to be printed, not the number of messages examined in the log. This is incompatible with -from and -for.

-reverse, -rv
: specifies that the log is to be printed starting with the most recent message selected by other control arguments, and proceed backwards.

-to time, -to number
: specifies the last message to be printed, either by message time or sequence number. If not specified, the default is all the remaining messages in the log. This is incompatible with -for.

CONTROL ARGUMENTS FOR MESSAGE SELECTION

-all_data_classes, -adc
: specifies that all messages, regardless of data class, are to be processed. This control argument cancels the effect of any preceding -match_data_class or exclude_data_class control argument. This is the default.

-all_severities, -asv
: specifies that messages of all severities are printed. This cancels the effect of any previous -severity control arguments. This is the default.

-all_text, -atxt
: specifies that all messages, regardless of text contents, are to be processed. This control argument cancels the effect of any preceding -match or -exclude control argument. This is the default.

-exclude_data_class, CLASS, -exdc CLASS
: specifies that no messages with the specified data class are to be processed.

-exclude STR-1 ... STR-n, -ex STR-1 ... STR-n
: specifies that no message whose text contains one of the specified strings (STR-1 to STR-n) is processed. A string is interpreted either as a text string, which must be an exact substring of the message text, or, if surrounded by slashes, as a regular expression to match against the message text. See the "Notes on String Matching" section below for details.

-match_data_class CLASS, -mdc CLASS
: specifies that only messages with the specified data class are processed. See "Notes on Data Classes," below.

-match STR-1 ... STR-n
: specifies that only messages whose text contains one of the specified strings (STR-1 to STR-n) are processed. The strings are processed as for -exclude.

-no_match_exclude, -nmx
: specifies that all log messages are to be processed, regardless of text contents, cancelling the effect of any preceding -match or -exclude. This is the default.

-severity SEV-1 ... SEV-n, -sv SEV-1 ... SEV-n
: specifies that only messages of the specified severity (severities) are processed. The severity values (SEV-1 to SEV-n) may either be decimal integers, or ranges consisting of a pair decimal integers separated by a colon ("20:29"). If multiple severities are specified, or the -severity control argument is specified more than once, all messages with any of those severities are printed. A severity value must be between -100 and 100. See the "Notes on Severity Values" below for details.

CONTROL ARGUMENTS FOR MESSAGE EXPANSION

-all_data, -ad
: specifies that all messages with binary data are to be processed. This control argument cancels the effect of any preceding -match_data or -exclude_data control argument. This is the default.

-exclude_data STR-1 ... STR-n, -exd STR-1 ... STR-n
: specifies that no messages whose interpreted expanded data contains one of the specified strings (STR-1 to STR-n) is processed. The strings are processed as for -exclude. Note: this control argument merely matches against the textual interpretation of the expanded data; if this interpretation is to be displayed as well, the -interpret control argument must also be specified.

-expand {CLASS-1 ... CLASS-n}
: specifies that binary data is to be expanded and displayed along with the message text, for the selected messages. If a data class value (CLASS-1 to CLASS-n) is specified, only binary data of the specified classes will be expanded; otherwise, all selected messages with binary data will be expanded. The type of expansion depends on whether the -octal or -interpret control arguments are also specified. See the "Notes on Data Classes" section below for details. By default, no messages are expanded.

-interpret, -int
: specifies that the binary data in expanded messages is to be displayed as interpreted text, by calling the appropriate expand_XXX_msg_ program for the data class of the message. If the -octal control argument is also specified, the binary data is displayed both in interpreted form and as octal data. This is the default.

-match_data STR-1 ... STR-n, -md STR-1 ... STR-n
: specifies that only messages whose interpreted expanded data contains one of the specified strings (STR-1 to STR-n) are processed. The strings are processed as for -exclude. Note: this control argument merely matches against the textual interpretation of the expanded data; if this interpretation is to be displayed as well, the -interpret control argument must also be specified.

-no_expand, -nex
: specifies that no messages are to be displayed with binary data expanded. This cancels the effect of any previous -expand control arguments. By default, no messages are expanded.

-octal, -oc
: specifies that the binary data in expanded messages is to be displayed in octal, rather than, or in addition to, the interpreted representation. If both octal and interpreted representations are desired, both the -octal and -interpret control arguments must be supplied.

CONTROL ARGUMENTS FOR MESSAGE FORMAT

-continuation_indent N, -ci N
: specifies that all messages are to be formatted for printing with continuation lines prefixed by N spaces, or, if the keyword "standard" or "std" is used in place of a number, with the continuation lines indented sufficiently to line up under the first character of the text of the message. The value of N must be between zero and fifty. By default, continuation lines are indented to the "standard" indentation.

-date_format FORMAT_STRING, -dfmt FORMAT_STRING
: specifies a date/time format string (see time_format.gi.info or the *Multics Programmer's Reference Manual*, Order No. AG91) to be used when formatting the date when successive messages are printed with different dates. The date string is printed on a line entirely by itself, preceded by a blank line. If the date format string is blank, no date separators will be printed; this should be used if a -time_format string is specified that includes the date as well. The default date string is "^9999yc-^my-^dm ^da ^za", which prints as (for example) "1984-10-31 Wed est".

  By specifying null strings for date, time, and number formats, the log can be printed and saved, so that it can be compared to another log script later, without spurious mis-compares because the times and sequence numbers do not match.

-duplicates, -dup
: inhibits the printing of "=" messages for messages whose text is the same as the previous message printed. All messages are printed exactly as they appear in the log.

-indent N, -ind N
: specifies that all messages are to be formatted for printing prefixed with N spaces. The value of N must be between zero and fifty. The indentation is printed before any data associated with the message, including the message prefix. By default, there is no indentation.

-line_length N, -ll N
: specifies the line length used when formatting message text and data for printing. The value (N) must be between 25 and 500. By default, it is the line length associated with the user_output I/O switch, or, if none (as for an absentee), it is 132 (for line printer output).

-no_duplicates, -ndup
: prints "==" for messages whose text is the same as the previous message printed. This is the default.

-number_format IOA_STRING, -nfmt IOA_STRING
: specifies an ioa_ string to be used when printing the sequence number for the message. If the string is null, no sequence number is printed with the message. The default is "^7d". (See the *Multics Subroutines and I/O Modules* manual, Order No. AG93 for a description of ioa_control strings).

-prefix STRING, -pfx STRING
: Specifies that all messages are to be formatted with the specified string as a prefix. This prefix appears after the indentation (if any was specified). The prefix must explicitly include trailing spaces, if any are desired to separate the prefix from the message text. By default, there is no prefix.

-time_format FORMAT_STRING, -tfmt FORMAT_STRING
: specifies a date/time format string (see time_format.gi.info or the *Multics Programmer's Reference Manual*, Order No. AG91) to be used when formatting the message time portion of the message. If the string is null, no time is printed with the messages. The default time format is "iso_time", which prints as (for example) "23:21:59".

MISCELLANEOUS CONTROL ARGUMENTS

-data_class, -dc
: specifies that the data class of each message is to be printed.

-header, -he
: specifies that a header is to be printed giving the times and sequence numbers of the first and last messages processed. (Default)

-no_data_class, -ndc
: specifies that the data class of each message is not to be printed. This is the default.

-no_header, -nhe
: specifies that no header is to be printed.

-no_process_id, -npid
: specifies that the process_id stored with each message is not to be printed. This is the default.

-output_switch NAME, -osw NAME
: specifies that the messages are to be written on the named I/O switch. The default is user_output.

-procedure NAME, -proc NAME
: specifies that entrypoints in the procedure called NAME are to be used instead of entrypoints in log_read_ to read the log. This is used to read logs protected by inner-ring subsystems, where the inner-ring subsystem provides a replacement log-reading procedure. See "Access Required" section, below.

-process_id, -pid
: specifies that the process_id stored with each message is to be printed.

NOTES ON STRING MATCHING

The strings specified by -match and -exclude, or by -match_data and -exclude_data, are processed in sequence. An arbitrary number of strings may follow any of those control arguments, and each string will be treated as if it was preceded by another instance of the control argument, except that any string beginning with a hyphen and not immediately following one of the match/exclude control arguments is treated as a new control argument, and no more strings are picked up until the next match/exclude argument.

A string may be either a text string, in which case it is tested simply to see whether it is a substring in the message, or it may be a regular expression, which is matched against the message. A string will be interpreted as a regular expression if it begins and ends with "/" characters.

Each log message is processed against the set of strings, matching its text (or data) to see if it contains the string. There are two simple cases: only match strings, and only exclude strings. In the case of only match strings, any log message that matches any of the strings will be printed. In the case of only exclude strings, a log message will be printed only if it matches none of the strings.

The more complicated case where match and exclude strings are mixed is handled as follows: test the message against each string in turn. If the message matches, and the string is a "-match" string, the "print-this-message" flag is set on. If the message matches, and the string is a "-exclude" string, the flag is set off. Otherwise, the flag is unaffected. The flag's initial value is on if the first string was a "-exclude" string, and off if the first string was a "-match" string.

NOTES ON DATA CLASSES

A data class is a short string (1 to 16 characters) stored with any message that contains binary data, and is used to identify the expander procedure used to expand the data into its interpreted textual form. The data class is specified when the message is placed into the log.

The data class "syserr" identifies an old-style syserr log message. The "syserr binary" code (see syserr_binary_defs.incl.pl1 for a list) is the first word of the data in the message; the remaining words of data are the real binary syserr data.

ACCESS REQUIRED

For all except inner-ring logs, read permission is required on the log segments themselves, and status permission is required on their containing directories. If an access error is encountered searching for older history logs, the search is stopped at that point, and no further history will be available. For the logs selected by control arguments, the control argument descriptions list the standard history directories for the logs.

For inner-ring logs (the data management system log is the only standard inner-ring log), access to the logs is required, as is access to the gate used by the log-reading procedure (see -procedure).

NOTES ON MESSAGE SELECTION

Messages are selected for printing in a series of steps, each of which filters out certain messages according to the control arguments specified. The set of messages at each step is any that were left after the previous step. If a control argument was not specified, then its corresponding step eliminates no messages. Note that the -expand control arguments do NOT select messages, but only affect how their contents are displayed

EXAMPLES

```
 1) -to             (stop looking after specified message)
 2) -from           (stop looking before specified number)
 3) -for TIME       (stop looking after specified time)
 4) -last TIME      (stop looking before specified time)
 5) -severity
 6) -exclude        (eliminate matching messages)
 7) -match          (eliminate non-matching messages)
 8) -exclude_data   (eliminate matching messages)
 9) -match_data     (eliminate non-matching messages)
10) -for NUMBER      (stop after NUMBER are printed)
11) -last NUMBER     (stop after NUMBER are printed)
```

Severity values in log messages are used to indicate the importance of the message being logged, in a general way. Most logs use increasing severity to indicate increasing importance, but the actual meaning depends on the log. For the Answering Service and Message Coordinator logs, the severities have the following meanings:

```
0 => Message just logged
1 => Message logged and printed on a console
2 => Message logged and printed on a console with bells
3 => Message logged, printed, and the system crashed
```

```
For the syserr log, the severities have different meanings:
```

```
0 => Message logged and printed on syserr console
1 => Message logged, printed, and the system crashed
2 => Message logged, printed, and the process writing the
     message is terminated.
3 => Message logged and printed, and console alarm sounded
4 => Message just logged, or printed if logging mechanism is
     inoperable
5 => Message just logged, or discarded if it can't be logged
```

The severities 20 to 25 are handled just like 0 to 5, but are different to indicate that the originating program was writing an access audit message, rather than just an informative message.

NOTES ON INNER-RING LOGS

Some applications create logs in an inner ring that must be read using a special interface. The only standard log to do this is the Data Management system log, and it is read by specifying the -dm_system control argument which supplies both the pathname and the procedure name (dm_log_read_). Other applications may provide their own special procedures for log reading, in which case both the log pathname and the procedure name must be supplied explicitly via the -pathname and -procedure control arguments. Note that a log read using a reader procedure may enforce additional access requirements as well as requiring access to the log itself. In particular, the user must have access to the reader procedure.

COMPATIBILITY FEATURES

The following control arguments are accepted for compatibility with the old print_syserr_log and print_log commands:

```
-action is equivalent to -severity
-next is equivalent to -for
-debug, -db is equivalent to -duplicates
```

The effect of print_syserr_log's -class argument can be achieved by supplying a range to the -severity argument: "-class 2" is replaced by "-severity 20:29".

Name: print_tuning_parameters, ptp

SYNTAX AS A COMMAND

```
ptp {name1 ... nameN} {-control_args}
```

FUNCTION

prints the current values of various tuning parameters within the system. The values of most of these tuning parameters can be changed by using the change_tuning_parameters command described earlier in this section.

ARGUMENTS

namei
: is the name of a tuning parameter whose value is to be printed. It can be either the long name or the short name of the parameter. If no names are supplied, all tuning parameters that can be changed while the system is running are printed.

CONTROL ARGUMENTS

-all, -a
: if no names are specified, prints all tuning parameters, including those that are "special" and not alterable while the system is running (e.g., max_max_eligible, which can only be changed by means of a bootload).

-long, -lg
: lists the short and long names of the parameter(s),as well as a pointer to the location of the tuning parameters in ring zero.

-short, -sh
: prints only the long name and the value of the parameter(s) (default).

ACCESS REQUIRED

This command requires access to metering_gate_.

NOTES

| See the *Multics System Maintenance Procedures* manual, Order No. AM81, for explanations of the tuning parameters.

EXAMPLES

The following is an example of the information printed when the print_tuning_parameters command is invoked: (See Appendix A for a representation of the contents of the configuration deck used to create the system from which the metering sample was taken.)

```
Current system tuning parameters:

tefirst                     0.5 seconds
telast                      1. seconds
timax                       8. seconds
priority_sched_inc          80. seconds
min_eligible                2.
max_eligible                20.
max_batch_elig              0
working_set_factor          0.5
working_set_addend          0
deadline_mode               off
int_q_enabled               on
post_purge                  off
pre_empt_sample_time        0.04 seconds
gp_at_notify                off
gp_at_ptlnotify             off
process_initial_quantum     2. seconds
quit_priority               0.
notify_timeout_interval     30. seconds
```

```
notify_timeout_severity   3
write_limit               335
gv_integration            4. seconds
realtime_io_priority      on
realtime_io_deadline      0. seconds
realtime_io_quantum       0.005 seconds
dirlock_writebehind       on
```

Name: print_urf

SYNTAX AS A COMMAND

```
print_urf {Person_id}
```

FUNCTION

prints the name, title, address, programmer number, project, and User_id of each entry in the user registration file (URF).

ARGUMENTS

Person_id
is the person whose entry is to be printed. If omitted, all entries will be printed.

Name: priv_move_quota

SYNTAX AS A COMMAND

```
priv_move_quota path1 quota_change1 {...  pathN quota_changeN}
```

FUNCTION

moves records of quota between two directories (one immediately inferior to the other) regardless of the authorization of the process.

This command is identical to the standard move_quota command except that directory system privileges are turned on while quota is being moved. The priv_move_quota command is needed only if the site is using the AIM access controls.

ARGUMENTS

pathi
: is the pathname of a directory branch. The quota change takes place between this branch and its containing directory. The working directory may be specified by -wd. The star convention may not be used.

quota_changei
: is the number of storage records to be subtracted from the containing directory quota and added to the quota on pathi. If this number is negative, the number of records is added to the containing directory quota and subtracted from the quota on pathi.

ACCESS REQUIRED

The user must have "re" access to system_privilege_ gate and must have modify permission specified on the ACL of both pathi and its containing directory.

Name: process_id

SYNTAX AS A COMMAND

```
process_id {identifiers} {-control_args}
```

SYNTAX AS AN ACTIVE FUNCTION

```
[process_id {identifiers} {-control_args}]
```

FUNCTION

prints or returns a process id of a specified process, in the form of a 12-digit octal number.

ARGUMENTS

identifiers
: can be User_ids, channel names, or APTE offsets. The three types of identifier are distinguished from one another by their format (see "Notes" below). Two of the types can be preceded by a control argument to eliminate any ambiguity (see "Control Arguments"). If no identifier is given, the process ID of the current process is used.

CONTROL ARGUMENTS

-absentee, -as
: selects absentee users.

-all, -a
    selects all three process types. This is the default.

-channel CHN, -chn CHN
    selects the user logged in over channel CHN.

-daemon, -dmn
    selects daemon users.

-interactive, -ia
    selects interactive users.

-multiple
    allows more than one process to be selected. Their process ids are returned, separated by spaces. This is the default if more than one identifier is given.

-single
    requires that the arguments select exactly one process. This is the default, unless more than one identifier is given.

-user User_id
    selects this user.

NOTES

Unless the -multiple control argument is given, or more than one identifier is given, it is an error if the arguments do not select exactly one process.

The type of an identifier not preceded by a control argument is determined as follows: if it contains only octal digits, it is an APTE offset; if it contains any upper case letters, it is a User_id; otherwise, it is a channel name. Channel names and User_ids can be starnames. User_ids are of the form Person.Project.tag. Any of the three components can be omitted, along with any trailing periods. Omitted components are treated as if they had been "*". The presence of a tag component restricts the search to the corresponding user table for that user only.

A channel is a communications channel for an interactive process (e.g., a.h017), an absentee slot number for an absentee process (e.g., abs3), or a message coordinator source name for a daemon process (e.g., bk, prta).

The APTE offset is given as a four- to six-digit octal number (see the print_apt_entry command).

The -absentee, -interactive, and -daemon control arguments can be given in any combination. The default, when User_id with a tag and none of these arguments is given is to search all three user tables.

ACCESS REQUIRED

Read access to the three user tables (absentee_user_table, answer_table, and daemon_user_table) in >sc1 is required, as well as access to the gate metering_ring_zero_peek_ (the latter only if an APTE offset is given as an identifier). None of the above access is required when no identifier is given and the ID of the current process is returned.

EXAMPLES

```
! [process_id *.SysAdmin -as]
```

prints the process_id of the single absentee process from the SysAdmin project. The example is in error if there is more than one absentee process from that project.

Name: proj_usage_report, pur

SYNTAX AS A COMMAND

```
pur {Project_id} {-control_args}
```

FUNCTION

prints a project usage report for the current billing period.

ARGUMENTS

Project_id
: is the Project_id of the project. If this argument is not given, the project under which the project administrator is currently logged in is assumed.

CONTROL ARGUMENTS

-brief, -bf
: prints reports in one short line per user.

-long, -lg
: prints detailed information about per shift, per absentee, per device, and I/O daemon queue usage.

-no_header, -nhe
: suppresses printing of the header.

-pathname path, -pn path
: is the pathname of a PDT. The pdt suffix must be given. This control argument is used to print a PDT not currently being used by the answering service. If this control argument is specified, the Project_id argument may not be given.

-reverse, -rev
reverses the order of the sort.

-sort XX
sorts output according to XX, where XX can be the string:

```
name
usage
rem
limit
fraction_used
```

to specify users' names, usage, remainder, limits, or entries in order of ratio between usage and limit. Only one string may be specified. The default prints the PDT as is.

-total, -tt
does not print a line for each user; rather prints a totals line (plus any other lines specified by other arguments).

-user Person_id
prints information on only the user specified by Person_id.

ACCESS REQUIRED

The user must have read access on the PDT; usually only project administrators have such access.

NOTES

If neither the -brief nor -long control argument is given, the report printed contains one detail summary line for each user.

See also the print_pdt command to get more detailed information about each user and the display_account_status command to obtain a summary of the charges accrued to the project.

Name: read_early_dump_tape, redt

SYNTAX AS A COMMAND

```
redt reel_num -control_args
```

FUNCTION

reads the contents of a tape produced by the early dump facility of BCE and produces a standard format dump in a specified directory.

ARGUMENTS

reel_num
: is the reel number of the early dump tape. This argument may by placed anywhere on the command line.

CONTROL ARGUMENTS

-dump N
: generates a dump with a dump number of N. This control argument is required.

-dump_dir directory
: places the dump into the specified directory. The default is to place the dump into >dumps.

-density N, -den N
: sets the tape density to N. Unless site modified, early dump tapes are written at 1600, which is the default.

-ring, -rg
: mounts the tape with a write ring.

Name: reclassify_dir

SYNTAX AS A COMMAND

```
reclassify_dir path {access_class}
```

FUNCTION

changes the AIM access class of a directory and all immediately inferior segments. The access class of inferior directories and system segments (in ring 1) is not changed.

The reclassify_dir command is needed only if the site is using the AIM access controls.

ARGUMENTS

path
: is the pathname of the directory to be reclassified.

access_class
: is the access class to be assigned to the directory and its segments. Use the print_auth_names command for a list of valid access class values. See the *Multics Programmer's Reference Manual*, Order No. AG91, for a detailed descussion of AIM access controls.

ACCESS REQUIRED

The user must have access to the system_privilege_gate to use this command. Additionally, the user must have modify permission specified on the acl of "path" and its containing directory. However, the reclassification is performed without checking the validation level of the process. Therefore, reclassification of all immediately inferior segments to "path" may be performed from the user ring regardless of the ring brackets of the segments.

NOTES

If the access_class argument is omitted, the current access class of path is assumed, and only the immediately inferior segments are reclassified.

If the new access_class is not greater than or equal to the access class of the containing directory of path, the reclassification is refused. If the new access_class would make the access class of a branch in path inconsistent, the branch is set security-out-of-service. If the new access_class would make a directory in path upgraded, but with 0 quota, this directory is set security-out-of-service. If the new access_class would make path upgraded, but with no terminal quota, the command fails, indicating that terminal quota is required for an upgraded directory.

This command corrects some of the possible inconsistencies that could cause the directory to be marked security-out-of-service by the salvager.

Name: reclassify_seg

SYNTAX AS A COMMAND

```
reclassify_seg path
```

FUNCTION

sets the access class of a segment equal to the access class of its containing directory.

The reclassify_seg command is needed only if the site is using the AIM access controls.

ARGUMENTS

path
: is the pathname of the segment to be reclassified.

ACCESS REQUIRED

The user must have access to the system_privilege_gate to use this command. Additionally, the user must have modify permission specified on the ACL of the containing directory. However, the reclassification is performed without checking the validation level of the process. Therefore, reclassification of ring 1 segments may be performed from the user ring.

NOTES

This command corrects one of the possible inconsistencies that could cause a directory to be marked security-out-of-service by the salvager. However, the directory is not placed back into service by this command. (See the reset_soos command for that function.)

When the specified pathname is a ring 1 system segment (i.e., a segment whose access class is greater than that of its containing directory), this command makes it a ring 1 normal segment (i.e., a segment whose access class is equal to that of its containing directory).

Name: reclassify_sys_seg

SYNTAX AS A COMMAND

```
reclassify_sys_seg path {access_class}
```

FUNCTION

changes the access class of a ring 1 system segment or converts a ring 1 normal segment to/from a system segment. A ring 1 system segment is a segment whose access class is greater than that of its containing directory, e.g., message segments such as those used for daemon queues and user mailboxes; a ring 1 normal segment is a segment whose access class is equal to that of its containing directory.

The reclassify_sys_seg command is needed only if the site is using the AIM access controls.

ARGUMENTS

path
    is the pathname of the segment to be reclassified.

access_class
    is the access class (sensitivity level and category) to be assigned to the segment. Use the print_auth_names command for a list of valid access class values. See the *Multics Programmer's Reference Manual*, Order No. AG91, for a detailed discussion of the AIM mechanism.

ACCESS REQUIRED

The user must have access to the system_privilege_gate to use the command. Additionally, the user must have modify permission specified on the acl of the containing directory. However, the reclassification is performed without checking the validation level of the process. Therefore, reclassification of ring 1 segments may be performed from the user ring.

NOTES

If the segment is to become a normal segment, the access_class argument must be omitted. A specified access_class argument must be greater than the access_class of the containing directory.

Name: reconfigure

SYNTAX AS A COMMAND

```
reconfigure operation type name {-control_args}
```

FUNCTION

adds or deletes selected reconfigurable entities to or from the current configuration.

ARGUMENTS

operation
    is one of the functions listed below under "List of Operations."

type
    is one of the reconfigurable entities listed below under "List of Reconfigurable Entities."

name
    is the name of the item being reconfigured. Examples of names are given under "List of Reconfigurable Entities."

CONTROL ARGUMENTS

-add_all_attachments
used only after an add operation. This control argument causes all reconfigurable entities which are newly accessible to be added.

-brief, -bf
does not print out a list of every item which is added or removed. This is the default.

-delete_all_attachments
used only after a delete operation. This control argument causes all reconfigurable entities which will become inaccessible to be deleted.

-force, -fc
causes an FNP to be deleted from the current configuration even if there are users connected to the FNP. The FNP is deleted and the users are disconnected. (If the -force control argument is not specified when trying to delete an FNP with users connected, an error message is produced and the FNP is not deleted.)

-long, -lg
prints out a list of every item which is added or removed.

LIST OF OPERATIONS

add
the specified item is to be added to the current configuration. It will thus become available for use.

delete, dl
the specified item is to be deleted from the current configuration. It will thus become unavailable for use.

LIST OF RECONFIGURABLE ENTITIES

channel, chan, chnl
a logical channel. A channel's name is the name by which IOI knows it (e.g., a9, b23).

cpu
a Central Processing Unit. A processor's name is its tag as it appears on a cpu card in the config deck (e.g., a, b). Only processors that are defined in the configuration deck can be added. You must set all of the switches correctly and initialize the processor before you issue this command.

device, dv, prph
a peripheral device. A device's name is the name by which RCP knows it (e.g., tapa_03, dskb_13, fnpc, opca).

iom
an Input/Output Multiplexer. An IOM's name is its tag as it appears on an iom card in the config deck (e.g., a, b).

link_adapter, la
a link adapter or physical channel. This is a shorthand way of specifying a collection of logical channels. A link adaptor's name is the name of its lowest numbered channel (e.g., b28).

mpc
a Microporgrammed Peripheral Controller. This is a shorthand way of specifying a collection of link adapters (and thus, a collection of logical channels). An MPC's name is its name as it appears on an mpc card in the config deck (e.g., mspa, mtpb).

page
a page of memory. A page's name is its number. Pages are numbered starting at 0. Numbers may be given in any form acceptable to the cv_integer_string_ function. A range of pages may be specified by an expression of the form <low>:<high>. Pages to be added must reside within system controllers already in the Multics configuration. Pages to be removed must reside within a single system controller.

mem, scu
a System Control Unit. An SCU's name is its tag as it appears on a mem card in the config deck (e.g., a, b). Only system controllers that are defined in the configuration deck can be added. You must set all of the switches on the system controller correctly before you issue this command.

ACCESS REQUIRED

Use of the reconfigure command requires re access to the hphcs_gate.

NOTES

When you add an SCU, all of its pages are also added. However, when you delete all of an SCU's pages, the SCU itself is not deleted.

For more details on the dynamic reconfiguration, see the *Multics System Maintenance Procedures* manual, Order No. AM81, and the *Operator's Guide to Multics*, Order No. GB61.

EXAMPLES

The examples below assume the following config deck fragment:

```
   iom b 1 iom off
   mpc mspd 607. b 28. 4
   prph dskd b 28. 4 0 16. 501. 16    <dskd_17 - dskd_32 are 501s>
   mem b 2048. on                     <contains pages 0 - 2047>
   mem a 1024. on                     <contains pages 2048 - 3071>
   cpu c 5 off dps8 70. 32.
```

```
!  reconfigure add cpu c
```

adds CPU c to the configuration.

```
!  reconfigure dl mem a -delete_all_attachments
```

deletes SCU a and pages 2048 through 3071 from the configuration.

```
!  reconfigure add iom b -add_all_attachments
```

adds IOM b to the configuration. MPC mspd is also added, and if any of dskd_17 through dskd_32 were previously deleted, they are added as well.

Name: reconstruct_registry

SYNTAX AS A COMMAND

```
reconstruct_registry registry_names {-control_args}
```

FUNCTION

recovers a current copy of RCP resource management registries after a catastrophic system failure causing the loss of one or more registries. It assumes that the registry to be reconstructed is a consistent earlier copy of the registry desired, and that the RCP resource management journal contains a record of all operations performed on the registry since the time represented by the earlier copy.

ARGUMENTS

registry_names
    are the entrynames of the registries to be reconstructed. You can use the star convention. If you give no .rcpr suffix, it is assumed.

CONTROL ARGUMENTS

-pathname path (-pn path)
to specify the directory in which the registries reside. If this control argument is not specified, the registries are sought in >sc1>rcp.

NOTES

An explanation of the creation and maintenance of checkpointed registry copies can be found in the documentation of the copy_registry command.

The prescribed sequence of operations is to delete the damaged registries; copy the desired checkpointed registries into place; and invoke the reconstruct_registry command to update the registries. The command locates the RCP Resource Management journal relative to the directory in which the registries to be updated reside.

If an online checkpoint copy of a system registry is not available, a copy of the registry may be retrieved from a system backup tape. In this case, the file retrieved must be from a time that is more recent than the last time the RCP Resource Management journal was reset (see the documentation of the copy_registry command).

The reconstruction of system registries must only be performed from the Initializer, in the "standard" environment, before the answering service is activated.

ACCESS REQUIRED

This command requires access to the rcp_sys_ gate.

Name: record_to_sector

SYNTAX AS A COMMAND

```
record_to_sector record_no {device_name}
```

SYNTAX AS AN ACTIVE FUNCTION

```
[record_to_sector record_no {device_name}]
```

FUNCTION

converts an octal sector number to a disk sector address.

ARGUMENTS

record_no
    is the octal Multics record number.

device_name
    is a valid device name (e.g., "m400", "m451").

Name: record_to_vtocx

SYNTAX AS A COMMAND

```
record_to_vtocx pv_name arg1 ...  argN
   record_to_vtocx pv_name -sector sector_arg1 ...  -sector sector_argN
```

FUNCTION

finds any VTOC entries corresponding to a specified record number on a storage system volume.

ARGUMENTS

pv_name
    is the name of the physical device.

argi
    is the octal record number.

sector_argi
    is the octal sector number.

ACCESS REQUIRED

You need access to the phcs_ gate.

NOTES

Looking for the correct match, this command scans the VTOCEs in ascending order for each argument; therefore it uses great amounts of CPU time and requires considerable I/O.

Name: register_mdir

SYNTAX AS A COMMAND

```
register_mdir {path} {-control_args}
```

FUNCTION

is used to register an existing master directory in the ring 1 master directory control segment (MDCS) for its volume. This may be necessary if the MDCS has become damaged or lost.

ARGUMENTS

path
: is the name of the master directory to register, or if -all is used, the starting node of the hierarchy tree to be scanned for master directories. This argument can only be omitted if the -all control argument is specified, in which case it defaults to the root.

CONTROL ARGUMENTS

-all
: specified that the path given is the starting node to scan for master directories to register. All directories including the starting node are checked and registered if they are master directories.

-brief, -bf
: specifies that the names of master directories registered by this command are not to be printed. (Applies only when -all is used.)

ACCESS REQUIRED

Access to the gate mdc_priv_ is required to use this command.

NOTES

This command is only required as part of an error recovery procedure in repairing a master directory control segment. Normally, all master directories are registered in the MDCS when they are created.

All master directories registered by this command are charged to the quota account Initializer.SysDaemon, which is automatically created if it does not exist.

Name: register_mdir

SYNTAX AS A COMMAND

```
register_mdir {path} {-control_args}
```

FUNCTION

is used to register an existing master directory in the ring 1 master directory control segment (MDCS) for its volume. This may be necessary if the MDCS has become damaged or lost.

ARGUMENTS

path
: is the name of the master directory to register, or if -all is used, the starting node of the hierarchy tree to be scanned for master directories. This argument can only be omitted if the -all control argument is specified, in which case it defaults to the root.

CONTROL ARGUMENTS

-all
: specified that the path given is the starting node to scan for master directories to register. All directories including the starting node are checked and registered if they are master directories.

-brief, -bf
: specifies that the names of master directories registered by this command are not to be printed. (Applies only when -all is used.)

ACCESS REQUIRED

Access to the gate mdc_priv_ is required to use this command.

NOTES

This command is only required as part of an error recovery procedure in repairing a master directory control segment. Normally, all master directories are registered in the MDCS when they are created.

All master directories registered by this command are charged to the quota account Initializer.SysDaemon, which is automatically created if it does not exist.

Name: register_resource, rgr

SYNTAX AS A COMMAND

```
rgr type STR1 ... STRN {-control_args}
```

FUNCTION

makes a particular resource known to the system. The registration process informs the system that the resource is available for users who are authorized to access it.

ARGUMENTS

type
: is a resource type defined in the RTDT. See "Reserved Names" below for additional information.

STRi
: is the unique identifying name of the particular resource being registered. If STR is specified in control argument format (i.e., if it is preceded by a hyphen), then it must be preceded by -name or -nm. (The string "scratch" is not permitted.)

CONTROL ARGUMENTS

-access_class accr, -acc accr
: sets the initial AIM access class parameters, where accr is an access class range. Users at any authorization within the access class range inclusive are allowed to read and write to the resource (provided they also meet other access requirements).

-acs_path path
: specifies the pathname of the access control segment (ACS) for this resource. The ACS is not created by this command, but must be created by the administrator, and the desired access control list set (see "Notes" below). If this control argument is not given, the accounting owner of the resource is given rew access by default. If path is a null string, the existing ACS, if any, is disassociated from the resource.

-alloc STR
: sets the allocation state of the resource to free or allocated, where STR must be either the string on or the string off. If this control argument is not given, the allocation state is free. (The allocation state flag is a convenience to the user and is largely ignored by resource management.) on sets the allocation state to allocated off sets the allocation state to free

-attributes STR, -attr STR
: specifies the initial values for the attributes of this resource. If this control argument is not given, the default attributes defined in the RTDT for this resource type are used (see "Naming Rules for Attributes" below).

-comment STR, -com STR
: specifies the initial value of the comment string for this resource.

-location STR, -loc STR
: specifies a descriptive location for the resource, to aid the operator in locating it when it is stored in a special place (e.g., a vault, a different room, etc.).

-lock STR
: locks or unlocks the resource, preventing or allowing use of that resource, where STR must be either the string on or the string off. If this control argument is not specified the lock is off.

```
on   prevents any use of the resource

off  allows use of the resource
```

-owner STR, -ow STR
: specifies that this resource, as part of the registration process, is to be acquired on behalf of the user specified by STR. If STR is the string "system", then the resource is acquired to the system pool. If STR is of the form Person_id.Project_id (where neither Person_id nor Project_id may be a star), then the user specified has all the rights of ownership to the resource as if he had acquired it personally, except that if -release_lock on is specified, the owner may not release (give up ownership of) the resource voluntarily. If this control argument is not given, the resource is entered by default into the free pool.

-potential_attributes STR, -pattr STR
: specifies the potential attributes to be assigned to this resource. If this control argument is not given, the default potential attributes defined in the RTDT for this resource type are used (see "Naming Rules for Attributes" below).

-potential_access_class accr, -pacc accr
: sets the potential AIM access class parameters, where accr is the access class range. Users at any authorization within the access class range inclusive are allowed to acquire the resource. If the control argument is not given, the default potential access class defined in the RTDT for this resources type is used. See "Access Class Ranges" below, for additional information.

-release_lock STR, -rll STR
specifies whether this resource may be released by the owner, or may only be released by a privileged process. The STR argument must be either the string on or the string off. It is primarily useful to implement special arrangements between a site and a user whereby the user agrees to pay a fixed amount for the privilege of administrative power over a resource for an agreed-upon length of time. If this control argument is not specified, the resource may be released by the owner (does not require special privilege).

on	resource may only be released by privileged processor
off	resource may be released by owner

-type subtype_name, -tp subtype_name
specifies that defaults for this resource are to be taken from the description of the resource subtype as defined in the RTDT (see "Application of Defaults" below for additional information).

NOTES

If multiple resources are specified to the register_resource command and an error occurs in the registration of any of these resources, none of the resources specified is registered.

If no -owner is specified, the resource is placed in the free pool.

The use of the -access_class, -acs_path, -attributes, or -comment control argument requires that the -owner control argument be specified.

ACCESS REQUIRED

The use of this command requires execute access to the rcp_admin_ gate.

Certain specifications of AIM access class parameters (e.g., an access class lower than the user's current authorization) are rejected unless the user has the AIM rcp privilege.

ACCESS CONTROL

There are three types of access control on Multics: discretionary access control, which is regulated by access control lists (ACL); nondiscretionary access control, which is regulated by the access isolation mechanism (AIM); and intraprocess access control, which is regulated by the ring structure. (For detailed information on types of access, see the *Multics Programmers' Reference Manual*, Order No. AG91.)

ACCESS CONTROL SEGMENTS

An important feature of RCP is its ability to control access to the various resources that it manages. It does this through the use of access control segments (ACSs). An ACS is a zero length segment whose ACL and ring brackets are used to define the discretionary access to a resource. RCP uses an ACS for each resource that it controls; however, an ACS can be shared by more than one resource. The name of an ACS consists of a name plus the suffix, acs (e.g., tape_01.acs). There are no restrictions on ACS names other than the required suffix. The user creates an ACS and generates/manipulates its ACL with the create, set_acl, and delete_acl commands and ring brackets with the set_ring_brackets command.

The pathname of the ACS for a resource is usually specified when it is acquired. The specified ACS can later be changed via the set_resource command (see the *Multics Commands and Active Functions* manual, Order No. AG92). If the ACS has not been specified or does not exist, access is by default rew for the owner of the resource and null for all other users.

RCP uses the ACS along with other nondiscretionary controls (AIM) to determine the RCP effective access to a resource.

ACCESS CLASS RANGES

Access class ranges are used by RCP to specify that a process within a range of authorizations can use a particular resource.

An access class range is simply a pair of AIM access classes separated by a colon. The first value of the pair is the minimum access class and the second is the maximum access class. If only a single access class is specified when an access class range is expected, the minimum and maximum access class values are both the same (i.e., a range of one value). The second access class of the pair (the maximum) must be greater than or equal to the first (the minimum).

The user should be aware of results which occur when categories are used in an access class range. For example, a process with authorization of:

```
level2,category1
```

would not be able to use a resource whose access class range was:

```
level1,category1,category2:level3,category1,category2,category3
```

where level3 is greater than level2, which is greater than level1. This is due to the fact that the authorization of the process is isolated from the minimum of the access class range. In order to allow this process access to the resource in question, the range would have to exclude category2 or the user would have to have category2 authorization. In general, to include categories within an access class range, both the minimum and maximum must include the categories desired. If combinations of categories are desired, the minimum should list only required categories and the maximum should include all categories allowed. For example, the access class range:

```
level1,category1:level3,category1,category2,category3
```

allows read and write access to any level1, level2, or level3 process with category1 and any combination of category2 and category3.

RCP EFFECTIVE ACCESS

Viewed separately, each type of access control answers the same question, "What access does a particular process have for a particular item?" The access mode granted a process to a resource by discretionary access control (the ACL) is known as the raw access mode.

The way RCP determines effective access to a resource for a process differs from the regular Multics method of determining effective access as follows. First, the effective access to the ACS for the resource is determined as for any segment. If the ACS does not exist, the user appears to have read, execute, and write access if he is the owner of the resource, or null access if he is not the owner. Then, two further checks are made. First, the current authorization of the process is compared to the maximum access class of the resource. If write access is not allowed (as defined by the write_allowed_ subroutine) then write and execute access are denied and only read is allowed. Next, the current authorization of the process is compared to the minimum access class of the resource. If read access is not allowed (as defined by the read_allowed_ subroutine) then all access is denied. The resulting access is termed the RCP effective access to the resource. One final restriction enforced by RCP is that, in order to use a device, the RCP effective access must include both read and write to that device (a restriction not imposed on volumes).

For example, the following table illustrates some examples of RCP effective access. In the examples below, L1, L2, L3 and L4 represent sensitivity levels and c1, c2, c3, and c4 represent categories. (This discussion mostly concerns devices--volumes should never be given a multiclassed access class range.)

Table 2-1. RCP Effective Access

Effective Access to ACS	Current Process Authorization	Resource Access Class Range	RCP Effective Access
rew	L1	L1:L3	rew
re	L1	L1:L3	re
rew	L1	L2:L3	null
rew	L3	L2:L3	rew
rw	L4	L2:L3	r
re	L4	L2:L3	r
rw	L2,c1	L1:L4	r
rw	L2,c2	L1,c1:L4,c1,c2	null
rw	L2,c1,c3	L1,c1:L4,c1,c2	r
rw	L2,c1	L1,c1:L4,c1,c2	rw

A user must have write RCP effective access to the resource named to perform any modification on the status of the resource. In addition, the user must have execute effective access to the resource named to modify protected attributes. Only the accounting owner may modify the ACS path.

For more information on AIM, access classes, authorizations, and comparisons involving access classes and authorizations, see the *Multics Programmer's Reference Manual*, Order No. AG91.

MANIPULATING RCP EFFECTIVE ACCESS

Since the access control mechanisms described above operate together to determine the RCP effective access of a process, there are actions that the user, as well as an administrator, can perform to control this effective access.

First, the user creates an ACS via the create command. Then, the desired ACL for that segment is established using the set_acl command to add desired ACL entries, and the delete_acl command to delete entries. (The above three commands are described in the *Multics Commands and Active Functions* manual, Order No. AG92.) To further affect the ACS, the user may modify its ring brackets by using the set_ring_brackets command (described in the *Multics Commands and Active Functions* manual, Order No. AG92). The system security administrator sets the AIM access class range of the resource itself at the time it is registered using the register_resource command, and can change it by using the set_resource command.

RESERVED NAMES

RCP uses the information in the RTDT to decide what classes of resources are known to the system, how they are to be handled, and what important attributes they possess. In the initial implementation, sites may use this flexibility to augment the standard complement of attributes for certain resources. For example, a site with tape drives in more than one location may register these drives with an additional simple attribute, thereby allowing users to request assignment of a tape drive in the remote location. Additionally, the tape reels in the remote location may be tagged with a matching attribute, marked in the RTDT as requiring that attribute of its tape drive.

Although this mechanism is very flexible, the necessity of having certain standard and reserved resource type names and attribute names cannot be avoided. Standard software (e.g., tape and disk I/O modules) needs to refer to a domain of resources by standard names, as well as certain attributes of the resources. Since these strings must be the same at all sites, certain resource types and certain resource attributes must be contained in all RTMFs. The cv_rtmf command checks for their existence and refuses to process an RTMF that lacks them. This list of required resource type names and attributes is also found in the include file, rcp_mandatories.incl.pl1.

RCP does not allow the name "scratch" to be used in registering a resource. A scratch tape is one of the unmarked tapes in an unreserved pool that is used for "scratch"--that is, no information is saved on it from session to session. After every use, it is demounted and returned to the system pool.

RESERVED RESOURCE NAMES

The following resources are mandatory and must appear in all RTMFs:

```
Device:   disk_drive
Device:   tape_drive
Volume:   tape_vol
Volume:   disk_vol
```

RESERVED ATTRIBUTE NAMES

The following attributes are mandatory for the devices named, and must appear in all RTMFs:

For the disk_drive device:

```
model=400          model=451          model=181
model=191          model=500          model=402
```

For the tape_drive device:

```
track=7          track=9
den=200          den=556
den=800          den=1600
model=400        model=500
model=600        model=610
```

NAMING RULES FOR ATTRIBUTES

Attributes provide a description of a volume or device that assists the resource management facility in the proper matching of volumes with compatible devices. To produce correct combinations, attribute names must comply with the set of rules described below.

Attributes may be grouped or ungrouped. Grouped attributes specify a set of properties applicable to a device or volume such that only one attribute of that set can be currently active at any given time. For example, a reel of tape may have potential attributes that allow it to be recorded at densities of 556, 800, or 1600; however, at any given time, the data on it is in only one of those densities. Grouped attributes have names of the form:

```
<identifier>=<value>
```

For example, the attributes mentioned above are named "den=556", "den=800", and "den=1600". This notation allows RCP to recognize that any request to make one of these attributes the current attribute of a device or volume also implies that all other attributes in that grouping must be made inactive.

When adding or changing an attribute in a string of attributes, all attributes in the string must be respecified or else existing attributes are nullified by the change. Also, any attribute string must contain a value for each grouped attribute. For example, if the attribute domain includes "track=..., model=..., and den=...," the device you are setting the attributes for (or registering) must contain values for each grouped attribute.

Ungrouped attributes have simple names, such as "trainok" (to specify that this device accepts a removable print train) or "building_12" (to specify that this device or volume is located in building 12).

APPLICATION OF DEFAULTS

When the system administrator registers a resource, that resource may be registered using the defaults for the registration parameters that are specified in the RTDT. Alternately, he may explicitly specify parameters for which defaults may also be specified in the table, such as attributes and AIM classes. If any such parameter is explicitly specified, the corresponding default for that parameter is overridden.

When the resource is registered, any default parameters defined for that resource type are applied in the absence of a corresponding explicitly specified parameter.

If the resource is registered with the "-type <subtype_name>" control argument, any default parameter defined for the special class named <subtype_name> is applied in the absence of a corresponding explicitly specified parameter. In the case of duplicate resource type and special class parameters, the special class default parameters override the general resource type parameters. In addition, any default parameters specified for that resource other than those defaults in the special class are applied.

If no special classes of a resource are defined, and the defaults for the resource are not all present, it is always necessary for the missing parameters to be explicitly specified for every registration request for a resource of this type. If special classes of a resource are defined, then defaults within the definition of special classes can be used either to replace corresponding defaults specified for the resource in general, or to supplement for missing defaults that are not specified for the resource in general. In the latter case, the system administrator cannot perform a simple default registration of the resource, but must either specify the missing items explicitly in the command line, or use the "-type <subtype_name>" control argument to take advantage of the additional defaults provided in a special class.

Name: reload

SYNTAX AS A COMMAND

```
reload {-control_args}
```

FUNCTION

reloads the entire storage system from a hierarchy dump tape. It deletes segments and directories from the hierarchy that were not present when the hierarchy dump tape was made, and replaces existing segments and directories with their counterparts from the tape. However, to avoid destroying useful information, it does not delete directories, and it does not replace segments and directories in the hierarchy that were modified after the time at which they were dumped to tape. The reload command calls the backup_load command to do the actual reloading.

The reload command places its map in the directory >reload_dir (to which the process must have sma access), and automatically dprints it. Quota on the reloaded directories is force-set to that specified on the hierarchy dump tape.

The reload command is one of the commands used for hierarchy reloading and retrieving of storage system segments and directories. The other commands are:

```
    backup_load
    reload (initializer command)
    reload_system_release
    retrieve
```

Do not confuse this reload command, which is a Multics command, with the reload initializer command.

You should note that argument processing for all of the hierarchy backup commands is performed by a common argument processing procedure. The values of all arguments are remembered in static storage and remain in effect for the life of the process, unless changed by arguments given in subsequent invocations of backup commands. It should also be noted that the dumping commands and the reloading/retrieving commands are all part of the same hierarchy backup system, and argument values set by the dumping commands remain in effect for the reloading/retrieving commands and vice versa, unless overridden. However, dumping and reloading cannot be done in the same process; use the new_proc command between dumping and reloading. See "Notes on Default Arguments" below.

CONTROL ARGUMENTS

-all
: causes segments to be retrieved from the tape regardless of their date/time dumped. This control argument overrides a previously given DATE argument. This is the default.

-brief_map, -bfmap
: creates a map file that lists the processed entires.

-debug
: disables those hphcs_ calls that set quotas and transparency switches.

-destination STR, -ds STR
: specifies a destination for printing maps and error file. The default is "incremental" for maps and "error file" for error files.

-dprint, -dp
: causes the reload command to dprint maps and error files. This is the default.

-error_of
: writes error messages into a file rather than printing them. The name of the error file is printed when the first error is encountered. This is the default.

-error_on
writes error messages on the user's terminal.

-first
prevents searching a tape for additional copies of a requested segment or subtree after the first copy has been retrieved.

-header STR, -he STR
specifies a heading for printing maps and error files.

-last
indicates that the last copy of a given segment or subtree on a tape or set of tapes is to be retrieved. This is the default. *

-map
writes a list of the segments and directories processed into a file. This is the default.

-nodebug
enables hphcs_ calls to set quotas and the transparency switches. This is the default.

-nodprint, -no_deprint, -ndp
causes the reload command not to dprint maps and error files. The default is -dprint.

-nomap
inhibits listing of the names of processed segments and directories. This control argument and the -noreload control argument are incompatible when used with the reload command; only one or the other can be used.

-noprimary, -npri
uses each pathname as given. The default is -primary.

-noqcheck
causes the hierarchy reload to be done with quota checking suspended. Access to hphcs_ is required. This is the default.

-noquota
inhibits resetting of quotas. See -quota. This is the default.

-noreload
inhibits actual hierarchy reloading of segments into the hierarchy. This control argument can be used with -map to create a table of contents of the tape. The -noreload control argument also causes the names that would have been reloaded to be put into the map. This control argument and the -nomap control argument are incompatible when used with the reload command; only one or the other can be used.

-nosetlvid
: inhibits the setting of the logical volume identifiers for each directory to be reloaded.

-notrim
: inhibits deletion of entries in a directory. Entries can only be added or modified.

-operator STR
: indicates that STR is the user's name or initials (up to 16 characters in length).

-primary, -pri
: replaces all directory names in each pathname with the primary names. This is the default.

-pvname STR
: indicates that segments and directories may only be reloaded onto the physical volume specified by STR.

-qcheck
: causes quota restrictions to be enforced during the reload.

-queue N, -q N
: specifies a queue number for any maps and error files that are dprinted. The default is queue 1.

-quota
: causes the quotas on directories being reloaded to be set to the values they had when the directories were dumped. Access to hphcs_ is required. This is the default.

-reload
: enables actual reloading of segments into the hierarchy. This is the default.

-request_type STR, -rqt STR
: specifies an output request type for printing maps and error files. Available request types can be listed by using the print_request_types command (described in the *Multics Commands and Active Functions* manual, Order No. AG92). The default is "printer".

-setlvid
: enables setting of the logical volume identifier for reloaded entries inferior to each directory reloaded. This is the default.

-trim
: enables deletion of all entries in a directory not found in the copy of that directory being reloaded. This causes entries deleted from an earlier version of the directory to be deleted when a later version is reloaded. This is the default. It has effect only in the case of a directory that is both on the tape and in the hierarchy.

DATE
: an argument beginning with a character other than "-", or ">" is assumed to be a date in a format acceptable to the convert_date_to_binary_ subroutine. If it can be converted successfully, then the hierarchy retriever only retrieves segments and directories dumped at or after the given date/time.

NOTES ON DEFAULT ARGUMENTS

The values of arguments given to any of the hierarchy backup commands are remembered in static storage and remain in effect for the life of the process, unless explicitly changed during the invocation of a subsequent backup command.

The following defaults are in effect for the reloader and retriever before any backup commands are given; they are not, however, reset to these values at the start of each backup command, except as noted below.

```
-all
-error_of
-map
-nodebug
-nohold
-noquota
-primary
-reload
-setlvid
-trim
```

The following defaults are set automatically at the time the respective commands are executed:

```
reload (initializer command), reload (Multics command),
  reload_system_release:
   -quota
   -trim

retrieve:
   -all
   -noquota
   -notrim

All of the above commands:
   -map
```

*

Name: remove_registry

SYNTAX AS A COMMAND

```
remove_registry paths
```

FUNCTION

remove RCP Resource Management registries from service. This command should only be used in exceptional circumstances. (See "Notes" below.)

ARGUMENTS

path
: is the pathname of a registry to be removed from service. The star convention is accepted. If the suffix rcpr is not given, it is assumed.

NOTES

When a registry is removed, its suffix is changed from rcpr to old.

The activity of removing registries is normally reserved to the Initializer process, which will automatically remove a registry when a new RTDT is installed that no longer contains an entry for the resource type associated with that registry. In general, manual removal of registries is only necessary in the process of recovery from a catastrophic system failure and reload, where the existing registries and the existing RTDT may be out of agreement. Manual removal of registries at other times can result in unrecoverable errors by RCP Resource Management.

ACCESS REQUIRED

This command requires access to the rcp_sys_ gate.

Name: remove_user

SYNTAX AS A COMMAND

```
remove_user user_name
```

FUNCTION

is used to delete information about a user from both the PNT and the URF.

ARGUMENTS

user_name
: (in the form Person_id) is the name of the user to be deleted.

Name: reset_cdt_meters

SYNTAX AS A COMMAND

```
reset_cdt_meters {path}
```

FUNCTION

is invoked by biller.ec at the end of a billing period to reset per-system and per-channel usage counters kept in the CDT.

ARGUMENTS

path
: is the pathname of the CDT. If path is not specified, the segment >sc1>cdt is reset.

Name: reset_disk_meters

SYNTAX AS A COMMAND

```
reset_disk_meters {path} {-control_arg}
```

FUNCTION

subtracts the time-page-product (tpp) for each directory in the Multics hierarchy listed in disk_stat from the tpp integrator of the directory itself. This operation is done once each month after users have been billed for disk usage. Privileged entry points in the hphcs_ gate are used to do the work. A comment is printed if the branch does not exist or a negative tpp would result, but the program continues.

ARGUMENTS

path
: is the pathname of the disk_stat segment containing the usage values to be subtracted from the tpp integrators. The default is the segment disk_stat in the working directory.

CONTROL ARGUMENTS

-brief, -bf
: specifies that the user is not to be informed when it is necessary to force access to a directory to reset its tpp.

NOTES

This command forces access to directories, if necessary, and removes the access when finished. It reports when it must do this, unless the -brief control argument is given, and it always reports if unable to force access.

After resetting all of the tpps, it zeros the per-project disk usage figures in the segment projfile, in the working directory. (This segment is an implicit input to the command.)

If the system crashes while this command is running, the command can be restarted from the beginning.

Name: reset_soos

SYNTAX AS A COMMAND

```
reset_soos path
```

FUNCTION

resets the security-out-of-service switch for the specified directory after verifying that it is consistent with respect to AIM access controls.

The reset_soos command is needed only if the site is using the AIM access controls.

ARGUMENTS

path
    is the pathname of the directory that is to be put back into service.

ACCESS REQUIRED

The user must have access to the system_privilege_gate to use this command.

NOTES

This command fails and prints an error message if the directory:

1. is upgraded without terminal quota;
2. contains segments with an access class unequal to the containing directory;
3. contains directories with an access class that is not greater than or equal to the containing directory; or
4. contains inconsistent upgraded directories or ring 1 system segments.

The star convention is not accepted by this command. Only one pathname argument is accepted.

LIST OF COMMANDS

The following commands may be used to correct inconsistencies:

```
    reclassify_seg
    reclassify_dir
    reclassify_sys_seg
    priv_move_quota
```

Name: reset_usage

SYNTAX AS A COMMAND

```
reset_usage sat_path pdt_dir admin_sat admin_pdt
```

FUNCTION

subtracts the resource usage in each PDT located in an administrative directory from the resource usage information in the system copy of each PDT. This procedure modifies only the system copy of the PDTs.

ARGUMENTS

admin_pdt
: is the pathname of the directory in which administrative copies of the PDTs reside.

admin_sat
: is the pathname of an administrative copy of the SAT.

pdt_dir
: is the pathname of the directory containing the system copies of the PDTs.

sat_path
: is the pathname of the system copy of the SAT.

NOTES

This command is used by biller.ec. Projects are billed for the usage recorded in the administrative copies of the PDTs. (These copies are created by the crank, called by master.ec, from the system copies of the PDTs.) Meanwhile, logged in users are accruing additional usage charges, recorded in the system copies of the PDTs. Execution of this command subtracts the charges that have just been billed, leaving in the system PDTs only the most recently accrued charges that have not yet been billed.

If the system crashes while this command is running, the command can be restarted from the beginning.

Name: reset_use_totals

SYNTAX AS A COMMAND

```
reset_use_totals use_totals control
```

FUNCTION

clears all totals in a month-to-date statistics data base at the end of a billing period and reloads the list of reporting categories.

ARGUMENTS

use_totals
    is the pathname of the statistical data base.

control
    is an ASCII file that classifies projects into reporting categories.

NOTES

Each line in the control file is of the form:

```
Project_id, reporting category
```

All projects in the same reporting category are classified together in the data base by the usage_total program. Lines beginning with "*" are ignored.

The last control line in the file must be of the form:

```
other, category title
```

to provide for the classification of all other projects. Up to 3258 individual projects (in addition to the "other" control line) and 300 category titles may be specified.

This command is executed by biller.ec with the arguments:

```
   today.use_totals daily_report.control
```

after a copy of the use_totals segment (containing statistics for the month just ending) has been saved for historical purposes. This command initializes the today.use_totals segment to begin collecting statistics for the next month. This is the only time during the month when the classification of projects into reporting categories may be changed. The pathname of the control segment, describing the classification, must be given, even if the classification is not changing.

If multiple rate structures are defined at the site, it is recommended that a separate category (or group of categories) be used for the projects in each rate structure (i.e., that projects of different rate structures not be placed in the same category). The rate structure of the first project in each category (in the control file) is used for computing usage statistics for all projects in that category.

If the use_totals segment does not exist, it is created. This segment is updated by the system_total and usage_total commands and is input to the system_daily_report and system_monthly_report commands.

Name: response_meters

SYNTAX AS A COMMAND

```
response_meters {-control_args}
```

FUNCTION

displays statistics on interactive response time.

CONTROL ARGUMENTS

-report_reset, -rr
    generates a report and then performs the reset operation.

-reset, -rs
    resets the metering interval for the invoking process so that the interval begins at the last call with -reset specified. If -reset has never been given in the process, it is equivalent to having been specified at system initialization time.

-total, -totals, -tt
    prints summary information for all work classes.

-work_class N, -wc N
    prints information only for work class number N.

ACCESS REQUIRED

This command requires access to phcs_ or metering_gate_.

NOTES

If neither -reset nor -report_reset is specified, response_meters prints a report.

The control arguments -total and -work_class cannot be used together. If either of these is used, the remaining control arguments must be such that response_meters prints a report.

The fundamental concept needed to understand the output produced by the response_meters command is that of a terminal interaction. A terminal interaction is essentially an atomic unit of work by a Multics user at a remote terminal. A terminal interaction is triggered by a set of characters keyed by a user at a terminal, which are passed into the user ring as a unit. The activity between the receipt of the set of characters at the Multics mainframe and the next request by user-ring software for more input is a terminal interaction. For a typical user, a terminal interaction corresponds to a line of input. Some user-ring software, however, receives input as sets of characters rather than as lines (the Emacs editor is an example of such software). For software of this sort, a terminal interaction corresponds to the set of characters handed out by ring 0 TTY software at one time. If, in processing a set of characters keyed by a user at a terminal, the user-ring software blocks itself on anything other than terminal input, that activity is not considered a terminal interaction. For example, any command that results in tape I/O is not considered a terminal interaction.

The output of response_meters is summarized by work class. Any work class for which no interactions were measured is not displayed. The following are brief descriptions of the variables printed out by response_meters for each work class and for the system:

WC
: is the work class number. Work class "All" represents statistics for the entire system.

# Thinks
: is the number of times a process in the work class blocked itself awaiting terminal input. The term "Think" is used because the delay represented here corresponds to what is commonly called "Think Time" in models of interactive computer systems. This term is derived from the assumption that during this time the interactive user is cogitating about the response just received and is formulating an appropriate next input.

Avg Thinks
: is the average time in seconds for all blocks awaiting terminal input. This is the average external delay between interactions to the same process in the work class. This external delay includes communications processing time (in the FNP), communications line delay, and user idle (or "Think") time.

# Queues
: is the number of times a process in the work class was queued for eligibility following receipt of terminal input. It may be less than the number of interactions because of type-ahead. That is, by keying input sufficiently rapidly, a user may cause several interactions to be processed during the same eligibility period. It may be larger than the number of Thinks because of perturbations associated with process initiation.

Avg Queues
: is the average eligibility queue time (in seconds) following the receipt of terminal input. This is the average clock time between the receipt of the terminal input at the Multics mainframe and the awarding of eligibility to the target process by traffic control. The average queue time corresponds to what is called "response time" in the output of traffic_control_meters.

Load Control Group
: is the set of Load Control Groups corresponding to the work class. Read access to the Answer Table and to the Master Group Table (MGT) is required for this information to be displayed. If the requesting process does not have this access, then this field is left blank.

Response times by VCPU Range
: Up to four detail lines and one summary line are printed, with each line corresponding to a set of interactions. Each interaction measured is categorized by the amount of virtual CPU time consumed by the interaction. Based on the virtual CPU time for the interaction, the interaction is recorded against one of four "buckets." Each of the detail lines represents all terminal interactions for processes in the work class that fell into the corresponding "bucket." A bucket is simply a range of virtual times. Intuitively, the first bucket represents trivial interactions, the last bucket represents exceptionally long interactions, and the remaining two buckets lie between these extremes. If there were no interactions recorded against a bucket, the detail line corresponding to that bucket is not printed. The summary line contains information on all terminal interactions for processes in the work class.

VCPU Range
  From
  To
: is the range of virtual CPU times for interactions presented on that line, in seconds. "From" is the beginning of the range, and "To" is the end of the range. The summary line is identified by "-----" in both the "From" and the "To" fields.

# Int
: is the number of interactions in the bucket for the line. That is, this is the number of interactions whose virtual CPU time fell between the values of "From" and "To."

Avg VCPU
: is the average virtual CPU time per interaction for the bucket.

Avg RT
: is the average response time per interaction for the bucket. It is the average clock time required to process an interaction, including any eligibility queue time.

Resp Fact
: is the response factor. This is defined as the ratio of the average response time per interaction to the average virtual CPU time per interaction. Intuitively, this ratio represents an "extension factor" for virtual CPU time. That is, it is a factor expressing the average amount of clock time required to process an interaction with a given virtual CPU time. This number can be used as a "quick" indicator of work class or system performance.

In addition to the per-work class data described above, the following summary information is displayed:

calls to meter_response_time
: is the number of subroutine calls to the ring 0 routine that collects the raw response time data.

invalid transitions
: is the number of invalid calls to this routine; meter_response_time implements a finite-state automaton that models a terminal user's interaction with the Multics system. The number of invalid transitions is the number of events that did not fit into this models. Typically, invalid transitions result from perturbations associated with process initiation.

Overhead
: is the CPU time consumed by the measurement routine, expressed as a percentage of total system CPU time, and as the average CPU time in milliseconds per call to this routine.

EXAMPLES

The following is an example of the information printed when the response_meters command is invoked with no control argument. (See Appendix A for a representation of the configuration deck used to create the system from which the metering saple was taken.)

```
Total metering time    0:15:30

WC  ---Thinks/--     ----Response Times by VCPU Range----  Load
    ---Queues---     -VCPU Range-     #  Avg    Avg   Resp   Control
         #  Avg       From    To    Int  VCPU   RT    Fact   Group

 0      66  2.74      0.00  0.50     78  0.05  0.37  6.78   Init
        79  0.14      1.00 10.00      1  1.09  2.63  2.42
                      ----- -----    79  0.07  0.40  5.89
```

```
  1      185  1.28     0.00  0.50     158  0.01  0.15 18.61  RTime
         220  0.13     ----- -----    158  0.01  0.15 18.61

  2      100 21.55     0.00  0.50     105  0.05  0.51  9.60  System
         104  0.07     0.50  1.00       3  0.81  2.92  3.58  SysAdm
                       1.00 10.00       5  2.24  6.65  2.97  OPR FED
                       ----- -----    113  0.17  0.85  4.98

  3     3410  5.06     0.00  0.50    4101  0.05  0.35  7.16  SysProg
        3524  0.09     0.50  1.00      97  0.70  3.79  5.44  SysDev
                       1.00 10.00      59  2.61 12.13  4.64
                      10.00 99.99       4 93.66 99.99  1.80
                       ----- -----   4261  0.19  0.75  4.02

  4      683  7.96     0.00  0.50     801  0.03  1.24 37.58  SEngr
         690  1.20     0.50  1.00       7  0.77 12.01 15.53
                       1.00 10.00       4  3.11 39.72 12.79
                      10.00 99.99       4 22.52 99.99 10.04
                       ----- -----    816  0.16  2.62 15.92

  5     1383  3.51     0.00  0.50    1139  0.04  0.23  5.26  HEngr
        1403  0.08     0.50  1.00      60  0.67  1.89  2.83
                       1.00 10.00      13  1.94  6.60  3.41
                       ----- -----   1212  0.10  0.38  4.02

  6      418 13.65     0.00  0.50     438  0.12  0.49  4.04  MktUS
         439  0.09     0.50  1.00      12  0.62  3.49  5.60  MktFor
                       1.00 10.00      11  2.37 10.15  4.28  MktEd
                      10.00 99.99       1 16.84 42.51  2.52
                       ----- -----    462  0.22  0.89  3.97

  7      185  9.04     0.00  0.50     228  0.07  1.21 18.54  DS-CC
         213  0.93     0.50  1.00      15  0.76  5.91  7.74
                       1.00 10.00       6  2.05 16.80  8.21
                      10.00 99.99       2 30.84 99.99  8.26
                       ----- -----    251  0.40  3.89  9.72

  8       18 17.38     0.00  0.50      23  0.13  0.93  7.16  OffAuto
          42  0.08     ----- -----     23  0.13  0.93  7.16

  9       14 95.89     0.00  0.50      10  0.17  1.01  5.97  Misc Mfg
          15  0.20     1.00 10.00       5  2.14  7.39  3.45
                       ----- -----     15  0.83  3.13  3.79

 10      106 16.56     0.00  0.50      92  0.09  0.65  7.30  Other
         113  0.10     0.50  1.00       7  0.67  3.37  4.99
                       1.00 10.00       4  1.75 13.21  7.56
                       ----- -----    103  0.19  1.33  6.84
```

```
   11     478  3.23     0.00  0.50     570  0.04  0.20  5.15  Special
          492  0.07     0.50  1.00       3  0.60  3.54  5.95
                        1.00 10.00       2  2.93 18.25  6.23
                        ----- -----    575  0.05  0.28  5.42

   All   7046  6.02     0.00  0.50    7743  0.05  0.46  8.97
         7334  0.22     0.50  1.00     204  0.69  3.66  5.29
                        1.00 10.00     110  2.42 12.18  5.04
                       10.00 99.99      11 49.39 99.99  3.92
                        ----- -----   8068  0.17  0.96  5.76

   90344 calls to meter_response_time        395 invalid transitions.
         Overhead =   0.09% (  0.046 ms./call)
```

Name: restore_pdt_access

SYNTAX AS A COMMAND

```
restore_pdt_access {Project_id}
```

FUNCTION

can be invoked after a crash to fix the ACLs on some system accounting data bases. The program ensures that the system and project administrators have the necessary access on the PDT and on the >sc1>proj_admin_seg segment and the >sc1>update directory.

ARGUMENTS

Project_id
: is the name of the project to be checked. If Project_id is supplied, only one project is repaired. Otherwise, all projects are checked.

Name: retrieve

SYNTAX AS A COMMAND

```
retrieve path {-control_args}
```

FUNCTION

retrieves specified storage system segments, directories and subtrees. It does this by copying them from a hierarchy dump tape into the hierarchy, possibly into different places in the hierarchy than those from which they were originally dumped. The retireve command calls the backup_load command to do the actual retrieving.

The retrieve command requires a retrieval control file, containing the pathnames of the objects to be retrieved. It calls backup_load once for each line in the control file. See "Notes on Format of a Retrieval Control File" below. Cross retrievals are allowed; i.e., objects can be retrieved into different places in the hierarchy than those specified on the dump tape. The retrieve command places its maps in the working directory and doesn't automatically dprint them. Quota on the retrieved directories is not force-set.

The retrieve command is one of the commands used for hierarchy reloading and retrieving of storage system segments and directories. The other commands are:

```
backup_load
reload (initializer command)
reload (Multics command)
reload_system_release
```

You should note that argument processing for all of the hierarchy backup commands is performed by a common argument processing procedure. The values of all arguments are remembered in static storage and remain in effect for the life of the process, unless changed by arguments given in subsequent invocations of backup commands. It should also be noted that the dumping commands and the reloading/retrieving commands are all part of the same hierarchy backup system, and argument values set by the dumping commands remain in effect for the reloading/retrieving commands and vice versa, unless overridden. However, dumping and reloading cannot be done in the same process; use the new_proc command between dumping and reloading. See "Notes on Default Arguments" below.

ARGUMENTS

path
: is the absolute pathname of a retrieval control file (see "Notes on Format of a Retrieval Control File" below). This argument is required. It can be given anywhere on the command line.

CONTROL ARGUMENTS

-all
: causes segments to be retrieved from the tape regardless of their date/time dumped. This is the default. This control argument overrides a previously given DATE argument.

-brief_map, -bfmap
: creates a map file that lists the processed entries.

-control path
: indicates that path is a hierarchy retrieval control file pathname. See "Notes on Format of a Retrieval Control File" below.

-debug
: disables those hphcs_ calls that set quotas and transparency switches.

-destination STR, -ds STR
: specifies a destination for printing maps and error file. The default is "incremental" for maps and "error file" for error files.

-error_of
: writes error messages into a file rather than printing them. The name of the error file is printed when the first error is encountered. This is the default.

-error_on
: writes error messages on the user's terminal.

-first
: prevents searching a tape for additional copies of a requested segment or subtree after the first copy has been retrieved.

-header STR, -he STR
: specifies a heading for printing maps and error files.

-last
: indicates that the last copy of a given segment or subtree on a tape or set of tapes is to be retrieved. This is the default.

-map
: writes a list of the segments and directories processed into a file. This is the default.

-nodebug
: enables hphcs_ calls to set quotas and the transparency switches. This is the default.

-nomap
: inhibits listing of the names of processed segments and directories.

-noprimary, -npri
  uses each pathname as given. The default is -primary.

-noqcheck
  causes the hierarchy reload to be done with quota checking suspended. Access to hphcs_ is required. This is the default.

-noquota
  inhibits resetting of quotas. See -quota. This is the default.

-noreload
  inhibits actual hierarchy reloading of segments into the hierarchy. This control argument can be used with -map to create a table of contents of the tape. The -noreload control argument also causes the names that would have been reloaded to be put into the map.

-nosetlvid
  inhibits the setting of the logical volume identifiers for each directory to be reloaded.

-notrim
  inhibits deletion of entries in a directory. Entries can only be added or modified. This is the default.

-operator STR
  indicates that STR is the user's name or initials (up to 16 characters in length).

-primary, -pri
  replaces all directory names in each pathname with the primary names. This is the default.

-pvname STR
  indicates that segments and directories can only be retrieved onto the physical volume specified by STR.

-qcheck
  causes quota restrictions to be enforced during the reload. *

-quota
  causes the quotas on directories being reloaded to be set to the values they had when the directories were dumped. Access to hphcs_ is required.

-reload
  enables actual reloading of segments into the hierarchy. This is the default.

-request_type STR, -rqt STR
  specifies an output request type for printing maps and error files. Available request types can be listed by using the print_request_types command (described in the *Multics Commands and Active Functions* manual, Order No. AG92). The default is "printer".

-setlvid
: enables setting of the logical volume identifier for reloaded entries inferior to each directory reloaded. This is the default.

-trim
: enables deletion of all entries in a directory not found in the copy of that directory being reloaded. This causes entries deleted from an earlier version of the directory to be deleted when a later version is reloaded. It has effect only in the case of a directory that is both on the tape and in the hierarchy. This is the default.

DATE
: an argument beginning with a character other than "-", or ">" is assumed to be a date in a format acceptable to the convert_date_to_binary_ subroutine. If it can be converted successfully, then the hierarchy retriever only retrieves segments and directories dumped at or after the given date/time.

NOTES ON DEFAULT ARGUMENTS

The values of arguments given to any of the hierarchy backup commands are remembered in static storage and remain in effect for the life of the process, unless explicitly changed during the invocation of a subsequent backup command.

The following defaults are in effect for the reloader and retriever before any backup commands are given; they are not, however, reset to these values at the start of each backup command, except as noted below.

```
-all
-error_of
-map
-nodebug
-nohold
-noquota
-primary
-reload
-setlvid
-trim
```

The following defaults are set automatically at the time the respective commands are executed:

```
reload (initializer command), reload (Multics command),
   reload_system_release:
   -quota
   -trim

retrieve:
   -all
   -noquota
   -notrim

All of the above commands:
   -map
```

NOTES ON FORMAT OF A RETRIEVAL CONTROL FILE

The hierarchy retrieval is controlled by an ASCII segment containing one line for each object to be retrieved. A line can contain a single pathname or two pathnames separated by an equal sign. The left-hand side specifies the segment or directory sought and the right-hand side, if present, specifies the new name under which that entity is to be retrieved. The sought pathname must begin with a > and end with either an entryname or the characters >**. If an entryname is specified, a single object of that name is retrieved.

If >** is specified, the entire directory hierarchy, beginning at the point indicated in the pathname, is retrieved. In this case, the right_hand pathname, if present, ends in the name of the directory under which these entries are to be reloaded. For example:

```
>udd>one_dir>**=>udd>two_dir
```

If a new name is specified on the right, it can be either a pathname or an entryname. If an entryname is given, the single object found is loaded with its former pathname and the new entryname.

If two pathnames are specified, both are checked against the current hierarchy and a new pathname consisting only of the primary entryname is created. This new pathname, as well as the original, is then used in searching the hierarchy. For example, >udd>sd is translated into >user_dir_dir>SysDaemon and both versions are sought. Primary names are used unless the -noprimary control argument is in effect.

A hierarchy retrieval control file can contain a maximum of 256 lines.

EXAMPLES

A hierarchy retrieval control file containing the line:

```
>udd>Multics>**
```

searches the tape for directories and segments whose first two pathname components are >user_dir_dir>Multics. These items are retrieved as found.

A hierarchy retrieval control file containing the line:

```
>ldd>a>b>c
```

searches the tape for the segment >library_dir_dir>a>b>c. This item is retrieved as found.

A hierarchy retrieval control file containing the line:

```
>ldd>a>b=c
```

searches the tape for the segment >library_dir_dir>a>b. This item is retrieved under the name >ldd>a>c.

A hierarchy retrieval control file containing the line:

```
>ldd>x>y>**=>ldd>z>y
```

searches the tape for directories and segments whose first three pathname components are >library_dir_dir>x>y. These items are retrieved in the subtree >ldd>z>y.

Name: ring_zero_dump, rzd

SYNTAX AS A COMMAND

```
rzd segname {offset} {length} {-control_args}
```

SYNTAX AS AN ACTIVE FUNCTION

```
[rzd segname {offset} {-control_args}]
```

FUNCTION

prints or returns the locations of the specified ring 0 or user ring segment in full-word octal format.

ARGUMENTS

segname
: is an octal segment number, the name of a ring 0 segment, or a pathname. To specify a segment name that consists entirely of octal digits, precede the name by -name.

offset
: is the (octal) offset of the first word to be dumped. If you omit both offset and length, the entire segment is dumped. If you use -ep, omit offset.

length
: is the (octal) number of words to be dumped. If you supply offset and omit length, one word is dumped.

CONTROL ARGUMENTS

-4bit
: prints or returns a translation of the octal or hexadecimal dump based on the Multics unstructured four-bit byte. The translation ignores the first bit of each nine-bit byte and uses each of the two groups of four bits remaining to generate a digit or a sign.

-address, -addr
: prints the address (relative to the base of the segment) with the data. (Default)

-bcd
: prints the BCD representation of the words in addition to the octal or hexadecimal dump; in the active function, returns BCD. There are no nonprintable BCD characters, so take periods literally.

-block N, -bk N
: dumps words in blocks of N words separated by a blank line. The offset, if being printed, is reset to the initial value at the beginning of each block.

-character, -ch, -ascii
: prints the ASCII representation of the words in addition to the octal or hexadecimal dump; in the active function, returns ASCII. Characters that cannot be printed are represented as periods.

-ebcdic8
: prints the EBCDIC representation of each eight bits in addition to the octal or hexadecimal dump; in the active function, returns eight-bit EBCDIC. Characters that cannot be printed are represented by periods. If an odd number of words is requested for a dump, the last four bits of the last word do not appear in the translation.

-ebcdic9
prints the EBCDIC representation of each nine-bit byte in addition to the octal or hexadecimal; in the active function, returns nine-bit EBCDIC. Characters that cannot be printed are represented by periods.

-entry_point name, -ep name
specifies that the offset of the first word to be dumped is relative to the location defined by the externally available symbol "name." You can use this control argument only for object segments (created by a compiler or the create_data_segment command). If you supply offset, omit -ep.

-header, -he
prints a header line containing the pathname (or segment number) of the segment being dumped as well as the date or time printed. (Default: to print a header only if the entire segment is being dumped, i.e., if you specify neither the offset nor the length argument)

-hex8
prints the dumped words in hexadecimal with eight hexadecimal digits per word, rather than 12 octal digits per word. Each pair of hexadecimal digits corresponds to the low-order eight bits of each nine-bit byte.

-hex9
prints the dumped words in hexadecimal with nine hexadecimal digits per word, rather than octal with 12 octal digits per word.

-long, -lg
prints eight words on a line. You can't use -long with -4bit, -bcd, -character, -ebcdic8, -ebcdic9, or -short. (Its use with these control arguments, other than -short, results in a line longer than 132 characters.) (Default: four)

-name path, -nm path
indicates that path is the name of a ring 0 segment or a pathname, even though it may look like an octal segment number.

-no_address, -nad
does not print the address.

-no_header, -nhe
does not printing the header line, even though the entire segment is being dumped.

-no_offset, -nofs
does not print the offset. (Default)

-offset N, -ofs N
prints the offset, relative to N words before the start of data being dumped, along with the data. If you supply no N, zero is assumed.

-rest
: prints from any given offset (specified by -ofs or the offset argument) to the end of the segment.

-short, -sh
: compacts lines to fit on a terminal with a short-line length. Single spaces are placed between fields, and only the two low-order digits of the address are printed, except when the high-order digits change. This shortens output lines to less than 80 characters.

ACCESS REQUIRED

You need no access to phcs_ gate for those segments accessible through the ring_zero_peek_ subroutine.

NOTES

You can give only one of -4bit, -bcd, -character, -ebcdic8, or -ebcdic9.

As an active function, rzd returns only one word of information, which is located at offset, with the segment. If you supply -4bit, -bcd, -character, -ebcdic9, -ebcdic8, -hex8, or -hex9, the information is returned in the specified format only. All other arguments are ignored in the active function.

Name: salvage_dir

SYNTAX AS A COMMAND

```
salvage_dir dir_path {output_path} {-control_args}
```

FUNCTION

verifies and/or rebuilds one directory. You can use it only in ring 4.

The command "x repair" causes one or more daemons to be logged in to perform an online salvage using salvage_dir. Preferably, perform an online salvage this way.

ARGUMENTS

dir_path
: is the pathname of the directory being salvaged.

output_path
: is the pathname of the segment or multisegment file (MSF) to which the salvager messages should be appended. If the segment does not exist, it is created. If you don't specify it, output goes to user_output.

CONTROL ARGUMENTS

-check_vtoce
: checks VTOC entries for all branches in the directory. These checks update the permanent information in the VTOC entry and detect connection failures.

-compact
: specifies rebuilding of the directory if one or more pages are recovered.

-delete_connection_failure, -dcf
: deletes branches for segments that do not have corresponding VTOC entries. The default is not to delete such branches. This control argument updates permanent VTOC entry information. This argument makes subsequent volume retrieval of the lost data more expensive because both the data and the branch must be recovered.

-rebuild
: forces rebuild of the directory.

LIST OF ADDITIONAL CONTROL ARGUMENTS

The following control arguments are used for software debugging:

-debug, -db
: prints additional trace information. Because this information includes offsets in the original directory, use -dump also.

-dump path
: specifies that a copy of the directory should be placed in a segment in the directory specified by path. The copy is created only when certain error conditions are detected during salvage.

EXAMPLES

The following command salvages all directories and appends the diagnostic messages to the segment salv_messages in >dumps:

```
walk_subtree "salvage_dir [wd] >dumps>salv_messages" -msf -priv
```

Name: salvage_mstb

SYNTAX AS A COMMAND

```
salvage_mstb path {N}
```

FUNCTION

rebuilds an overloaded MSTB, or recovers the contents of a damaged MSTB.

ARGUMENTS

path
    is the pathname of the MSTB to be recovered.

N
    is the number of entries to make in the new table. The default is three times the number of used entries in the old table, making the new MSTB 33% full.

NOTES

Never salvage a "live" MSTB. A salvage operation should be performed on a saved copy only. The PNT and the mail table should be salvaged in special session, then moved back in place before the answering service is restarted.

The URF can be salvaged whenever it is not in use by any process.

Entries in an MSTB are accessed using a hashing algorithm. For most efficient operation, it is recommended that the PNT be kept between 30% and 50% full and the URF between 50% and 70% full.

Name: save_history_registers

SYNTAX AS A COMMAND

```
save_history_registers {state} {-control_args}
```

FUNCTION

allows a user to save processor history registers upon each occurrence of a signalable fault in the signalers stack frame. By default, the history registers are not saved, and the history register block in the signalers stack frame is set to all zeros.

ARGUMENTS

state
    can be either "on" or "off." If state is not specified, it is off.

CONTROL ARGUMENTS

-priv
: specifies manipulation of the per-system state by directing the state and -print arguments to operate on the per-system history register save switch, wired_hardcore_data$global_hregs. When set, this switch causes all processes to save their history registers upon each occurrence of a signalable fault in the signalers stack frame. If -priv is not specified, then the state and -print arguments operate on pds$save_history_regs, the per-process history register save switch of the process executing this command.

-print, -pr
: displays the current state of the history register save switch if the state argument is not specified; if the state argument is specified, the state of the switch is displayed before the new state is applied.

ACCESS REQUIRED

When -priv is used, hphcs_ access is required.

Name: scavenge_vol

SYNTAX AS A COMMAND

```
scavenge_vol {pvname} {-control_args}
```

SYNTAX AS AN ACTIVE FUNCTION

```
[scavenge_vol {pvname} {-control_args}]
```

FUNCTION

invokes the Scavenger to scavenge one or more physical volumes. The Scavenger examines and validates VTOCEs, deletes per-process and per-bootload VTOCEs from previous bootloads, recovers lost VTOCEs, recovers lost records, and checks for reused addresses (two VTOCES claiming the same record address). See the *Multics System Maintenance Procedures* manual, Order No. AM81, for a description of scavenging.

If no control arguments are specified, no default actions are taken.

The initializer command "x scav" logs in a daemon to perform a scavenge using the scavenge_vol command.

ARGUMENTS

pvname
: is the name of the physical volume to be scavenged. It must be part of a mounted logical volume. This argument is required unless the -all or -lv control argument is specified.

CONTROL ARGUMENTS

-all, -a
: scavenges, in turn, all mounted physical volumes.

-auto
: controls scavenging of physical volumes when the -lv or -all control argument is specified. When this option is used, those physical volumes with volume inconsistencies are scavenged.

-check
: validates the command control arguments and prints a list of physical volumes that would be scavenged as a result of the command line input, but does not actually perform the scavenge.

-debug
: should be used only by systems programmers debugging the Scavenger. It causes additional information to be printed on the bootload console during the scavenge.

-dump
: records the VTOCE image of any VTOCE found inconsistent into the syserr log. The image recorded is the VTOCE prior to any correction by the Scavenger.

-lv lvname
: scavenges, in turn, all physical volumes belonging to the logical volume specified by lvname. The logical volume must be mounted.

-no_optimize, -nopt
: disables optimization of Scavenger processing. This causes the Scavenger to take longer, but reduces its effect on other users of the system. With this option, VTOCE read-ahead is disabled, and the Scavenger periodically lowers its traffic control priority.

ACCESS REQUIRED

Access to the hphcs_ gate is required.

NOTES

The scavenge of each volume is done entirely in ring 0. Any error messages are recorded into the syserr log. It is possible to interrupt a scavenge of a volume with the QUIT key, but it cannot be restarted afterwards with the start or program_interrupt commands.

When the scavenge_vol command is invoked as an active function, it returns a list of physical volumes that would be scavenged, separated by spaces. The -check control argument is the default for active function usage, i.e., scavenging is not done.

Name: sc_command

SYNTAX AS A COMMAND

```
sc_command initializer_command {arguments}
```

FUNCTION

allows initializer commands to be executed from admin mode.

NOTES

This command is used in exec_coms such as system_start_up.ec and admin.ec, and in some uses of the send_admin_command (sac) command. In these cases, the initializer process is executing in admin mode (see the admin command earlier in this section) and expects normal Multics commands. Any initializer commands to be executed in these cases must be executed by using sc_command.

However, because sc_command is executed from within admin mode (whether typed by the user or sent via the sac command), the initializer_command argument cannot be a command that re-enters admin mode (for example, the exec ["x"] command). In order to accomplish this, the x command can be replaced by "ec admin {args}".

Administrators should avoid using the sac command to send initializer commands that ask questions, because the questions will be asked on the bootload console. Use of the answer command solves this problem for Multics commands sent via sac, but the answer command cannot be used to supply answers to initializer commands, since most initializer commands ask their question using a different interface from normal Multics commands.

EXAMPLES

```
In admin.ec:  sc_command reply prta driver

From an administrator:  sac "sc_command unbump Smith SysAdmin"
```

Name: sector_to_record

SYNTAX AS A COMMAND

```
sector_to_record record_no {device_name}
```

FUNCTION

converts an octal sector address to a Multics record number.

ARGUMENTS

record_no
: is the octal Multics record number.

device_name
: is a valid device name (e.g., "m400", "m451").

Name: send_admin_command, sac

SYNTAX AS A COMMAND

```
sac {-control_args} commandline
```

FUNCTION

sends a command line to the initializer process for execution.

ARGUMENTS

commandline
: is the command line to be sent to the initializer. To send special characters you must enclose them (or the whole command line) in quotes. If the first character of the command is a "-" character, the "-string" control argument must be used.

CONTROL ARGUMENTS

-address MAIL_ADDRESS
: specifies that mail sent for notifications is to be sent to the mail address specified by MAIL_ADDRESS. In most cases, the MAIL_ADDRESS can be in the form Person_id.Project or Person_id. (Type "help mail_addresses.gi" for additional information on mail address specification).

-brief, -bf
: suppresses the printing of informational messages.

-brief_query, -bfqy
    specifies that a brief query is to be posed before the command is executed.

-long, -lg
    prints the messages suppressed by -brief. This is the default.

-no_notify_by_message, -nntmsg
    specifies that the user is to receive no interactive messages concerning the execution of the command, except in the special case that -no_wait is specified and the execution of the command is denied due to lack of access.

-no_query, -nqy
    specifies that no query is to be posed before sending the command line. This is the default.

-no_wait, -nwt
    specifies that the command return to the listener as soon as the commandline has been sent to the initializer.

-notify_by_mail, -ntmail
    specifies that the user is to be sent mail in the following cases: (1) -no_wait is specified, and execution of the command line is denied due to lack of access. (2) the command completes execution.

    By default, the mail will be sent to the destination specified in the user's mail table entry. If no destination is found there, the full User.Project specification will be used. The default may be overridden with the -address control argument. If the command is executed, the mail will contain all output produced on the error_output, user_output, and user_i/o switches during the command execution.

-notify_by_message, -ntmsg
    specifies that the user is to receive a brief notification by interactive message in the following cases: (1) -no_wait is specified, and execution of the command line is denied due to lack of access. (2) The command completes execution.

    If possible, the user's mail table entry will be used to determine the delivery address. If not, the full User.Project specification will be used. The -address control argument does not apply to interactive messages.

-query, -qy, -long_query, -lgqy
    specifies that the command line is to be displayed on the terminal and that the user is to be queried before the command line is executed. This control argument is useful for validating the effects of abbrevs and active functions.

-string
    specifies that the rest of the command line is to be interpreted as the command to be executed, whether or not it begins with a "-".

-wait WAIT_TYPE, -wt WAIT_TYPE
specifies that the command must wait for a specified event before returning to the listener. WAIT_TYPE can be one of the following values:

start
specifies that the command is to wait for the initializer to acknowledge receipt of the command line before returning to the listener.

finish
specifies that the command is to wait for the inititalizer to finish execution of the command line before returning to the listener. This is the default.

ACCESS REQUIRED

RW access to >sc1>admin_acs>send_admin_command.acs is required to use this command.

Name: send_daemon_command

SYNTAX AS A COMMAND

```
send_daemon_command    request    {Person_id.Project_id}    source_id
    {command_line}
```

FUNCTION

allows control over daemon processes from normal user processes. This command is used to login/logout the daemon, send a quit signal to the daemon, or send a reply to the daemon.

ARGUMENTS

request
can be one of the following requests.

login
login in the specified daemon.

logout
logout the specified daemon.

quit
send a quit signal to the specified daemon.

reply
send a reply to the specified daemon.

Person_id.Project_id
: is the user_id of the daemon process to be logged in. This argument must be supplied for the login request and is valid for that request only.

source_id
: is the message coordinator source name associated with the daemon (e.g., bk, vinc, ut, scav).

command_line
: is the command line to be submitted to the daemon via the reply request or optional login control arguments for the login request. This argument is not valid for the logout or quit requests. If the command line contains special characters, they must be enclosed in quotes (or the whole command line can be enclosed in quotes).

ACCESS REQUIRED

If the validate_daemon_commands keyword in installation_parms is set "off," rw access to >sc1>admin>send_daemon_command.acs is required to use this command.

If the validate_daemon_commands keyword in installation_parms is set "on," then the following access is required to use the send_daemon_command.

1. The login and logout requests require "e" (control) access on the acs segment associated with the daemon source_id. Additionally, the login request requires that the daemon identified by Person_id.Project_id have "d" (daemon) access to the acs segment associated with the daemon source_id.

2. The quit request requires "q" (quit) access on the acs segment associated with the daemon source_id.

3. The reply request requires "r" (reply) access on the acs segment associated with the daemon source_id).

NOTES

The send_daemon_command cannot be used within the sac command (use sc_command).

EXAMPLES

```
send_daemon_command login rp Repair.SysDaemon -auth system_high

send_daemon_command quit rp

send_daemon_command reply rp pwd

send_daemon_command logout rp
```

Name: send_ips

SYNTAX AS A COMMAND

```
send_ips process_id signal_name
```

FUNCTION

sends an IPS signal to a process. It is a command interface to the hphcs_$ips_wakeup subroutine entry point.

ARGUMENTS

process_id
: is a 12-digit octal number specifying the ID of the process that is to receive the signal. You can omit leading zeros from process_id.

signal_name
: is name of one of the system-defined IPS signals; it can be up to 32 characters long. The signal names must be defined in sys_info_$ips_mask_data. Presently the defined signal names are quit, alrm, neti, cput, trm_, sus_, wkp_, pgt_, system_shutdown_scheduled_, and dm_shutdown_scheduled_.

ACCESS REQUIRED

You need access to the highly privileged hphcs_ gate.

NOTES

No error message is given if you supply an undefined IPS signal or a nonexistent process.

The process_id active function is a convenient way of obtaining a process_id, given a User_id or channel name.

Name: send_wakeup

SYNTAX AS A COMMAND

```
send_wakeup process_id event_channel {event_message}
```

FUNCTION

sends an IPC wakeup to a process. It is a command interface to the hcs_$wakeup subroutine entry point.

ARGUMENTS

process_id
    is a 12-digit octal number specifying the ID of the process that is to receive the wakeup. Leading zeros can be omitted.

event_channel
    is a 24-digit octal number specifying the event channel over which the wakeup is to be sent. Leading zeros can be omitted.

event_message
    is an optional 72-bit event message, given as either a 24-digit octal number or an eight-character ASCII string. The default is all zero bits. Leading zeros or trailing blanks can be omitted. The event message is assumed to be in octal form if it contains only octal digits.

NOTES

Nonexistent processes and event channels of invalid format are diagnosed; however, validly formed but nonexistent event channels are not diagnosed.

The process_id active function is a convenient way of obtaining a process id, given a User_id or channel name.

Name: set_dir_quota

SYNTAX AS A COMMAND

```
set_dir_quota path1 quota1 ...  pathN quotaN
```

FUNCTION

places an arbitrary secondary storage quota for directories on a specified directory. Incorrect use of this command can make portions of the hierarchy unusable because of quota violations.

ARGUMENTS

pathi
    is the name of the directory on which the directory quota is to be placed. The active function -wd can be used to specify the working directory.

quotai
    is the directory quota in 1024 word pages.

ACCESS REQUIRED

The user must have access to the highly privileged gate hphcs_ to use this command.

NOTES

No permission in the directory is required to use this command. It is not necessary that the new directory quota be greater than the current number of pages being used by directories in this directory. This command attempt to set a terminal directory quota even if it is set to zero. This command does not cause the inferior counts of the superior directory to be updated.

Name: set_flagbox

SYNTAX AS A COMMAND

```
set_flagbox flagbox_variable
```

FUNCTION

sets the value of a specified flag in the BCE/Multics communication area.

ARGUMENTS

flagbox_variable
is one of the valid flagbox variables listed below:

bce_command command
is a command to be invoked by BCE when it reaches a command level. This command is set so that if the system should crash, this command is automatically executed. Refer to the *Multics System Maintenance Procedures* manual, Order No. AM81 for more details.

keyword value
may be either a number from 1 to 36 or the corresponding name of one of the flagbox flags as given below. The names of the flagbox flags are:

1	auto_reboot
2	booting
3	bit3
4	rebooted
5	unattended
6	bit6
7	bit7
8	bit8
9	bit9
10	bit10

11	bit11
12	bit12
13	bit13
14	bit14
15	bit15
16	bit16
17	bit17
18	bit18
19	bit19
20	bit20
21	bit21
22	bit22
23	bit23
24	bit24
25	bit25
26	bit26
27	bit27
28	bit28
29	bit29
30	bit30
31	bit31
32	bit32
33	bit33
34	bit34
35	bit35
36	bit36

value
is either true or false.

ACCESS REQUIRED

Privileged access to hphcs_ is required to use this command.

Name: set_log_history_dir

SYNTAX AS A COMMAND

```
set_log_history_dir log_seg_path history_dir_path
```

FUNCTION

changes the pathname of older log segments recorded in a log segment's header.

ARGUMENTS

log_seg_path
: specifies the log segment history_dir pathname that is to be changed.

history_dir_path
: specifies the new history_dir log segment pathname. If this pathname is not the name of an existing directory, you are queried whether or not this change is to be made.

ACCESS REQUIRED

You need write access to the selected log segment.

Name: set_mc_message_limits

SYNTAX AS A COMMAND

```
set_mc_message_limits {-control_args}
```

FUNCTION

controls the printing of messages by the message coordinator. The system administrator can control the size of the burst and the duration of the pause with this command (see "Notes").

CONTROL ARGUMENTS

-count N, -ct N
: sets the size of a message burst to N messages. N must be a decimal integer greater than zero. (Default: 20 messages)

-delay N, -dly N
: sets the interval between message bursts to N seconds. N must be a decimal integer greater than zero. (Default: 5 seconds)

-print, -pr
: displays the current message burst size and delay values.

NOTES

Execute this command in admin mode. You can include it in the system_start_up.ec.

Name: set_mos_polling_time

SYNTAX AS A COMMAND

```
set_mos_polling_time {N}
```

FUNCTION

sets the time interval used by the system for polling MOS memories to check for and log EDAC errors.

ARGUMENTS

N
: is a decimal integer representing the time in minutes between MOS memory polls. If omitted, the command prints the current polling interval. If N is 0, MOS memory polling is disabled.

ACCESS REQUIRED

You must have re access to hphcs_.

NOTES

MOS memory polling is disabled when the system is initialized. Use this command to enable it.

MOS memory polling should not be enabled on systems that have core memories unless the TEST/NORMAL switch on the maintenance panel of the memory, not of the controller, is set to TEST. If this switch is set to NORMAL, spurious errors may be logged for the memory.

Name: set_proc_required, sprq

SYNTAX AS A COMMAND

```
sprq {tag1}...{tag2}...{tagN} {-control_arg}
```

FUNCTION

restricts processes to run only on specified CPUs. You can use it to specify the set of CPUs on which the invoking process can be run and the default set of CPUs for all processes that have not requested specific CPUs.

ARGUMENTS

tagN
: is the tag for one of the CPUs in the group being specified. It can be one of the letters a through h, or A through H. If you give no tag, the group is assumed to contain all CPUs (tags a through h). If you give -priv, at least one tag is required.

CONTROL ARGUMENTS

-priv
: indicates that the group of CPUs specified is to become the default group for processes that have not requested specific CPUs. If omitted, the group of CPUs specified applies only to the invoking process.

ACCESS REQUIRED

You need access to phcs_. If you give -priv, you need access to hphcs_.

NOTES

If none of the CPUs specified are online, an error message is printed and the command has no effect.

EXAMPLES

The command line

```
   set_proc_required A B
```

restricts the requesting process to run only on CPUs A and B.

The command line

```
   set_proc_required
```

allows the requesting process to run on any online CPU.

The command line

```
   set_proc_required A B E -priv
```

restricts all processes that have not requested specific CPUs to run only on CPUs A, B, and E.

The command line

```
   set_proc_required  -priv
```

allows all processes that have not requested specific CPUs to run on any online CPU.

Name: set_process_audit_flags

SYNTAX AS A COMMAND

```
set_process_audit_flags {flags_str}
```

FUNCTION

sets the process security audit flags to the supplied value, turning on the specified flags and off all others.

ARGUMENTS

flags_str
: is an audit flags string acceptable to convert_access_audit_flags_. If you omit it, the command enters a prompt loop in which you are asked for an audit flags string. A period entered alone on a line causes the loop to exit.

ACCESS REQUIRED

You must have re access on system_privilege_ and access_audit_gate_ and r to >udd>SysAdmin>admin>sys_admin_data if you use the "default" keyword.

NOTES

For a description of the process audit flags, see the new_user command in this section and the section "Security Auditing" in the *System Administration Procedures* (AK50).

This command is meant for experimentation with security audit control flags by the system security administrator. Permanent settings for processes should be placed in the appropriate PNT/SAT entries with the new_user and edit_proj commands.

The special keywords "none," "all," and "default" are also recognized for the audit flags string. They specify

```
   "none"     all audit flags turned off
   "all"      all audit flags turned on
   "default"  audit flags set to default found in sys_admin_data.
```

See also the display_process_audit_flags and edit_process_audit_flags commands.

EXAMPLES

```
  set_process_audit_flags default
  display_process_audit_flags
  fsobj=N/R,fsattr=N/R,rcp=N/R,admin=R/R,special=R/R,...
     ...other=N/R,admin_op,priv_op,fault,^small_cc,^moderate_cc

  set_process_audit_flags rcp=n/r,admin=r/,special=/r
  display_process_audit_flags
  fsobj=N/N,fsattr=N/N,rcp=N/R,admin=R/N,special=N/R,...
     ...other=N/N,^admin_op,^priv_op,^fault,^small_cc,^moderate_cc

  set_process_audit_flags default
  current flags: fsobj=N/R,fsattr=N/R,rcp=N/R,admin=R/R,...
    ...special=R/R,other=N/R,admin_op,priv_op,...
       ... fault,^small_cc,^moderate_cc
  enter flags:   fsobj=r/
  new flags:     fsobj=R/R,fsattr=N/R,rcp=N/R,admin=R/R,...
    ...special=R/R,other=N/R,admin_op,priv_op,...
       ...fault,^small_cc,^moderate_cc
  enter flags:   fsobj=/m
  new flags:     fsobj=R/M,fsattr=N/R,rcp=N/R,admin=R/R,...
    ...special=R/R,other=N/R,admin_op,priv_op,...
       ... fault,^small_cc,^moderate_cc
  enter flags:   ^admin_op,^fault,other=ma/ma
  new flags:     fsobj=R/M,fsattr=N/R,rcp=N/R,admin=R/R,...
    ...special=R/R,other=MA/MA,^admin_op,priv_op,...
       ... ^fault,^small_cc,^moderate_cc
  enter flags:   .
```

Name: set_quota, sq

SYNTAX AS A COMMAND

```
sq path1 quota1 {... pathN quotaN}
```

FUNCTION

sets the segment record quota of a specified directory, without affecting, or being limited by, the quota of the superior directory. It is intended for use on the root directory only.

ARGUMENTS

pathN
: is the name of the directory on which the quota is to be set.

| quotaN
is the quota in 1024-word records to be set on the directory.

ACCESS REQUIRED

You need access to hphcs_. You need no permission in the directory whose quota is being set.

NOTES

No need that the new quota be greater than the current number of records being used by this directory. This command attempts to set a terminal quota even if it is set to zero. It does not update the inferior counts of the superior directory. Use it preferably only on the root, and transfer quota to inferior directories using move_quota.

EXAMPLES

To set quota on the root directory to allow 50,000 words of storage, type:

```
set_quota > 50000
```

Name: set_sons_volume

SYNTAX AS A COMMAND

```
set_sons_volume path LVNAME
```

FUNCTION

sets the logical volume on which the segments of a directory reside.

ARGUMENTS

path
    is the pathname of the directory whose sons logical volume ID is to be set.

LVNAME
    is the name of the logical volume.

ACCESS REQUIRED

You need access to hphcs_.

NOTES

This is a highly privileged command, which defeats the normal quota management for logical volumes. Normally, this command is only used in system_start_up.ec to set the logical volume for the directory >pdd. This operation is valid only on an empty directory. The usual method of creating a directory with a specific "sons logical volume" (a master directory) is to use the Multics create_dir command (see the Commands manual, AG92).

Name: set_special_password

SYNTAX AS A COMMAND

```
set_special_password special_id {-control_args}
```

FUNCTION

sets the password required for special (restricted) activities. Use it only in the accounting_start_up exec_com.

ARGUMENTS

special_id
is the special activity identifier. Currently, only one such identifier is defined, as follows:

operator_admin_mode
specifies operator "admin" mode.

CONTROL ARGUMENTS

-none
sets the specified identifier to have no password. (Setting a value of -none for operator_admin_mode permits admin mode to be entered without a password.)

-password PASSWORD
sets the password PASSWORD.

-prompt
queries you twice for the password. (Default)

Name: set_system_audit_flags

SYNTAX AS A COMMAND

```
set_system_audit_flags -control_args
```

FUNCTION

sets the system parameters that control auditing of user access to system resources.

CONTROL ARGUMENTS

| -covert_channel auth, -cch auth
specifies that operations that can be used as covert channels are to be audited. The value specified for auth must be a valid AIM authorization string. Sending processes whose authorization is greater than auth are audited. Receiving processes are audited regardless of authorization. (See "Notes on Covert Channels," below.)

-no_covert_channel, -ncch
specifies that auditing of covert channel operations is to be turned off.

-no_successful_access, -nsa
specifies that there is to be no auditing of successful operations.

-no_unsuccessful_access, -nusa
specifies that there is to be no auditing of operations on system resources that are denied by the system.

-successful_access access_class, -sa access_class
: specifies that all successful operations on system resources are to be audited. Only those operations involving resources with an AIM access class equal to or greater than access_class are audited.

-unsuccessful_access access_class, -usa access_class
: specifies that all operations on system resources that are denied by the system are to be audited. Only those operations involving resources with an AIM access class equal to or greater than access_class are audited.

ACCESS REQUIRED

You need access to the highly privileged gate hphcs_.

NOTES ON COVERT CHANNELS

A covert channel is a means for passing information between two processes that is outside of the normal system mechanisms for data transmission. The Multics operating system was enhanced to identify operations that could be exploited as covert channels. The information generated by -cch provides administrators with information concerning the processes and data involved in the operation. The Multics literature contains no description of those operations identified as potentially exploitable as covert channels.

NOTES

Some parameter settings can result in degradation of system performance (for example, if you audit successful operations, including those involving resources with the lowest AIM access class).

Because some system operations can be catagorized in more than one way, they can be audited in more than one way. For instance, a particular operation could be audited for successful access and as a covert channel. Additionally, the per-process audit flags (see ed_installation_parms) can be set for additional auditing of operations.

This command changes only those auditing parameters as specified in the control arguments. Other parameters are not modified.

See also display_system_audit_flags.

Name: set_system_console

SYNTAX AS A COMMAND

```
set_system_console {console_name} {-control_args}
```

FUNCTION

controls the configuration of system consoles.

ARGUMENTS

console_name
: is the name of the console as it is specified in the configuration deck. If don't give it, the bootload console is assumed.

CONTROL ARGUMENTS

-crash
: specifies that the system should crash in case of console recovery failure.

-reset
: forcibly resets the bootload console as well as the oc_data database. If the console specified by console_name is not the bootload console, no action is taken.

-run
: specifies that the system should continue running in the event of console recovery failure. It overrides -crash.

-state STATE
: changes the operational state of the specified console to STATE, where STATE must be one of the following:

on
: makes the specified console the bootload console and the primary recipient of I/O. If a bootload console is currently assigned, it is made an alternate console.

alternate, alt
: makes the specified console an alternate console. In case of a bootload console failure, the first alternate console that appears in the configuration deck is selected as the bootload console.

io
: specifies that this console exists, but it is not to be used as an alternate console. If this is the bootload console, it is unassigned.

inop
: specifies that this console is inoperative. If this is the bootload console, it is unassigned.

off
: specifies that this console is being deconfigured and should not be used by the console software. This state cannot be used for the bootload console.

WARNING

When the bootload console is deconfigured with the set_system_console command, Multics will crash if the Crash on Console Recovery Failure option has been selected, either by using the -crash control argument above or by presence of the ccrf parameter in the config_file. The following message will be printed:

```
ocdcm_ (reconfigure):  Bootload console deconfigured with CCRF set.
```

NOTES

If the bootload console's state is set to IO or INOP and no other console is assigned as the bootload console, Multics will send subsequent output to the Message Coordinator. If no Message Coordinator is available, Multics will act with respect to the specification of the ccrf parameter.

Although it is possible to run the system from the Message Coordinator, it is not recommended that sites run without an active bootload console for extended periods of time. During the period that there is no bootload, console sites will be restricted to running only those commands executable at an initializer terminal. Failure of the FNP to which the initializer terminal is attached while running without a bootload console could produce severe problems.

Name: set_system_priv

SYNTAX AS A COMMAND

```
set_system_priv privilege_name1 {...privilege_nameN}
```

FUNCTION

is used to turn on and off the system privileges that allow the process to function outside the restrictions of the AIM access controls. Whenever the privileges are turned on, the user must carefully check to ensure that his actions do not accidentally disclose information that was previously protected by the AIM access controls.

The set_system_priv command is needed only if the site is using the AIM access controls.

ARGUMENTS

privilege_namei
    may be any of a character string which is listed below.

LIST OF CHARACTER STRINGS

comm
    turn on communications privilege.

dir
    turn on directory privilege.

seg
    turn on segment privilege.

ipc
    turn on interprocess communication (IPC) send/receive privilege.

ring1
    turn on privilege for ring 1 subsystems.

soos
    turn on security-out-of-service privilege.

rcp
    turn on privilege for RCP resource management.

If any of the above privilege_names is preceded by the character "^", the corresponding privilege is turned off. All privileges not mentioned in the argument string remain unchanged.

ACCESS REQUIRED

The user must have access to the system_privilege_ gate to use this command.

EXAMPLES

```
set_system_priv  dir ^seg ipc
```

turns on directory and IPC privileges and turns off segment privileges. The soos and ring1 privileges remain unchanged.

Name: set_system_search_rules

SYNTAX AS A COMMAND

```
set_system_search_rules path
```

FUNCTION

is a highly privileged command used in the initializer process to set the site's default search rules for all processes.

ARGUMENTS

path
: is the pathname of a default search rules segment (described below) created earlier by the system administrator.

DEFAULT SEARCH RULES SEGMENT

Each line in the default search rules segment may be either a keyword or the absolute pathname of a directory to be searched. The order of the lines in the default search rules segment gives the order in which the rules are applied by a user process.

The valid keywords are:

```
    initiated_segments
    referencing_dir
    working_dir
    home_dir
    process_dir
```

The absolute pathname search rules and the keywords may be followed by one or more tags. Tags are single word identifiers chosen by the system administrator and used to categorize search rules into groups. They are spearated from pathname or keyword search rules by a comma.

The user can find out the site-defined tags by issuing the get_system_search_rules command (described in the *Multics Commands and Active Functions* manual, Order No. AG92). For example, if the get_system_search_rules command returns the following:

```
    initiated_segments,default,fast
    referencing_dir,default,fast
    working_dir,default,fast
    >system_library_standard,default,fast,system_libraries
    >system_library_unbundled,default,fast,system_libraries
    >system_library_tools,default,system_libraries
    >system_library_auth_maint,default,system_libraries
```

the user knows there are three tags he can specify in his search rules: default, fast, and system_libraries. For convenience, the user can use these tags in his own search rules rather than specify the entire list of directories and keywords containing these tags. For example, the system_libraries tag in the user's search rules expands to:

```
>system_library_standard
>system_library_unbundled
>system_library_tools
>system_library_auth_maint
```

The order of the expanded list is the same as the order of the directories in the default search rule segment. Recursion is not allowed.

Up to 10 tags and up to 50 search rules may be specified in the default search rules segment.

A user process may specify up to 22 search rules.

EXAMPLES

Assume the system administrator has created a default search rules segment named site_rules that contains the following lines

```
initiated_segments,default
referencing_dir,default
working_dir,default
>system_library_standard,default,system_libraries
>system_library_unbundled,default,system_libraries
>system_library_tools,default,system_libraries
>system_library_auth_maint,default,system_libraries
```

To set the site's default search rules for all processes, the system administrator types:

```
set_system_search_rules site_rules
```

Name: set_timax, stm

SYNTAX AS A COMMAND

```
stm N
```

FUNCTION

sets the maximum value that ti (time since interaction) can assume for an interactive process.

ARGUMENTS

N
: is the number of seconds, in decimal, to which timax is to be set. A value less than or equal to zero causes it use the default timax from tc_data$timax.

ACCESS REQUIRED

The user must have access to both the privileged and the highly privileged gates phcs_ and hphcs_.

EXAMPLES

The command line:

```
   stm 3.5
```

sets timax to 3500000 microseconds for the current user's process and prints appropriate messages on both the user's terminal and operator console.

The command line:

```
   stm 0
```

sets timax to the default timax and prints messages on the user's terminal and operator console.

Name: set_tpp, stpp

SYNTAX AS A COMMAND

```
stpp directory {-control_args}
```

FUNCTION

prints and modifies the time-page product (tpp) of a specified directory. It should be used if the charge_disk command reports an inconsistency between the total tpp for a project, as recorded in projfile, and the total tpp for that project, as computed from the tpps of the project's directories. (This should be done only after it has been determined that the figure in projfile is the correct one, and the tpp of some directory was damaged.)

ARGUMENTS

directory
: is the pathname of the directory whose tpp is to be printed or modified.

CONTROL ARGUMENTS

–directory, –dr
: print or modify the tpp for directory pages.

–long, –lg
: print the current tpp and its equivalent dollar charge, plus the quota, pages used, time tpp last updated, sons logical volume identifier, and quota switch.

–print, –pr
: print the current tpp and its equivalent dollar charge.

–segment, –sm
: print or modify the tpp for segment pages. This is the default.

–set N
: modify the tpp as specified by N. See "Notes" below for the possible formats of the N value, and more details on the functioning of the set operation.

NOTES

If none of the arguments: –print, –long, or –set is given, –print is assumed; more than one may be given.

The N value given with the –set control argument may be in units of page–seconds or dollars and cents, a leading dollar sign ($) indicating the latter. Value may be an absolute value or a relative value (i.e., a change to be applied to the current tpp), a leading plus (+) or minus (–) sign indicating the latter. When both a dollar sign and a plus or minus sign are given, they may appear in either order. No blanks may appear between the signs and the numeric value.

When an absolute value is given for N, the setting of that value is only approximate. The equivalent change from the current tpp is computed, and then applied as a change; since the tpp of an active directory is updated frequently by the storage system, the resulting tpp, after this change is applied, is larger than the specified value by the number of page–seconds that have accrued during the execution of the command. Use of the relative form of the value argument is recommended over the absolute form.

When the –set control argument is specified, the size and direction of the change and the (approximate) resulting new value are computed and displayed, in both page–seconds and dollars, and the user is asked if the change should actually be made. If the answer is yes, the change is made and then the actual resulting new value is printed.

EXAMPLES

```
!  set_tpp >udd>sa -print
   tpp=5138695885 ($1189.51)
!  set_tpp >udd>sa -set $+2.01

   set_tpp:  This operation will INCREASE the segment page charge on >udd>sa
   by $2.01 to 5147392077 ($1191.54).  Do you want to do that?
!  yes
   tpp is now 5147408682 ($1191.55)

!  set_tpp >udd>sa -set $-2.01

   set_tpp:  This operation will INCREASE the segment page charge on >udd>sa
   by $2.01 to 5140898662 ($1190.02).  Do you want that?
!  yes
   tpp is now 5140913009 ($1190.03)
```

Name: set_work_class, swc

SYNTAX AS A COMMAND

```
swc wc_number {id}
```

FUNCTION

moves a process or set of processes from one work class to another without installing a revised master group table (MGT). The effect of this command is temporary since the answering service recomputes and resets a process' work class if the shift changes, a new MGT is installed, the user issues a new_proc command, or the operator issues the "maxu auto" command.

ARGUMENTS

wc_number
: is the number of the work class to which processes are to be moved.

id
: may be a User_id or a process identifier. If a User_id is given it must be of the form Person.Project.tag, and any or all components may be "*". If a process identifier is given it must be an octal number. If this argument is not given, only the process executing the command is moved to the specified work class.

ACCESS REQUIRED

This command requires access to hphcs_. Additionally, this command requires read access to one or more system tables if the person_id.project_id.tag option is used. If the tag specified is "a" or "*", access to the answer_table is required. If the tag specified is "m" or "*", access to the absentee_user_table is required. If the tag specified is "z" or "*", access to the daemon_user_table is required. All of these tables are located in the directory >system_control_1.

NOTES

In order to prevent severe errors, set_work_class never matches any User_id to the Initializer process. If for some reason it is necessary to move the Initializer out of work class zero, this must be done by specifying the Initializer's process identifier.

EXAMPLES

The following are examples of valid command lines:

```
swc 1 *.*.*
```

moves all processes to work class 1.

```
swc 2 *.SysDaemon.z
```

moves all processes with the SysDaemon Project_id and the tag of "z" to work class 2.

```
swc 3 *.*.m
```

moves all absentee process to work class 3.

```
swc 4
```

moves the executing process to work class 4.

Name: set_x25_packet_threshold

SYNTAX AS A COMMAND

```
set_x25_packet_threshold channel_name packet_size
```

FUNCTION

sets the minimum size of X.25 "long" packets. Packets of this size or larger are given lower priority than short packets.

ARGUMENTS

channel_name
  is the name of an X.25 multiplexer channel.

packet_size
  is the minimum length of a long packet, in characters.

ACCESS REQUIRED

Use of this command requires access to the hphcs_ gate.

NOTES

If packet_size is set larger than the maximum packet size in use by the multiplexer, no packets are given preferential treatment on the basis of size. The initial value of the minimum long packet size is determined by the "packet_threshold" parameter in the terminal type definition for the multiplexer. The present value of the parameter, and the maximum size for all packets, is included in the output from the tty_dump command.

Name: sort_projfile

SYNTAX AS A COMMAND

```
sort_projfile {path}
```

FUNCTION

sorts the entries of the projfile segment into the order required by the billing programs. The sort is by Project_id.

ARGUMENTS

path
  is the pathname of the projfile segment. The default is the segment projfile in the working directory.

Name: sort_reqfile

SYNTAX AS A COMMAND

```
sort_reqfile {path}
```

FUNCTION

sorts the entries of the reqfile segment into the order required by the billing programs. The major key is the account number and the minor key is the requisition number.

ARGUMENTS

path
: is the pathname of the reqfile segment. The default is the segment reqfile in the working directory.

Name: summarize_sys_log, ssl

SYNTAX AS A COMMAND

```
ssl {log_selector} -control control_path {-control_args}
```

FUNCTION

summarizes activity in a system log. The specified range of messages in the log are matched against the selection and printing criteria in the control file and written or summarized to the specified I/O switches. This command requires that a set of I/O switches (used to write log messages and summaries) be attached before the command is executed and detached afterwards. Control arguments determine what range of messages are processed, and how those messages are formatted in the output.

ARGUMENTS

log_selector
: is the pathname of a log to be summarized The pathname must specify the first segment in the log. This argument is incompatible with any of the log selection control arguments.

CONTROL ARGUMENTS FOR LOG SECTION

-pathname LOG_PATH, -pn LOG_PATH
: specifies the pathname of the log to be summarized. the pathname must specify the first segment in the log. This argument is incompatible with any of the other log selection control arguments, or an explicit log pathname.

-admin
: specifies that the admin log is to be summarized. The admin commands log is called "admin_log", and is located in the >sc1>as_logs directory. This argument is incompatible with any of the other log selection control arguments, or an explicit log pathname.

-answering_service, -as
: specifies that the answering service log is to be summarized. The Answering Service log family is called "log", and is located in the >sc1>as_logs directory. This argument is incompatible with any of the other log selection control arguments, or an explicit log pathname.

-dm_system, -dms
: specifies that the data management system log for the process's current AIM authorization is to be summarized. The data management log is called "dm_system_log", and its location depends on the AIM access class of the log. This argument is incompatible with any of the other log selection control arguments, or an explicit log pathname. Reading the log requires access to the dm_admin_ gate.

-mc_log LOG_NAME, -mcl LOG_NAME
: specifies that the message coordinator (daemon) log named LOG_NAME is to be summaried. All message coordinator logs are located in the >sc1>as_logs directory; their names depend on the daemon to which they belong. This argument is incompatible with any of the other log selection control arguments, or an explicit log pathname.

-syserr
: specifies that the syserr log is to be summarized. The syserr log is named "syserr_log". The first segment in the family is >sl1>syserr_log; there may be a history segment in >sl1, and older history segments are in the directory >sc1>syserr_log. This argument is incompatible with any of the other log selection control arguments, or an explicit log pathname.

CONTROL ARGUMENTS FOR CONTROL

-control CONTROL_PATH
: specifies the pathname of the selection control file to be used in summarizing the log messages. See "Control File Syntax" and "List of Selection Operators", below for details of the syntax. This argument must be specified.

-from TIME, -fm TIME, -from NUMBER, -fm NUMBER
: specifies that the first message processed is the first message at or after the specified time or sequence number; if -reverse is specified, the first message is the one at or before the specified value. If no -from value is specified, the default is the first message in the log, or the last if -reverse is specified. This is incompatible with -last.

-to TIME, -to NUMBER
: specifies the last message to be processed, either by message time or sequence number. If not specified, the default is all the remaining messages in the log. This is incompatible with -for.

-last NUMBER, -lt NUMBER, -last TIME, -lt TIME
: specifies that only the last NUMBER messages, or the messages since TIME, are to be processed. If a NUMBER is specified, it specifies the actual number of messages to be processed, not the number of messages examined in the log. This is incompatible with -from and -for.

-for TIME, -for NUMBER
: specifies a number of messages to process, or a time interval relative to the starting time (specified by -from) in which the messages must be contained. The number of messages is the actual number of messages processed, not the number of messages examined in the log. This is incompatible with -to and -last.

CONTROL ARGUMENTS FOR MESSAGE EXPANSION

-octal, -oc
: specifies that the binary data in expanded messages is to be displayed in octal, rather than, or in addition to, the interpreted representation. If both octal and interpreted representations are desired, both the -octal and -interpret control arguments must be supplied.

-interpret, -int
: specifies that the binary data in expanded messages is to be displayed as interpreted text, by calling the appropriate expand_XXX_msg_ program for the data class of the message. If the -octal control argument is also specified, the binary data is displayed both in interpreted form and as octal data. This is the default.

CONTROL ARGUMENTS FOR MESSAGE FORMAT

-line_length N, -ll N
: specifies the line length used when formatting message text and data for printing. The value (N) must be between 25 and 500. By default, it is the line length associated with the user_output I/O switch, or, if none (as for an absentee), it is 132 (for line printer output).

-indent N, -ind N
: specifies that all messages are to be formatted for printing prefixed with N spaces. The value of N must be between zero and fifty. The indentation is printed before any data associated with the message, including the message prefix. By default, there is no indentation.

-continuation_indent N, -ci N
specifies that all messages are to be formatted for printing with continuation lines prefixed by N spaces, or, if the keyword "standard" or "std" is used in place of a number, with the continuation lines indented sufficiently to line up under the first character of the text of the message. The value of N must be between zero and fifty. By default, continuation lines are indented to the "standard" indentation.

-prefix STRING, -pfx STRING
specifies that all messages are to be formatted with the specified string as a prefix. This prefix appears after the indentation (if any was specified). The prefix must explicitly include trailing spaces, if any are desired to separate the prefix from the message text. By default, there is no prefix.

-duplicates, -dup
inhibits the printing of "=" messages for messages whose text is the same as the previous message printed. All messages are printed exactly as they appear in the log.

-no_duplicates, -ndup
prints "==" for messages whose text is the same as the previous message printed. This is the default.

-date_format FORMAT_STRING, -dfmt FORMAT_STRING
specifies a date/time format string (see time_format.gi.info or the *Multics Programmer's Reference Manual*, Order No. AG91) to be used when formatting the date when successive messages are printed with different dates. The date string is printed on a line entirely by itself, preceded by a blank line. If the date format string is blank, no date separators will be printed; this should be used if a -time_format string is specified that includes the date as well. The default date string is "^9999yc-^my-^dm ^da ^za", which prints as (for example) "1984-10-31 Wed est".

By specifying null strings for date, time, and number formats, the log can be printed and saved, so that it can be compared to another log script later, without spurious mis-compares because the times and sequence numbers do not match.

-time_format FORMAT_STRING, -tfmt FORMAT_STRING
specifies a date/time format string (see time_format.gi.info or the *Multics Programmer's Reference Manual*, Order No. AG91) to be used when formatting the message time portion of the message. If the string is null, no time is printed with the messages. The default time format is "iso_time", which prints as (for example) "23:21:59".

-number_format IOA_STRING, -nfmt IOA_STRING
specifies an ioa_ string to be used when printing the sequence number for the message. If the string is null, no sequence number is printed with the message. The default is "^7d". (See the *Multics Subroutines and I/O Modules* manual, Order No. AG93 for a description of ioa_control strings.)

MISCELLANEOUS CONTROL ARGUMENTS

-procedure NAME, -proc NAME
: specifies that entrypoints in the procedure called NAME are to be used instead of entrypoints in log_read_ to read the log. This is used to read logs protected by inner-ring subsystems, where the inner-ring subsystem provides a replacement log-reading procedure. See "Access Required," below.

NOTES

The summarize_sys_log command produces multiple copies of printable files, each containing different abstracts of the log being scanned. There are two basic abstracting techniques: writing lines selected by character string matching into the file, and writing the total number of occurrences of specified types of lines.

There are two possible methods for specifying the starting point of an invocation of the sys_log_scan_report command. The recommended method is to use a value segment to record the times, as follows:

```
vs first_entry [clock calendar_clock [vg last_entry] +1usec]
vs last_entry [clock calendar_clock]
ssl LOG_PATH -from [vg first_entry] -to [last_entry]
```

NOTES ON OUTPUT

Rather than writing directly into files, the command writes its output through user-specified I/O switches. No I/O switch attachment or detachment is done by the program. It is the responsibility of the caller to ensure that all switches named in the control segment are attached and opened with the io_call command. Usually, these are attached to storage system segments through the vfile_ I/O module.

NOTES ON CONTROL FILE SYNTAX

The control file names the output switches and selection operations used by the command. The general format of a control line in the file is:

```
SWITCHNAME,SEVERITY,OPCODE,LITERAL.
```

switchname
: is the name of an I/O switch to which this information will be written.

severity
: indicates the minimum severity message this control line applies to, or a range of severities. The severity value may either be a decimal integer, or ranges consisting of a pair decimal integers separated by a colon ("20:29").

opcode
: describes the kind of selection desired. It may have any one of the values listed below:

all
: selects all lines.

any
: selects any lines containing the string <literal>.

allx
: like all, but messages with binary data have the data expanded when printed.

anyx
: like any, but messages with binary data have the data expanded when printed.

bcount
: counts all lines that begin with the string <literal> and places the total on the named switch after all entries are written.

begin
: selects all lines with the string <literal> as their beginning.

beginx
: like begin, but messages with binary data have the data expanded when printed.

count
: counts all lines that contain the string <literal> and places the total on the named switch after all entries are written.

nbegin
: diverts any lines that begin with the string <literal> from the output switch if later selected by an all, any, or begin opcode.

not
: diverts any lines that contain the string <literal> from the output switch if later selected by an all or begin opcode.

To be effective, the not and nbegin lines for a particular switch may precede the all, any, and begin control lines. The not and nbegin lines do not affect the counting of lines. If no lines were written on a switch, and a count is zero, the total line is omitted.

If any lines were written on a switch, a count of total lines written follows the totals for count and bcount. Thus, nothing is written on a switch only if all selections fail.

literal
: is an operand used to select the message; its function is described individually for each opcode. All characters following the comma are taken literally; no quote processing is performed.

ACCESS REQUIRED

For all except inner-ring logs, read permission is required on the log segments themselves, and status permission is required on their containing directories. If an access error is encountered searching for older history logs, the search is stopped at that point, and no further history will be available. For the logs selected by control arguments, the control argument descriptions list the standard history directories for the logs.

For inner-ring logs (the data management system log is the only standard inner-ring log), access to the logs is required, as is access to the gate used by the log-reading procedure (see -procedure).

NOTES ON SEVERITY VALUES

Severity values in log messages are used to indicate the importance of the message being logged, in a general way. Most logs use increasing severity to indicate increasing importance, but the actual meaning depends on the log. For the Answering Service and Message Coordinator logs, the severities have the following meanings:

```
0 => Message just logged
1 => Message logged and printed on a console
2 => Message logged and printed on a console with bells
3 => Message logged, printed, and the system crashed
```

For the syserr log, the severities have different meanings:

```
0 => Message logged and printed on syserr console
1 => Message logged, printed, and the system crashed
2 => Message logged, printed, and the process writing the
     message is terminated.
3 => Message logged and printed, and console alarm sounded
4 => Message just logged, or printed if logging mechanism is
     inoperable
5 => Message just logged, or discarded if it can't be logged
```

The severities 20 to 25 are handled just like 0 to 5, but are different to indicate that the originating program was writing an access audit message, rather than just an informative message.

NOTES ON INNER-RING LOGS

Some applications create logs in an inner ring that must be read using a special interface. The only standard log to do this is the Data Management system log, and it is read by specifying the -dm_system control argument which supplies both the pathname and the procedure name (dm_log_read_). Other applications may provide their own special procedures for log reading, in which case both the log pathname and the procedure name must be supplied explicitly via the -pathname and -procedure control arguments. Note that a log read using a reader procedure may enforce additional access requirements as well as requiring access to the log itself. In particular, the user must have access to the reader procedure.

Name: sweep

SYNTAX AS A COMMAND

```
sweep {path} {-control_args}
```

FUNCTION

obtains the disk usage figures for the hierarchy, or any specified subtree, and places them in a segment for subsequent use by accounting and disk usage analysis tools. By default, the segment is named disk_stat. It is not an ASCII segment, and the disk_stat_print command must be used to print its contents.

ARGUMENTS

path
: is the pathname of the directory at which the sweep is to start. The usage figures for this directory, and for the entire subtree beneath it are recorded. If path is omitted, the sweep begins at the root directory. See "Notes" below.

CONTROL ARGUMENTS

-brief, -bf
: does not print an error message when unable to force access to a directory to read its usage figures. This is the default.

-long, -lg
: prints an error message if unable to force access to a directory.

-output_file XX, -of XX
: places the output in the specified segment XX rather than in the default segment, named disk_stat, in the working directory.

-pdd
: do not exclude >pdd from the sweep. By default, a sweep that starts at the root deliberately omits >pdd. This argument is not used in the accounting application of this program, but it may be used during an analysis of disk space usage, when complete and accurate total page usage figures are desired. The initializer process must first be used to give sma access to >pdd to the process that will run the sweep.

NOTES

The figures recorded in disk_stat are the quota, records used, and time-record product (trp). These figures are recorded separately for segment pages and directory pages.

Since directories with zero quotas have their record usage and trp figures included in those of the first superior directory with nonzero quota, it is not necessary for sweep to walk the entire hierarchy. Instead, whenever it encounters a directory with a zero quota, it goes no further down in that particular branch, but instead goes on to the next directory at the same level. This applies separately to segment and directory quotas.

Therefore, any directory whose usage figures are to be recorded separately for purposes of accounting or disk usage analysis must be given nonzero segment and directory quotas. It is recommended that at least all directories immediately inferior to the root and all project directories under >udd be given nonzero segment and directory quotas.

Project administrators may use this command to obtain detailed information about the disk usage of a project's users by executing sweep with the project directory pathname as the first argument. This is possible, however, only if the users' home directories have nonzero quotas.

EXAMPLES

```
   sweep >udd>Alpha -output_file Alpha.disk_stat -long
```

places disk usage information for the Alpha project directory and all directories beneath it with quotas in the segment Alpha.disk_stat, and prints an error message if unable to force access to any directory. Alpha.disk_stat can then be printed using the disk_stat_print command.

Name: sweep_pv

SYNTAX AS A COMMAND

```
sweep_pv pvname {-control_args}
```

FUNCTION

performs utility functions that require walking through the VTOC of a mounted physical volume. Among these functions are the listing of contents and unconnected VTOCEs, the deletion of such VTOCEs, and the evacuation of physical volumes (for logical volume compression). The volume is processed in VTOC order.

ARGUMENTS

pvname
is the name of the mounted physical volume whose VTOC is to be walked through.

CONTROL ARGUMENTS

-adopt
attempts to reconstruct directory branches for VTOCEs encountering reverse connection failures. Unique names are derived from the primary name in the VTOCE. The -gc control argument must be given to use the -adopt control argument. If the -dl control argument is given as well, VTOCEs that cannot be adopted are deleted. The success or failure of adoptions is reported to the report produced by the -gc control argument. For a description of the definition attributes provided for adopted branches, see the description of the adopt_seg command. This request requires hc_backup_ access.

-debug, -db
is for system programmer use, and disables highly privileged calls in order to allow debugging of this program.

-delete, -dl
deletes unconnected VTOCEs. This control argument is only allowed when the -gc control argument is specified. Access to hphcs_ is required to use this control argument. The collection report is modified to indicate that these VTOCEs were deleted.

-force, -fc
forces access to all directories that need to be scanned where status permission is lacking. Status permission is necessary for all components of the pathnames of VTOCEs processed. If this control argument is not used, or status permission is lacking and cannot be forced, an error is reported.

-from first
: indicates the first VTOCE to be processed. If first is not given, VTOCEs are processed starting at VTOC index 0.

-gc
: performs garbage collection, i.e., all VTOCEs for which a reverse connection failure is encountered in resolving their pathnames are logged in a collection file. This file is produced whether or not the -list control argument, above, is specified. At the end of the collection file report, the total number of VTOCEs and records held by them in this state is printed. Selected information, such as the name in the VTOCE and date/time used and modified, is also put in the report for each such VTOCE. The collection file report is given a three-component name and is placed in the working directory. The three-component name has "pvgc" as the first component, the physical volume name as the second, and the time the segment is created as the third component (e.g., pvgc.public.0814).

-list
: creates a listing file containing the VTOC index, time listed, and pathname of every segment on the volume at the time its VTOCE is scanned. An asterisk on the line of the listing for a particular VTOCE indicates a reverse connection failure (i.e., no branch exists for the entry). For nondirectory segments, the page fault meter of the segment is given. The convention for listing incomplete pathnames is the same as that used by the vtoc_pathname command (see the vtoc_pathname description in the *Multics Commands and Active Functions* manual, Order No. AG92). The listing file is given a three-component name and is placed in the working directory. The three-component name has "pvlist" as the first component, the physical volume name as the second, and the time the segment is created as the third component (e.g., pvlist.public.0749).

-move, -mv
: moves all segments corresponding to VTOCEs processed off this volume onto another physical volume of the same logical volume. Segments that suffer reverse connection failures are not moved. If segments are moved for some purpose other than evacuation, the inhibit_pv command should be invoked with the -off control argument after moving has taken place. The sweep_pv command attempts to target segment moves to the physical volume in the logical volume with the most page space available by inhibiting and disinhibiting volumes dynamically; all volumes except the volume being evacuated are disinhibited when the sweep_pv command is exited. Volumes found to be inhibited at the time the command is invoked are not disinhibited; this facilitates simultaneous evacuation of volumes. Any volume that is completely evacuated by this means should be deregistered before the logical volume is reaccepted. Access to the hphcs_ gate is required to use this control argument. (See the *Multics System Maintenance Procedures* manual, Order No. AM81, for more details.)

-only vtocx
: processes only the single VTOCE whose octal index is vtocx.

-to last
    indicates last VTOCE to be processed. If last is not given, processing proceeds to the end of the VTOC.

ACCESS REQUIRED

The sweep_pv command requires phcs_ access, access to hc_backup_ if the -adopt control argument is used, and access to hphcs_ if the -delete or -move control arguments are used.

NOTES

Any errors encountered during the VTOC sweep are reported to a file that is given a three-component name and placed in the working directory. The three-component name has "pvef" as the first component, the physical volume name as the second, and the time the segment is created as the third component (e.g., pvef.public.1321). Errors in command usage, such as specifying a nonexistent or unmounted physical volume, are reported to the bootload console.

When sweep_pv is invoked with the -move control argument, it first inhibits segment creation on the volume being vacated, and then vacates the pack. When sweep_pv terminates, it leaves the volume inhibited and prints a message to this effect. The volume can be uninhibited manually by means of the inhibit_pv command.

Name: sys_full_report

SYNTAX AS A COMMAND

```
sys_full_report {log1 log2 ...  logN} {-control_arg}
```

FUNCTION

is used by the crank in master.ec to update, and by biller.ec to print, the contents of a segment containing a history of all instances where a user was refused login because the system was full.

ARGUMENTS

logi
    is the pathname of a system log segment.

CONTROL ARGUMENTS

-print
    print the contents of the segment.

NOTES

The history is kept in the segment sys_full_report_seg in the working directory (normally >udd>SysAdmin>admin). This segment is created if it does not exist.

If given, the -print control argument must be the last, or the only, argument. The crank (in master.ec) invokes this command with the name of just one log segment at a time; the biller.ec segment invokes it with just the -print control argument.

Name: system_comm_meters

SYNTAX AS A COMMAND

```
system_comm_meters {-control_args}
```

FUNCTION

prints out metering information for ring zero Multics Communications Management.

CONTROL ARGUMENTS

-report_reset, -rr
: prints metering information and then resets the metering interval.

-reset, -rs
: resets the metering interval for the invoking process so that the interval begins at the last call with -reset specified. The metering information is not printed. If -reset has never been given in a process the interval begins at system initialization time.

NOTES

Use of the system_comm_meters command requires access to the metering_gate_ and phcs_ gates.

EXAMPLES

The following is a sample of the output of the system_comm_meters command.

```
Total metering time 05:43:27

THROUGHPUT
                              before conversion  after conversion  ratio
Total characters input          17,234,567         15,543,210       0.90
Total characters output        168,012,345        185,876,543       1.14
Average length of input             12.3 characters
Average length of output            59.7 characters
Input characters preconverted       20,435 (1.2% of total)

                              read                 write
Number of calls               1,456,789            26,357,924
Average time per call          6.37 msec.           9.63 msec.
Average chars. processed      13.5                 57.8
Average chars. per msec.       2.1                  5.8
CHANNEL INTERRUPTS
                          input         output         other        total
software "interrupts"    678,901       423,440        110,011    1,212,352
average time (msec.)       1.34           0.56           0.23         1.01


TTY_BUF SPACE MANAGEMENT

Total size of buffer pool          11,480 words
Number of channels configured         143
Number of multiplexed channels          8

% of buffer pool in use              current   average
    input                              6.9       6.5
    output                            13.4      15.6
    control structures                15.8      15.3
    ---------------------------------------------------
    total                             36.1      37.4

Smallest amount of free space ever    4,358 words (38% of buffer pool)

                                 allocate       free         total
Number of calls                 24,657,988   20,665,443   45,323,431
Average time per call (msec.)        0.23         0.37          0.29
% of total CPU                       0.14         0.17          0.31
Calls requiring loop on tty_buf lock        1,249,340  (2.83% of total)
Average time spent looping on lock         0.14 msec.  (0.01% of total CPU)
Number of allocation failures                      0   (0.00% of attempts)
```

```
CHANNEL LOCK CONTENTION

Number of calls to tty_lock                 40,392,817
Times channel lock found locked              2,364,758 (5% of attempts)
Average time spent waiting for lock          1.8 msec.
Maximum time spent waiting for lock          3.7 msec.
Number of interrupts queued because
 channel locked                              25,437 (2.2% of interrupts)


ECHO NEGOTIATION

Average time of transaction                  3.2 msec.
Number of characters echoed by supervisor  21,576 (0.13% of input chars)
Number of characters echoed by FNPs       335,466 (1.87% of input chars)


ABNORMAL EVENTS

Input restarts                             12,576 (0.8% of read calls)
Output restarts                           304,289 (1.2% of write calls)
Output space overflows                     16,384 (0.1% of write calls)
"needs_space" calls                             0
```

Name: system_daily_report

SYNTAX AS A COMMAND

```
system_daily_report today.use_totals yesterday.use_totals
```

FUNCTION

prints a report of the system usage for the day just ending.

ARGUMENTS

today.use_totals
is a data base giving month-to-date usage information including the usage of the current day.

yesterday.use_totals
is the previous day's month-to-date usage segment.

NOTES

The day's usage totals are computed by subtracting the month-to-date totals in yesterday.use_totals from those in today.use_totals. This command is executed by the crank (in master.ec) and assumes that the two use_totals segments contain the information for the respective days, as suggested by their names. The information in these segments is placed there by the system_total and usage_total commands, also executed by the crank.

Name: system_link_meters

SYNTAX AS A COMMAND

```
system_link_meters {-control_arg}
```

FUNCTION

prints out system-wide statistics regarding usage of the Multics linker. Information is obtained from the active_hardcore_data and tc_data data bases.

CONTROL ARGUMENTS

-report_reset, -rr
generates a full report and then performs the reset operation.

-reset, -rs
resets the metering interval for the invoking process so that the interval begins at the last call with -reset specified. If -reset has never been given in a process, it is equivalent to having been specified at system initialization time.

ACCESS REQUIRED

This command requires access to phcs_ or metering_gate_.

NOTES

If the system_link_meters command is given with no control argument, it prints a full report.

Statistics are given for overall use of the linker, and are also broken down by task. The three major tasks of the linker are:

1. Searching the definition section of the object segment for the symbolic name of the referenced segment.

2. Searching for the segment using the standard search rules.

3. Getting the linkage to the referenced segment.

The following are brief descriptions of the variables printed out by the system_link_meters command.

CPU Metering time
: is the amount of time for which the processor was busy. It equals total processor time minus idle time.

Total time in linker
: is the total amount of CPU time spent in the linker, expressed as hh:mm:ss.

Average time per link
: is the average time to completion (in milliseconds) for a call to the linker.

Percentage of real time in linker
: is the percentage of total metering time that was spent in the linker.

Percentage of CPU
: is the percentage of virtual CPU metering time that was spent in the linker.

Time Slot
: are the time slots into which calls to the linker are broken down. The four slots are for calls completed in less than 25 milliseconds, calls completed in between 25 and 50 ms, calls completed in between 50 and 75 ms, and calls completed in more than 75 ms.

Calls
: is the number of calls that were completed in each time slot.

Total time in slot
: is the total amount of virtual CPU time taken by calls in each time slot.

Percent total time
: is the percentage of the virtual CPU time spent in the linker that was taken by calls in each slot.

Percent total calls
: is the percentage of calls to the linker that fell into each time slot.

Average time
: is the average time (in milliseconds) to complete a call to the linker that ended up in each time slot.

Average page faults
: is the average number of page faults for a call in each slot.

The following statistics are given for each of the three major tasks of the Multics linker: definition search, segment search, and get linkage.

Average time
: is the average time (in milliseconds), for a call in each slot, spent on that particular function of the linker.

Average page faults
: is the average number of page faults for a call in each slot, which occurred during that particular task of the linker.

Percent time in slot
: is the percentage of the total time spent in the slot that was taken up by that particular task. These percentages do not add up to 100% because some time used by the linker does not fit into any of the three categories.

EXAMPLES

The following is an example of the information printed when the system_link_meters command is invoked with no control argument. (See Appendix A for a representation of the configuration deck used to create the system from which the metering sample was taken.)

```
Linkage Meters:
CPU Metering time                      0:15:33

Total time in linker                   0:02:17
Average time per link                 5.83 msec.
Percentage of real time in linker     14.78
Percentage of CPU time in linker       3.16
```

```
Time slot (msec)           <25         25-50        50-75          >75

Calls                    22995           569           56           51
Total time in slot     0:01:44       0:00:17      0:00:03      0:00:11
Percent total time       76.11         12.88         2.36         8.65
Percent total calls      97.14          2.40         0.24         0.22
Average time              4.57         31.21        58.24       233.93
Average page faults       0.21          2.59         5.55         6.71

Segment Search
 Average time             2.37         25.95        48.39         6.82
 Average page faults      0.04          1.12         2.10         0.04
 Percent time in slot    56.12         82.81        83.74         2.88

Get Linkage
 Average time             0.81          4.18         8.04       229.12
 Average page faults      0.07          0.79         1.95         6.50
 Percent time in slot    19.29         13.35        13.91        96.69

Definition Search
 Average time             0.22          0.24         0.21         0.19
 Average page faults      0.02          0.11         0.15         0.00
 Percent time in slot     5.26          0.76         0.36         0.08
```

Name: system_monthly_report

SYNTAX AS A COMMAND

```
system_monthly_report today.use_totals last_month.use_totals
```

FUNCTION

prints a report that shows system resource usage for the current billing period and percent changes from respective figures for the previous billing period.

ARGUMENTS

today.use_totals
    is the pathname of the month-to-date statistical data base.

last_month.use_totals
    is the pathname of the copy of the statistical segment that was saved at the end of the previous billing period.

NOTES

This command is executed by biller.ec as part of the monthly billing.

The use_totals segments are created and initialized by the reset_use_totals command and updated by the system_total and usage_total commands.

Name: system__performance__graph, spg

SYNTAX AS A COMMAND

```
spg sample_time {-control_args}
```

FUNCTION

generates a system of graphs that meter information concerning system performance and operation. The output can be directed to a file or to the controlling terminal. Periodically, metering information is presented in an output line. The initial line contains the cumulative values since system initialization. Whenever there is a change in system configuration or any of several parameters affecting system performance, an additional line noting the change is issued before the sample line. In this way, a system of graphs is developed where various metered quantities are plotted against time. Because the sampling is implemented by means of an event call channel, it is possible to use the terminal in a restricted way for other purposes while metering is in progress. All output is produced on the I/O switch spg_output_.

ARGUMENTS

sample_time
: is a decimal integer giving the time, in minutes, desired between meter display lines.

CONTROL ARGUMENTS

-halt, -ht
: terminates plotting.

-output_file {path}, -of {path}
: directs output to a file called spg_output, or if a path is supplied, directs output the the file specified by path.

-short
: compresses the width of the meter display lines (see "Notes" below).

ACCESS REQUIRED

This command requires access to phcs_ or metering_gate_.

NOTES

Description of the output pattern:

1. An initial line gives the date and time that metering sampling began.

2. A line is given describing configuration and scheduling parameter settings.

3. The current state of the meters since system initialization is on the next line where the sample time is replaced by the system initialization line.

4. Each subsequent meter display line gives the incremental status of the meters since the previous line. In addition, whenever the configuration or scheduling parameter settings change, a notification line is interspersed.

Description of the meter display line:

Each line contains, in the left margin, the time that the sample was taken. Each sample is scheduled to be taken at an exact minute so that the amount that the time given exceeds the minute represents the response time. Strictly interpreted, the time discrepancy is the response time of a trivial request only if the metering computation is less than the quantum and if the command argument sample_time is greater than one minute so that interactive scheduling occurs.

The remainder of the meter display line consists of a sequence of superimpositions over a grid 100 units wide. When the -short control argument is given, the total width of the grid is only 50 units, so all individual components are correspondingly compressed. The "Example" section below shows the output when the command is invoked with the -short control argument. The wider display would, of course, be easier to read. The 100-unit grid is created by vertical bars every 10 spaces with periods at the intervening midpoints between the bars. Over this grid, various metering quantities are superimposed in the order shown below. When the superimposition is complete, the resultant line containing only the last characters superimposed is printed.

1. Time Usage Percentages

   blank
      located to the right of y to right margin user processing not in ring 0. (The position of y is an estimate; it is a figure that divides user processing into ring 0 and non-ring 0 sections.)

   blank
      located to the right of s to left of y user processing in ring 0.

   s
      time spent handling segment faults.

   p
      time spent handling page faults.

t
time spent in the traffic controller.

i
interrupt processing.

multiprogramming idle, *
nonmultiprogramming idle.

blank
located from the left margin to left of *'s zero idle.

2. Other Values

The current average is determined from samples taken at one-second intervals weighted backwards in time exponentially, with a smoothing constant of 1/64. The effect is to average over roughly the last minute.

q
relative to the left margin
current average of the ready list length.

e
relative to the left margin
current average of the number of eligible processes.

r
relative to the left margin
current average of the response time in seconds, for trivial requests.

Q
relative to the left margin
average over a sample of quits/minute.

S
relative to the left margin
average over a sample of schedulings/(10 seconds).

D
: relative to the right margin
average over a sample of disk read and write traffic in units of pages/(.1 seconds). Full scale equals 1000/sec.

P
: relative to the right margin
average over a sample of all read and write traffic (bulk store and disk) in units of pages/(.1 seconds). Full scale equals 1000/sec.

V
: relative to the right margin
average over a sample of VTOC entry read and write traffic in units of VTOCES/(.1 seconds). Full scale equals 1000 VTOCES transferred per second.

-
: relative to the left margin
number of load units at the time of the sample. If this number is greater than 100, 100 is assumed.

+
: relative to the left margin
number of users at the time of the sample. If this number is greater than 100, 100 is assumed.

EXAMPLES

The following example was generated by the command line:

```
spg -short 1
```

Note that the grid marks are not overwritten if the character that would appear is the same as both adjacent characters. (See Appendix A for a representation of the configuration deck used to create the system from which the metering sample was taken.)

```
                              12/12/83  1247.3 mst Mon
up= 12/10/83  0449.2 mst, sys_hours= 56.0, cpu_hours= 279.8
    cpu= 5, pages= 8826, min_e= 2, max_e= 20, wsa= 0, wsf= 0.50,
    tefirst=  0.50, telast=  1.00, timax=  8.0
1247.39 Q    .e  q|    *****|*********|******Sts|y   .  D +
1248.01 rmQieiqtpp|    .   y|    .    |    .    |   D.   V+
1249.05 *Qiiiitteppqs  .    | y  .    |    .    D    .    +
1250.08 *Qiiiitepps q  . y  |    .    |    .    | D  .    +
1251.04 *Qiiiiteppsq   .   y|    .    |    .    |D   .   V+
1252.02 *Qiiiietppq    .    y    .    |    .    | D  .   V+
1253.05 *Qiiiiteppqs   .    y    .    |    .    |D   .    +
1254.09 *rQiiitteppssq .    | y  .    |    .    | D  .   V+
1255.07 *rQiiiteppps q .   y|    .    |    .    D    .    +
1256.07 *Qiiittpeps q  .  y |    .    |    .    | D  .    +
1257.03 *rQiiietppqp   .    y    .    |    .    D    .    +
1258.06 Qriiiitepppq   .    y    .    |    .    |D   .   V+
1259.05 Qriiiettpqs    .  y |    .    |    .    | D  .   V+
1300.15 *Qiiittepp| q  . y  |    .    |    .    | D  .   V+
1301.16 Qriiiitepps q  . y  |    .    |    .    | D  .   V+
1302.05 *Qiiitteppq    .   y|    .    |    .    |D   .    +
```

Name: system_total

SYNTAX AS A COMMAND

```
system_total meter_data use_totals
```

FUNCTION

inspects a daily copy of the answering service metering segment, extracts system availability and performance information, and places it in a month-to-date statistical data base.

ARGUMENTS

meter_data
    is the pathname of the meter data base in the form produced by as_meter_.

use_totals
    is the pathname of the statistical data base.

NOTES

This command is executed by the crank (in master.ec). The meter_data segment is a copy, made by the copy_as_meters command, of the segment actually being used by as_meter_. The use_totals segment is today.use_totals, which is subsequently used as input to the system_daily_report program.

Name: test_cpu

SYNTAX AS A COMMAND

```
test_cpu {-control_args}
```

FUNCTION

checks the CPU hardware for problems that have existed on the processors. By running various tests invoked by this command, you can determine whether the given CPU has had specific problems fixed. This command is usually used with the set_proc_required command if the system being tested has multiple CPUs configured. If one of the test scripts fails and the successful execution of that test is dependent upon installation of a particular FCO, the FCO number is displayed in the error message.

CONTROL ARGUMENTS

-brief, -bf
: inhibits the printing of the test number and name, prior to the execution of a test.

-cycle COUNT
: repeats each test for COUNT times, then proceeds on to the next test.

-exclude LIST, -excl LIST
: excludes tests specified in the LIST, where LIST can be either a valid test number or a name.

-exclude TEST_LIST, -excl TEST_LIST
: excludes the tests identified by TEST_LIST, where TEST_LIST is either a set of test names or numbers, from the tests that are run.

-from TEST NUMBER/NAME, -fm TEST NUMBER/NAME
: starts testing from the test identified by TEST NUMBER or NAME. The default is to start testing from test 1.

-help
: displays a brief usage statement. This control argument should not be used with any other control argument.

-history_regs, -hregs
displays history registers when a test fails. The default is not to display them.

-long, -lg
displays machine conditions and history registers after a fault has occured. The default is not to display them.

-machine_conditions, -mc
displays machine conditions when a test fails. The default is not to display them.

-repeat COUNT, -rpt COUNT
repeats an entire sequence of tests for COUNT times. The default is to run the test set one time.

-select TEST_LIST, -sel TEST_LIST
executes only those tests specified by TEST_LIST, where TEST_LIST is either a set of test names or numbers from the tests that are run.

The tests are described briefly below. To find out the exact details of each test, see the test_cpu program. The default is to run all tests. The command line:

```
test_cpu -select cmpc tmir -repeat 5 -count 2
```

executes the "cmpc" test twice, then the "tmlr" test twice. The sequence is repeated five times.

-stop_on_failure, -sof
stops testing when a test failure occurs and returns to a new Multics command level. The default is to continue testing with the next test.

-test_names
lists valid test names and the associated test numbers known to test_cpu. This control argument should not be used with any other control argument.

-to TEST NUMBER/NAME
stops testing after the test identified by TEST NUMBER or NAME. The default is to run all tests.

DIAGNOSTIC TESTS

mlrstern
checks a failure in which the fill character is placed as the first character on a page. This test causes a MME1 fault if the hardware fails.

tmlr
tries several MLR instructions, in several working combinations, across a page boundary. Messages are printed for any failures.

csl_oob
: checks a particular use of a CSL instruction where the first descriptor is 0. This test causes an out_of_bounds fault if the hardware fails, and a MME1 fault if it succeeds.

mvn
: checks the use of an MVN instruction that moves a number to a shorter number. The first two characters are dropped when the hardware fails.

mvn_ofl
: checks the use of MVN to move the number 0. An overflow indicates that the hardware failed.

tct
: checks a particular TCT use. The test causes an op_not_complete if the hardware fails, and a MME1 fault if it succeeds.

sreg
: checks the use of an SREG instruction that occurs as the last instruction in a page. The test causes an op_not_complete if the hardware fails, and a MME1 fault if it succeeds.

csl_onc
: checks a particular CSL use. The test causes an op_not_complete if the hardware fails, and a MME1 fault if it succeeds.

test_sc2
: checks the use of the SC modifier interacting with page faults. A MME1 fault occurs if the hardware fails.

test_ci
: checks the use of the CI modifier interacting with page faults. A MME1 fault occurs if the hardware fails.

rpd_test
: checks a particular use of the RPD instruction as it interacts with the hardware. A MME1 fault occurs if the hardware fails.

mlr_test
: checks the use of the MLR instruction across a bounds fault boundary. The bounds fault is followed by a segment fault and a page fault. A MME1 fault occurs if the hardware fails.

cls_test
: checks the CSL instruction across a bound fault boundary. A MME1 fault occurs if the hardware fails.

cmpc
: checks the CMPC instruction in a way that fails if a timer runout or connect fault occurs in midexecution when the hardware is failing. A MME1 fault occurs if the hardware fails.

bad_fill
: checks the success of moving or comparing fill characters in the first two words of a page. Failure is indicated by a miscompare and a message to the user.

mpy_ofl
: multiplies -2**35 by itself and checks for an overflow fault (which indicates failure).

test_xed
: checks a particular indexed XED usage that fails if the first executed instruction is an APU-type instruction. Failure is indicated by a miscompare and a message to the user.

cmpc7
: checks a CMPC failure when both strings begin seven words from a page boundary and run into the next page. A MME1 fault occurs if the hardware fails.

extra_fill
: checks the MLR instruction to see if extra fill characters are placed after a string when the string crosses a page boundary. A MME1 fault occurs if the hardware fails.

test_cmpc_fill
: checks the fill mechanism of the CMPC instruction near a page boundary. A MME1 fault occurs if the hardware fails.

acv_restart
: checks that machine conditions can successfully be restarted after an access violation fault that is caused by a reference to data via an EIS (MLR) instruction. Failure is indicated by successive no_write_permission conditions.

scm_tally
: checks to see if the SCM instruction works with the tally runout indicator set correctly. The test calls a small alm program that uses an SCM instruction. Because the hardware fails erratically, the test is run 10 times to get a (limited) statistical sampling. Failure is indicated by a message to the user indicating the number of times the SCM instruction failed.

| mvt_ascii_to_bcd
checks nine to six (ASCII to BCD) conversion using the MVT instruction. A large ASCII data segment is generated. Then a BCD segment is generated using non-EIS conversion. Three segments are then converted from ASCII to BCD using the MVT instruction, and these segments are compared to the known good BCD segment. If any compare errors are detected, the contents of both segments are dumped in octal at the failing location.

| mvt_bcd_to_ascii
checks six to nine (BCD to ASCII) conversion using the method described for the mvt_nine_to_six test above. If any compare errors are detected, the contents of both segments are dumped in octal at the failing location.

mvt_nine_to_four
checks 9-bit to 4-bit (decimal to packed decimal) conversion using the MVT instruction. A large segment of data, containing 9-bit characters of values 0 to 15 in a rotating pattern, is generated. Then a second segment is generated, converting the 9-bit characters into 4-bit characters using non-EIS conversion techniques. The 9-bit data segment is then converted to three 4-bit data segments using the MVT instruction and compared to the known good 4-bit data. If any discrepancies are found, the contents of both segments are dumped in octal at the failing location.

mvt_four_to_nine
checks 4-bit to 9-bit (packed decimal to decimal) conversion using the method described for the mvt_nine_to_four test above. If any compare errors are found, the contents of both segments are dumped in octal at the failing location.

mvt_ascii_to_ebcdic
checks nine to nine (ASCII to EBCDIC) character conversion using the method described for the mvt_nine_to_four test above. If any discrepancies are found, the contents of both segments are dumped at the failing location.

mvt_ebcdic_to_ascii
checks nine to nine (EBCDIC to ASCII) character conversion using the method described for the mvt_nine_to_four test above. If any discrepancies are found, the contents of both segments are dumped in octal at the failing location.

ci_mod_case_2
checks character indirect modification with two tally words and two data character strings, each located at a page boundary. An LDA instruction is executed on one tally word, CI mod, and a CMPA is executed with a second tally word, CI mod. Both tally words point to a character string that should be equal. If the zero indicator does not come on as a result of the CMPA, a MME1 fault is taken, indicating that the hardware failed.

acv_restart_csl
validates that machine conditions can be successfully restarted after an access violation fault that is caused by a reference to data via an EIS (CSL) instruction. Failure is indicated by successive no_write_permission conditions.

cmpn_tst
: checks that numeric data moved with an MVN instruction can be successfully compared with a CMPN instruction. Failure is indicated by a MME1 fault.

itp_mod
: checks that an EPP2,* to a word pair that contains an ITP modifier with a bit offset actually loads PR2 with the correct information. A MME1 fault indicates failure.

mvnoosb
: checks the prepage logic of the CPU for EIS numeric instructions. Failure is indicated by a MME1 fault.

cmpb_with_sixbit_offset
: checks the CMPB instruction with a six bit offset. A MME1 fault indicates that the hardware failed.

cmpb_with_rotate
: checks the CMPB instruction with a rotating pattern. A MME1 fault indicates that the hardware failed.

cmpc_pgbnd
: compares a 38-character data string against a zero-length string, for a CMPC instruction that is located at seg|1767. Either an out_of_bounds condition or a MME1 fault indicates that the hardware failed.

csl_pgflt
: checks that a CSL instruction does not get a no_write_perm condition if it causes a page fault on the target string and the source string is read-only.

scm_pgflt
: tests a problem with the SCM instruction whereby the target operand takes a page fault and the resulting comparison is not made. Failure is indicated by a message to the user indicating the number of miscompares.

scd_con_flt
: tests a failure with the SCD instruction that fails when interrupted by a connect fault. Failure is indicated by displaying the number of times the SCD failed.

xed_dirflt_even
: tests the ability of the CPU to perform an XED, located on an even word location, of a pair of instructions located on a page boundary. Failure is indicated by an IPR fault.

xed_dirflt_odd
tests the ability of the CPU to perform an XED, located on an odd word location, of a pair of instructions located on a page boundary. Failure is indicated by an IPR fault.

cmpc_adj_len
tests the ability of the processor to perform a CMPC instruction which takes a fault on D2 and D2 has residue, indicated by the MIF flag. The test fails when the level count on D2 is not adjusted correctly on the SPL. Failure is indicated by an IPR fault.

cmpc_zero_ind
tests the ability of the processor to correctly restore the zero indicator after returning from a page fault on D2 after a match has occurred utilizing the CMPC instruction. Failure is indicated by an IPR fault.

scm_tro
tests the ability of the processor to find the correct character using a SCM instruction. The tally runout indicator should not be set. Failure is indicated by an IPR fault.

rpt_test_odd
checks that a RPT instruction in an odd location does not fail after a page fault on a STZ instruction, after crossing a page boundary. Failure is indicated by an IPR fault.

rpt_test_even
checks that a RPT instruction in an even location does not fail after a page fault on a STZ instruction, after crossing a page boundary. Failure is indicated by an IPR fault.

scd_oob_tst
tests the conditions when D3 resides in a different segment than D1 or D2. Also tests the conditions if there is no match or if the scan ends a few words from the end of a 64K seg. A seg fault is taken on the seg described by D3.

cmpb_onc
checks the CPU for cmpb to complete correctly. An op_not_complete fault will occur on a failure.

cmpc_a
checks the CPU to insure that the indicators are set correctly. An illegal_opcode indicates an error.

cmpc_b
same as cmpc_a except the data and addresses are changed.

sreg_no_write
checks that the TRS is used (not the PSR) if a sreq instruction is executed two locations from a page boundary.

tnz
: checks for the conditonal transfer at a page boundary. An illegal_opcode indicates an error.

NOTES

All the tests run by test_cpu are contained in the segment >system_library_tools>bound_cpu_tests_. This segment has an added name of cpu_tests_. To display the machine condition trace of a test, use the mc_trace command with the test_cpu command.

Name: test_dcw

SYNTAX AS A COMMAND

```
test_dcw {device} {name} {-control_args}
```

FUNCTION

constructs and executes arbitrary DCW lists on any device supported by the I/O interfacer.

ARGUMENTS

device
: is the name of the device to be used. This can be either a specific device name, such as "tape_02" or "puna," or a generic device type, such as "printer" or "disk." If the device name is omitted, "tape" is assumed.

name
: is the name of the tape or disk volume to be mounted. This argument is only used if the device is a tape or a disk, and is the name of the volume the operator is requested to mount. If the tape or disk volume name is omitted, "scratch" is assumed.

CONTROL ARGUMENTS

-7track, -7tr
: specifies a 7-track tape drive. This argument only applies if the device is a tape.

-debug, -db
: runs the program in debug mode. In this mode, only the editing requests are recognized; no execution is allowed, and no actual device attachment takes place.

-priv
: specifies a privileged attachment (see "Device Attachment" below.)

-read
    places the device in read-only mode. This control argument only applies if the device is a disk or a tape.

-sys
    sets the system_flag in the rcp_ info structure during attachment (see "Device Attachment" below).

DEVICE ATTACHMENT

The test_dcw command attaches the device selected using the rcp_ subroutine. Normally, the call is made to rcp_$attach as a nonsystem process. However, if -priv is used, the call is made to rcp_priv_$attach. In both cases, if -sys is used, the system_flag in the rcp_ info structure is set, to indicate to rcp_ that you are to be considered a system process. You must have re access to the rcp_sys_ gate to make this kind of attachment. If the device specified in the command line is a device type rather than a specific device, rcp_ is relied upon to select the actual device to be used. In either case, the name of the device actually attached is printed after attachment completes.

Commands

After the test_dcw command is invoked, commands are read from the user_input I/O switch. The following commands are recognized:

tdcw
    constructs a transfer DCW

idcw
    constructs an instruction DCW

nidcw
    constructs a nondata transfer IDCW

iotp
    constructs an I/O transfer and proceed DCW

iotd
    constructs an I/O transfer and disconnect DCW

iontp
    constructs an I/O transfer and proceed DCW

odcw
    constructs a DCW from octal input

pcw
    constructs a PCW

opcw
constructs a PCW from octal input

edit, e
selects a DCW list to edit

update, u
places editor in "update" mode

insert, i
places editor in "insert" mode

delete, dl, d
deletes a DCW from the list

print, p
prints a DCW list

name
names a DCW list so that it can be referenced by name instead of number

save
saves all the current DCW lists in a segment

restore
restores DCW lists from a segment created by the save command

execute, x
executes a DCW list

getstat, g
checks for status from a previous operation

block, b
blocks process until an event occurs

xs
executes a DCW list, but leaves process blocked until special interrupt occurs

xr
executes a DCW list repeatedly, until some unusual status is returned

xre
executes a DCW list repeatedly, regardless of whether the operations succeed or fail

status, st
sets the current status reporting mode

rs
    reprints the status from a previous operation

dump
    dumps data from the I/O buffer on to the terminal

patch
    inserts data into the I/O buffer from the terminal

pattern
    inserts data into the I/O buffer from the terminal by storing repeated copies of the data given

survey
    displays data returned by a "survey devices" tape controller command

dtstat
    displays data returned by a "read detailed status" tape handler command

chan
    selects a specific IOM and channel for I/O

time
    sets or prints the current time. limit for ioi timeout

prompt
    stores a character string to be used as a prompting message

susp
    suspends I/O on devices connected to an MPC by calling ioi_$suspend_devices

rel
    restores I/O on devices connected to an MPC by calling ioi_$release_devices

?
    types out the current DCW list number, current DCW number, and the current editor mode

    types the word "test_dcw" to verify that the test_dcw command is still in control

quit, q
    releases attached device and returns

I/O BUFFER AREA

Once the device is attached, an I/O buffer is allocated using the ioi_ subroutine. The default length is 1024 words, although this can be changed later. The first 32 words of the buffer are reserved for DCW lists, and the second 32 words are reserved for the ioi_ status queue. When constructing a DCW list, care should be taken to avoid modifying the first 64 words (100 octal) of the buffer, or results (especially status reporting) may be unpredictable.

DCW LIST PREPARATION

The test_dcw command contains an editor that can create and update DCW lists using simple input statements. Up to 32 different DCW lists, each up to 32 words in length, can be created and selectively updated and executed. Each DCW list also has a PCW associated with it that, if present, is used instead of the system-supplied PCW when the list is executed. The 32 DCW lists are numbered from 1 to 32 in decimal. Each DCW list can also be given a name. The 32 DCWs in each list are numbered from 0 to 37 in octal.

The DCW editor keeps track of several quantities as DCWs are entered. These are the current list, the current DCW number, and the current mode. When the test_dcw command is invoked, the current list is 1, the current DCW is 0, and the mode is update.

When a DCW is entered in update mode, the new DCW replaces the current DCW in the current list, and the current DCW number is increased by one.

The editor can also be placed in insert mode. In this mode, when a new DCW is entered, all DCWs starting with the current DCW are shifted one position down the list, the new DCW replaces the position formerly occupied by the old current DCW, and the current DCW is increased by one. DCWs shifted out of position 37 octal are lost.

The edit command can be used to select a DCW list to edit, as follows:

```
edit {list} {name}
```

where:

list
: is either the name or number of the DCW list to edit. The list can also be specified as "*", in which case, the first available empty list is used.

name
: is the name given to the DCW list selected by the first argument. If omitted, the name of the list is not changed.

This command sets the current list to the one specified, the current DCW to 0, and the mode to update. If the list argument is omitted, the current list is not changed, but the current DCW and mode are set to 0 and update respectively.

A DCW list can be given a name (or a new name) with the name command.

```
name {name}
```

where:

name
: is the name to be placed on the current list. If omitted, the current list becomes unnamed. If some other list has the name specified, that list becomes unnamed.

The mode of the editor is controlled by the insert and update commands, as follows:

```
update {n}
insert {n}
```

where:

n
: is a DCW number, in octal. The update command puts the editor in update mode and sets the current DCW to n. Similarly, the insert command places the editor in insert mode. If n is omitted, the current DCW is not changed.

A DCW can be deleted from the middle of the list with the delete command.

```
delete {n}
```

where:

n
: is a DCW number, in octal. DCW n is deleted by moving everything after it in the list up one position. If n is omitted, the current DCW is deleted. The current DCW number is not changed.

Any of the following commands can be used to create a DCW:

```
idcw
nidcw
tdcw
iotd
iotp
iontp
odcw
```

After a DCW is constructed with any of these commands, it is edited into the current list, in the current position, according to the current mode, as described above. In all of the DCW commands described below, all numeric quantities are entered in octal. Any of the parameters shown are optional, and if omitted, the corresponding DCW field is zero (except for the device address field that is set to the address of the device assigned).

To create an IDCW, the command is entered as follows:

```
idcw {di} {args}
```

where:

di
: is the value to be placed in the device instruction field.

args
: are used to set the remaining fields in the IDCW and can be selected from the following:

  da oo
  : places the value oo in the device address field.

  ci oo
  : places the value oo in the channel instruction field.

  ae oo
  : places the value oo in the address extension field.

  t oo
  : places the value oo in the tally field.

  ec
  : sets the extension control bit (ec bit).

  cont
  : sets the continue bit.

  mark
  : sets the marker status bit.

A nondata transfer IDCW can be entered more easily using the nidcw command. It is identical in format to the idcw command, but the tally defaults to 01 and the channel instruction defaults to 02.

A transfer DCW is created as follows:

```
tdcw {addr} {args}
```

where:

addr
: is the value to be placed in the address field.

args
: are used to set the remaining bits in the TDCW and can be selected from the following:

  ec
  : sets the extension change bit (ec).

  res
  : sets the restricted bit.

  rel
  : sets the relative mode bit.

IOTD, IOTP, and IONTP DCWs can be entered using the commands shown below.

```
iotd  {addr} {tally} {cp}
iotp  {addr} {tally} {cp}
iontp {addr} {tally} {cp}
```

where:

addr
: is the value to be placed in the address field.

tally
: is the value to be placed in the tally field.

cp
: is the value to be placed in the character position field.

Any arbitrary DCW can be entered using the odcw command.

```
odcw {word}
```

where:

word
: is the octal DCW to be used. If word is omitted, an all-zero (and invalid) DCW is created.

Each DCW list can have one PCW associated with it. The PCW can be entered with the following command:

```
pcw {di} {args}
```

where:

di
: is the value to be placed in the device instruction field.

args
: are any of the optional args listed under the idcw command, with the following additions:

  mask
  : sets the mask bit.

  reset
  : sets bits 21, 22, and 23 to form a reset PCW.

Any arbitrary PCW can be entered with the opcw command as follows:

```
opcw {word}
```

where:

word
: is the octal PCW to be used. If word is omitted, an all-zero PCW is created and the system-supplied PCW is used on subsequent executions of the DCW list.

The DCW list can be displayed at any time using the print command.

```
print {list}
```

where:

list
: is either the name or number of a DCW list. If list is omitted, the current list is displayed. If list is specified, the current list is set to that list, the current DCW is set to 0, and the mode is set to update. Instead of a list name, "all" can be used to indicate that all DCW lists are to be displayed, or "names" can be used to list the names of all DCW lists.

SAVING DCW LISTS

Once edited, a permanent copy of all the current DCW lists can be saved in a segment for later use by invoking the save command.

```
save path
```

where:

path
: is the pathname of the segment where the data is to be saved. The segment always has a suffix of "test_dcw", which is supplied automatically.

To restore the previously saved DCW lists, type:

```
restore path
```

where:

path
: is the name of the segment created by the save command. If the command was not invoked in debug mode, all IDCWs and PCWs are updated with the device address of the device currently assigned.

I/O BUFFER EDITING

Several commands are available to edit and display the contents of the I/O buffer. To enter data into the buffer, the patch command is used.

```
patch offset word1...word2...wordi
```

where:

offset
: is the octal offset in the buffer to be patched.

wordi
: is the value to be placed in word offset+i.

Offsets less than 100 octal should not normally be used, as this could interfere with the DCW list, or the status queue.

If a repeating pattern is desired, use the pattern command.

```
pattern offset repeats word1...word2...wordi
```

where:

offset
: is the octal offset in the buffer where the data is to start.

repeats
: is an octal number representing the number of times the data is to be repeated.

wordi
: are the data words to be repeated.

To display the contents of a buffer (in octal), use the dump command.

```
dump {offset} {length}
```

where:

offset
  is the offset in the buffer to be dumped. If omitted, 100 octal is assumed.

length
  is the number of words to dump, in octal. If omitted, 10 octal is assumed.

If the data consists of 8-bit bytes in binary mode (unaligned, 9 in each two words), the dump command can be used to dump them. The format is the same as the dump command, except that the data is displayed in binary, and the length is given in bytes, instead of words.

If the data to be displayed is the output of a survey devices command issued to a tape controller, a special command can be used to display the data in a more meaningful way.

```
survey {offset}
```

where:

offset
  is the location in the I/O buffer where the data has been stored. If the offset is omitted, 100 octal is assumed.

If the data to be displayed consists of the output of a read detailed status command issued to a tape handler, it can be displayed with:

```
dtstat {offset}
```

where:

offset
  is the location in the buffer where the status has been stored. If omitted, 100 octal is used.

EXECUTING THE DCW LIST

Once the DCW list is constructed, it can be executed as follows:

```
   execute {list}
```

where:

list
: is the name or number of the DCW list to execute. If omitted, the current list is executed. If specified, the current list is changed to that list, the current DCW is set to 0, and the mode is set to update. The current list is copied into the I/O buffer starting at 0, and ioi_$connect is called to connect to relative address 0. If the list executed has a PCW associated with it, a call is made to ioi_$connect_pcw instead. After the connect is made, the process becomes blocked until an interrupt occurs. The status of the interrupt is then printed. If the status indicates that the channel is still running, the process goes blocked again waiting for another interrupt. If the channel is not running, test_dcw is ready to accept another command after the status is displayed.

If the DCW list being executed generates a terminate interrupt and a special interrupt (such as loading a tape drive), the following command can be useful:

```
   xs {list}
```

This command is identical to the execute command, except that the process goes blocked after displaying the status from each interrupt until a special interrupt occurs.

A DCW list can be executed repeatedly using the following command:

```
   xr {list}
```

This command executes the DCW list specified without displaying any status until an error condition is detected. The final status is printed normally.

Another variation of this can be used when it is necessary to repeat the DCW list, even though it has errors.

```
   xre {list}
```

executes the list specified repeatedly regardless of the status. To terminate this, it is necessary to quit and to use the Multics program_interrupt command, afterwards.

Two other commands are occasionally useful in executing a DCW list.

```
block, b
getstat, g
```

The block command causes the process to go blocked waiting for an interrupt to occur. When it occurs, the resulting status is printed and test_dcw is ready for another command. The getstat command checks to see if any status is available, and prints it if it has occurred. The getstat command does not cause the program to go blocked if no status is available.

Using the block command (or if a channel fails), it is possible to put the process in a state where it is waiting for an event that never occurs. If this happens, a quit followed by a Multics program_interrupt command can be used to return to the test_dcw input routine.

STATUS REPORTING

Status is normally reported when received by printing it on the terminal. Status can be reported in three modes, as follows:

brief, bf
: is the default mode. The status message consists of the interrupt level in decimal, two words of IOM status in octal, and the major and minor status fields in binary.

long, lg
: consists of all eight words of the ioi_ status queue entry, in octal.

edited, ed
: is an English-language interpretation of the status.

The status mode is initially set to brief, but this can be changed as follows:

```
status {mode}
```

where:

mode
: is one of the three status modes described above. If omitted, the current mode is printed.

The previous status can also be redisplayed using the reprint status command.

```
rs {mode}
```

where:

mode
: is one of the three modes described above. If omitted, edited mode is assumed.

OTHER COMMANDS

Several other commands exist that can be useful. To set the length of the ioi_ timeout interval, use the time command.

```
time {n}
```

where:

n
: is the time limit in decimal seconds. If n is omitted, the command prints the current limit.

To change the size of the I/O buffer, type:

```
work {n}
```

where:

n
: is the buffer length desired in decimal words. If n is omitted, the work command displays the current buffer length.

To select a specific IOM and channel for I/O, the chan command can be used.

```
chan {iom} {channel}
```

where:

channel
: is the IOM channel, in octal.

iom
: is the IOM selected.

If channel is specified, but IOM is omitted, the IOM is assumed to be 1. If both are omitted, both are set to 0, indicating that ioi_ should make its own selection. The test_dcw command must be invoked with the -priv control argument in order to use this feature.

To suspend I/O on devices to connect to an MPC, use:

```
   susp
```

To restore I/O, use:

```
   rel
```

These commands call the appropriate ioi_ entry points to accomplish their task. They are valid only if test_dcw was invoked with the -priv control argument and the device is connected to an MPC.

To read the special device status stored by the previous operation, use:

```
   get_special_status
```

To read the detailed device status stored by the previous operation, use:

```
   get_detail_status
```

If a prompt message is desired when test_dcw is ready for input, you can supply one as follows:

```
   prompt {chars}
```

where:

chars
: is the data to be used for prompting. If chars is omitted, no prompt message is used.

To exit from the test_dcw command, type:

```
   quit, q
```

The I/O device currently attached is detached, and the program terminates.

Name: test_fnp

SYNTAX AS A COMMAND

```
test_fnp fnp_tag {-control_args}
```

FUNCTION

tests FNPs with the CSD-supplied FNP test programs.

ARGUMENTS

fnp_tag
: is the tag of the FNP to be tested. This FNP must have been shut down or dumped; it cannot be involved in testing by another process. Level 6 FNPs cannot be tested with this command.

CONTROL ARGUMENTS

-exec name
: specifies the FNP executive to be run initially. The name can be either "BOS" or "IOS." The default is BOS. (*Note*: BOS here does *not* refer to the Bootload Operating System.)

-input_switch name, -isw name
: specifies the I/O switch from which operator input is read. The default switch is user_input.

-message_switch name, -msw name
: specifies the I/O switch to which messages intended for the T&D line printer are written. The default switch is user_output. The FNP T&D programs generate output of this form if their query "IS A PRINTER AVAILABLE?" is answered affirmatively.

-output_switch name, -osw name
: specifies the I/O switch to which messages intended for the bootload console are written. The default switch is user_output.

NOTES

The FNP type of the FNP selected for testing is obtained from information contained in the Channel Definition Table (>system_control_1>cdt). If the user does not have access to this data base, the following query is issued:

```
   test_fnp:  What is the FNP type of FNP fnp_tag?
```

If the user hits RETURN, he is prompted with the following:

```
   Answer  DN6600, DN6670, DN355, or quit.
```

If the "quit" response in entered, control is returned to the current command processor.

Users should be familiar with the CSD offline version of TST3BT. The test options, queries, and message diagnostics relevant to FNP testing are produced by the FNP test programs themselves. The documentation for the offline version of TST3BT running under the PAS2 EXEC, and the T&D documentation for the FNP tests, contain information on actual dialogue with this program; it is the same as the dialogue with the offline version.

The bootload console of TST3BT is simulated by the Multics terminal controlling the process running test_fnp. By default, test output appears on the terminal, and responses are expected from the terminal. Normal Multics input line editing applies to all responses, and lowercase input is acceptable.

The response "quit" to any query of test_fnp, regardless of how it was generated, terminates the test session, releases the FNP, and returns to command level.

The REQUEST button of the bootload console is simulated by striking the QUIT key and using the program_interrupt (pi) command. to return to test_fnp. Normally, the REQUEST button causes an interrupt to be sent to the FNP directing the FNP executive to enter its request loop.

Access to the tandd_ gate is required. Access to >sc1>cdt is required to obtain the correct model number of the FNP. If you do not have access to the CDT, the default model number is DN6670. (This is the only FNP currently supported.)

The tests executed by test_fnp are sorted in the keyed sequential vfile_ >system_library_tandd>tandd_deck_file. These tests are loaded from the CSD-distributed "FNP binary deck tapes" by the load_tandd_library command (described in the *Multics Online Test and Diagnostics Reference Manual*, Order No. AU77).

Name: test_io_daemon

SYNTAX AS A COMMAND

```
test_io_daemon pathname
or
test_io_daemon -control_arg
```

FUNCTION

runs the I/O daemon subsystem in test mode in the user's process.

ARGUMENTS

pathname
: is the pathname of an iod_tables segment produced by the iod_tables_compiler command. The directory containing the iod_tables segment is used as the test directory.

CONTROL ARGUMENTS

-directory dir_pathname, -dr dir_pathname
: specifies the pathname of a test directory. The iod_tables segment contained in the directory is used as the test iod_tables segment.

NOTES

For more details on the use of this command, refer to the *System Maintenance Procedures* manual, Order No. AM81.

LIST OF REQUESTS

coord
: allows the coordinator part of your test process to come to command level. If you want to run just a coordinator, this request is the only one you have to issue. If you want to run both a coordinator and a driver, you have to follow this request with the driver request.

debug
: calls the system debug command to allow you to set and reset break points and execute interactive Multics commands (by using the ".." debug request). You may issue the debug request from both coordinator command level and driver command level. The system response to this request is "Calling debug."

driver
: allows the driver part of your test process to come to command level. If you want to run just a driver, this request is the only one you have to issue. If you want to run both a coordinator and a driver, you have to preceed this request with the coord request. *Note*: the driver request is not accepted by the coordinator part of your test process if you've suspended the driver part previously by using the coord request.

pi
: generates a program_interrupt signal. This request allows you to discard any undesirable output (or occurence) by generating a quit signal, and to then return to the last stack frame with a program_interrupt handler (i.e., debug or probe). Normally, you use this request to return to the debug or probe request after you've interrupted one of its functions with a quit signal. You may issue this request from both coordinator command level and driver command level.

probe
: calls the system probe command to allow you to set and reset break points and execute interactive Multics commands (by using the ".." probe request). You may issue the probe request from both coordinator command level and driver command level. The system response to this command is "Calling probe."

resume
: directs the driver to attempt recovery from iodd signal command level (in test mode, the driver will not attempt recovery of error conditions, but instead, after all error messages are displayed, will stop at iodd signal command level) or to return to normal command level from request command level or to quit command level (aborting any current request), as if it were not in test mode. You may only issue this request at driver command level.

return
: does the same thing as the logout command, except that it doesn't display any messages. You may issue this request from both coordinator command level and driver command level. When you issue it from driver command level, your process returns to coordinator command level (assuming you were running both a driver and a coordinator). The coordinator is not notified that the driver has logged out. When you issue the return request from coordinator command level, the entire coordinator/driver test environment is released and your process returns to the original process command level.

EXAMPLES

First, type either:

```
test_io_daemon >udd>SysMaint>Margie>iod>new_tables
```

or:

```
test_io_daemon -dir >udd>SysMaint>Margie>iod
```

The system will respond with:

```
Enter command: coordinator, driver, or return:
```

At this point, you have four choices:

1. If you want to run both a coordinator and a driver, proceed as follows:

```
! coord

 I/O Coordinator Version: x.x
 I/O Coordinator initialized

! driver

 I/O Daemon Driver Version: x.x
 Driver running in test mode.
 Enter command or device/request type:
 .
 .
 .
```

   To exit from test mode in this case, type "return" or "logout" twice.

2. If you want to run just a coordinator, proceed as follows:

```
! coord

 I/O Coordinator Version: x.x
 I/O Coordinator initialized
 .
 .
 .
```

   To exit from test mode in this case, type "return" or "logout" once.

3. If you want to run just a driver, proceed as follows:

```
! driver

 I/O Daemon Driver Version: x.x
 Driver running in test mode.
 Enter command or device/request type:
 .
 .
 .
```

   To exit from test mode in this case, type "return" or "logout" once.

4. If you want to return to Multics, type "return."

Name: test_tape

SYNTAX AS A COMMAND

```
test_tape {-control_args}
```

FUNCTION

tests a tape drive or tape reel.

CONTROL ARGUMENTS

-comment STR, -com STR
: allows you to pass additional information about the requested volume mount to the operator.

-compare STR, -comp STR
: writes and then reads a tape on device STR1, and then automatically has the operator mount the tape on device STR2 and read the tape. The mounting and reading continues to device STRn. At least two devices must be specified. Only one device is attached at a time. The full device name (e.g., -comp tapa_05 tapa_07) must be used. This control argument cannot be used with -device.

-count N, -ct N
: indicates the number of records to be written or read, where N is a decimal integer. Each write operation creates one 1040 word physical record. If this control argument is not given, then the entire tape is written or read.

-density N, -den N
: indicates the tape density, where N can be either 6250, 1600, or 800. The default is 1600.

-device STR, -dv STR
: selects a specific tape unit; STR must be the complete device name. If this control argument is not given, the system finds a free tape unit (e.g., -device tapb_08). It is incompatible with -compare.

-no_data_compare, -ndc
: disables comparison of the data read to a known pattern. This control argument is useful for verifying that a tape can be read without knowing what data is on the tape.

-pattern N, -ptrn N
: specifies N as the word of octal data to fill the data buffers, where N can be a maximum of 12 octal digits. If fewer than 12 digits are given, the field is padded on the left with zeroes. If this control argument is not given, a pattern of 222222222222 is used. The -pattern control argument cannot be used with -random.

-random
: fills the data buffers with a known random data pattern. It cannot be used with -pattern.

-raw
: displays raw hex detailed status with each error message in addition to an interpreted display.

-read, -r
: identifies the mode of the test. The tape is mounted without a write ring and the read-only pass is performed.

-track7, -tk7
: specifies a 7-track tape drive as the test unit. The default is 9 track.

-volume ID, -vol ID
: specifies a tape by its volume identification number, which can have a maximum of nine characters. If -volume is not given, a default of "test_tape" is used.

-wait N, -wt N
: attempts to attach the device N times, after one-minute waits, if the device desired is being used by another process. If after N waits the device still cannot be attached, the program bypasses the device. The default for N is two times.

-write, -w
: identifies the mode of the test. The tape is written and the read pass is bypassed.

-write_read, -wr
: identifies the mode of the test. The tape is written and the read pass is performed. This is the default.

NOTES

The test_tape command senses the End of Tape Mark (EOT) and stops even if the record count has not been exhausted. Typing test_tape with no control arguments has the same effect as:

```
test_tape -vol test-tape -den 1600 -ct 100000 -ptrn 222222222222 -wr
```

Listed below is a summary of the default control argument values.

-volume	(test-tape)	-count	(100000{entire tape})
-comment	(NONE)	-ndc	(OFF)
-device	(one previously assigned, or a free device)	-random	(OFF)
-compare	(OFF)	-pattern	(222222222222)
-density	(1600)	-write	(ON)
-track	(9)	-read	(ON)
-wait	(OFF)	-raw	(OFF)

Name: total_time_meters, ttm

SYNTAX AS A COMMAND

```
ttm {-control_arg}
```

FUNCTION

prints out the CPU time percentage and average CPU time spent doing various tasks.

CONTROL ARGUMENTS

-report_reset, -rr
  generates a full report and then performs the reset operation.

-reset, -rs
  resets the metering interval for the invoking process so that the interval begins at the last call with -reset specified. If -reset has never been given in a process, it is equivalent to having been specified at system initialization time.

ACCESS REQUIRED

This command requires access to phcs_ or metering_gate_.

NOTES

If the total_time_meters command is given with no control argument, it prints a full report.

The following are brief descriptions of each of the variables printed out by total_time_meters. Average CPU times are given in microseconds. In the description below, system CPU time is the total amount of CPU time generated by all configured CPUs. Idle time is CPU time consumed by an idle process; an idle process is given a CPU only if no other (nonidle) process can be given that CPU. System nonidle time is the difference between system CPU time and the aggregate idle time. In this computation, MP idle time, work class idle time, and loading idle time are considered as overhead time and are included in system nonidle time. That is, system idle time is defined to include only the idle time caused by light load; it does not include the idle time caused by system bottlenecks; that time is counted as overhead.

The three columns in the display contain, respectively, the percent of system CPU time, percent of system nonidle time, and average time per instance (for the overhead tasks). The percents of non-idle time are included to assist the user in comparing values measured under light load with those measured under heavy load. It can not be emphasized too often that measurements made under light load should not be used to make tuning or configuration decision.

Several of the overhead task names are indented, to indicate that they are part of the preceding, non-indented task. The percents for these indented tasks are also included in the percent for the preceding task. That is, in the example at the end of this description, page faults used 1.49% of system CPU time; 0.14% was used by PC Loop Locks, and the remaining 1.35% was used by other page fault overhead.

Page Faults
: is the percentage of CPU time spent handling page faults and the average time spent per page fault.

PC Loop Locks
: is the percentage of CPU time spent looping on the page table lock, and the average time spent per looplocking. This number will be nonzero only on a multiprocessor system. This number is also included in page fault time.

PC Queue
: is the percentage of CPU time spent processing the core queue, and the average time spent per core queue processing. The core queue is used to prevent loop looks in page control on interrupt side. If an interrupt for a page I/O is received when the page table is locked, an entry is made into the core queue. When the page table is next unlocked, the core queue is processed.

Seg Faults
: is the percentage of CPU time spent handling segment faults, and the average time spent per segment fault. These values do not include the time spent handling page faults that occurred during the segment fault handling.

Bound Faults
: is the percentage of CPU time spent handling bound faults and the average time spent per bound fault. These values do not include time spent handling page faults that occurred during bound fault processing.

Interrupts
: is the percentage of CPU time spent handling interrupts, and the average time spent per interrupt.

Other Fault
: is the percentage of CPU time spent handling certain other faults. The fault processing time included is fault handling time that is not charged to the user process as virtual CPU time and that does not appear elsewhere in the total_time_meters output (i.e., it is not page fault, segment fault, or bound fault processing). The vast majority of the time included as Other Fault processing is related to the processing of connect faults and timer_runout faults.

Getwork
: is the percentage of CPU time spent in the getwork function of traffic control, and the average time spent per pass through getwork. The getwork routine is used to select a process to run on a CPU and to switch address spaces to that process. This number is also included in other fault time.

TC Loop Locks
: is the percentage of CPU time spent looping on a traffic control lock, and the average time spent per looplocking. The locks included in this category are the global traffic control lock and the individual Active Process Table Entry (APTE) locks. This time is nonzero only on a multiprocessor system. This number is also included in other fault time.

Post Purging
: is the percentage of CPU time spent in post purging processes that have lost eligibility, and the average time spent per post purge. Post purging a process involves moving all of its per-process pages that are in main memory into the "most recently used" position in the core map and computing the working set of the process. This time is nonzero only if the "post_purge" tuning parameter is set to "on." This number is also included in other fault time.

MP Idle
: is the multiprogramming idle. This is the percentage of CPU time that is spent idling when processes are contending for eligibility, but not all contending processes are eligible. This occurs because some site-defined or system limit on eligibility has been reached--e.g., maximum number of eligible processes (tuning parameter "max_eligible"), maximum number of ring 0 stacks (tuning parameter "max_max_eligible"), per-work-class maximum number of eligible processes, working set limit, etc. MP idle is CPU time wasted in idling because the eligibility limits are set too low for the configuration, or because there is not enough memory in the configuration to hold the working sets of a larger number of eligible processes.

Work Class Idle
: is the percent of CPU time spent idling because the only processes that could have been run belonged to work classes that had used their maximum percentage of CPU time. Setting upper limits on work classes will cause the system to go idle rather than run processes in those work classes that have reached their maximum percent. This meter indicates the percent of CPU time wasted in idling because of the setting of these limits.

Loading Idle
: is the percentage of CPU time that is spent idling when processes are contending for eligibility, not all contending processes can be made eligible, and some eligible processes are being loaded. Being loaded means wiring the two per-process pages that must be in main memory in order for a process to run--the first page of the descriptor segment (DSEG) and the first page of the process descriptor segment (PDS).

NMP Idle
: Is the nonmultiprogramming idle--the percentage of system CPU time that is spent idling when all processes contending for eligibility are eligible. Time is charged to NMP idle under two different circumstances: 1) there are fewer processes contending for eligibility than there are processors in the configuration; 2) there are fewer non-waiting processes than there are processors in the configuration (that is, most of the eligible processes are waiting for system events such as page faults), and no additional processes are contending for eligibility. Both of these circumstances are caused by light load; therefore NMP idle time, along with zero idle time, is subtracted from system CPU time to get system non-idle time.

Zero Idle
: is the percentage of system CPU time that is spent idling when no processes are ready and contending for eligibility.

Other Overhead
: is the percentage of system CPU time that is overhead but cannot be attributed to any of the above categories of overhead. This is almost entirely instrumentation artifact, due to a small but indeterminable amount of time between the occurrence of a fault or interrupt and the reading of the system clock (which begins the charging of time to some overhead function). Due to hardware features such as cache memory and associative memory, this time is not constant per fault, even though the same instruction sequence is executed each time. Other Overhead represents the effect of this nondeterminism.

Virtual CPU Time
: is the precent of CPU time delivered to user processes as virtual CPU time. Virtual CPU time is time spent running user ring code (commands, application programs, etc.) or inner ring code in direct response to user ring requests (via gate calls). System virtual CPU time is total system CPU time less all system overhead and idle time. It is the sum of the virtual CPU time charged to all processes. One objective of tuning is to maximize virtual CPU time.

EXAMPLES

The following is an example of the information printed when the total_time_meters command is invoked with no control argument.

```
Total metering time    91:33:53

                          %       %NI       AVE

Page Faults             1.49     4.28    2301.466
  PC Loop Locks         0.14     0.41    3439.733
PC Queue                0.17     0.49     306.381
Seg Faults              0.84     2.40    9628.827
Bound Faults            0.05     0.14   15850.365
Interrupts              2.66     7.61    1959.442
Other Fault             3.17     9.07
  Getwork               1.49     4.27     638.160
  TC Loop Locks         0.08     0.24     309.842
  Post Purging          0.09     0.25     790.584
MP Idle                 0.20     0.58
Work Class Idle         0.09     0.26
Loading Idle            0.02     0.05
NMP Idle               36.36
Zero Idle              28.72
Other Overhead          0.10     0.29
Virtual CPU Time       26.13    74.84
```

Name: traffic_control_meters, tcm

SYNTAX AS A COMMAND

```
tcm {-control_args}
```

FUNCTION

prints out the values of various traffic control meters.

CONTROL ARGUMENTS

-counters, -ct
: prints out the number and frequency of certain paths through the traffic controller.

-gen
: prints out general traffic control information.

-queue, -qu
: prints out certain resource usage as a function of depth in the eligible queue.

-report_reset, -rr
: generates a full report and then performs the reset operation.

-reset, -rs
: resets the metering interval for the invoking process so that the interval begins at the last call with -reset specified. If -reset has never been given in a process, it is equivalent to having been specified at system initialization time.

ACCESS REQUIRED

This command requires access to phcs_ or metering_gate_.

NOTES

If the traffic_control_meters command is given with no control arguments, it prints a full report.

The following meters reflect activity of the traffic controller, and some constants used therein. They are printed if the -gen control argument is specified.

Ave queue length
: is the average number of processes in the eligible and priority queues. This is the average number of ready, waiting, or running processes.

Ave eligible
: is a recent average of the number of eligible processes.

Response time
: is the average time between a process' receiving an interactive wakeup and the awarding of eligibility to the process. The response time seen by the user is larger than this meter.

The following meters pertain to the number and frequency of certain paths through the traffic controller. They are printed if the -ct control argument is specified.

Interactions
: is a count of, and the average time between, terminal interactions.

Loadings
: is a count of, the average time between, and the number per interaction of process loadings.

Blocks
: is a count of, and the average time between, calls to "block" to block some process.

Pauses
: is a count of the number of times processes were delayed in ring 0. These pauses are due to suspected covert channel activity. Refer to "Segment State Change" under the output of the file_system_meters command.

Wakeups
: is a count of, and the average time between, wakeup signals being sent.

Schedulings
: is a count of, the average time between, and the number per interaction of trips through the scheduler/rescheduler function that caused priorities to be changed.

Lost priority
: is the number of times the alarm clock went off indicating a priority process that had been running lost its eligibility because it had used up its eligible time; i.e., its eligible time exceeded the CPU quantum that the process remains in the queue. The process reenters the traffic controller to be rescheduled.

Priority boosts
: is the number of times the alarm clock went off indicating a priority scheduling process on the ready list should be granted high priority; i.e., have its waiting time before rescheduling set to 0. The process is then resorted into the ready list with its new, higher priority.

Wait Page
: is a count of, the average time between, and the number per interaction of calls to force some process to a wait state in order to wait for page transfer.

Wait PTL
: is a count of, the average time between, and the number per interaction of calls to force some process to a wait state in order to wait for the page table lock.

Wait Other
: is a count of, the average time between, and the number per interaction of calls to force some process to a wait state in order to wait for events other than page control events.

Total Waits
: is a count of, the average time between, and the number per interaction of calls to force some process to a wait state.

Notify Page
: is the number of, and average time between, calls to notify processes waiting for page transfer events.

Notify PTL
: is the number of, and average time between, calls to notify processes waiting for page table unlockings.

Notify Other
: is the number of, and average time between, calls to notify processes waiting for all other events.

Total Notifies
: is the number of, and the average time between, notify calls (i.e., returning a waiting process to the ready state).

Get Processor
: is the number of, and average time between, calls to get_processor. Get_processor is called at notify time to find a CPU on which to run the notified process. An idle process or lower priority running process is preempted.

Pre-empts
: is a count of, average time between, and the number per interaction of process preemptions and timer runout faults.

Getwork
: is the number of, and average time between, calls to getwork. Getwork is the dispatcher portion of the scheduler; it finds a process to run on the executing CPU.

Retry getwork
: is the number of, and average time between, retries of the getwork function.

Extra notifies
: is the number of, and average time between, notify calls that found no process waiting on the notified event.

Last EN event
: is the last notified event for which no process was waiting.

Notify timeout
: is the number of times a notify was not received by a waiting process within notify_timeout_interval (a tuning parameter). This is printed only if the count is nonzero.

Last NTO event
: is the last event on which a notify timeout occurred.

The following meters pertain to the eligible queue. They are printed if the -qu control argument is specified.

Depth
: is the depth of the process within the eligible queue. A process deep in the eligible queue is run only if processes above it cannot run.

%PF
: is the percentage of page faults that occurred from processes at this depth.

TBPF
: is the average time between page faults at this depth.

%GTW
: is the percentage of getwork calls being made when a member of this priority relinquishes control.

TBS
: is the average time between getwork calls at this priority level.

%CPU
: is the percentage of CPU time consumed by members of this priority.

EXAMPLES

The following is an example of the information printed when the traffic_control_meters command is invoked with no control arguments. (See Appendix A for a representation of the configuration deck used to create the system from which the metering samples were taken.)

```
Total metering time    0:15:34

Ave queue length    17.13
Ave eligible        12.14
Response time        0.201 sec

  COUNTER          TOTAL          ATB          #/INT

Interactions        9044       0.103 sec
Loadings           15966       0.059 sec      1.765
Blocks             16221       0.058 sec
Pauses                 2     467.187 sec
Wakeups            23147       0.040 sec
Schedulings        16432       0.057 sec      1.817
Lost priority          1     934.375 sec
Priority boosts        0       0.000 sec
I/O boosts           276       3.385 sec
Wait Page         105154       8.886 msec    11.627
Wait PTL           55014      16.984 msec     6.083
Wait Other         44769      20.871 msec     4.950
Total Waits       204937       4.559 msec    22.660
Notify Page       107309       8.707 msec
Notify PTL         55014      16.984 msec
Notify Other       43562      21.449 msec
Total Notifies    205885       4.538 msec
Get Processor     220207       4.243 msec
Pre-empts          92336      10.119 msec    10.210
```

```
Getwork             311046       3.004 msec
Retry getwork         8035       0.116 sec
Extra notifies        5491       0.170 sec
 Last EN event      144162154163

 DEPTH     %PF    TBPF    %GTW    TBS    %CPU

   1      10.4   29.3     9.5    13.6    7.6
   2      10.7   23.0     8.6    12.1    6.1
   3      10.3   25.5     8.3    13.5    6.5
   4       9.4   31.1     8.0    15.4    7.2
   5       8.9   35.2     7.9    16.8    7.8
   6       8.3   39.5     7.6    18.3    8.1
   7       7.6   47.7     7.4    20.7    8.9
   8      34.2   56.6    42.7    19.1   47.8
```

Name: traffic__control__queue, tcq

SYNTAX AS A COMMAND

```
tcq {-control_arg}
```

FUNCTION

prints out the state of the traffic control queue at the time of the call.

CONTROL ARGUMENTS

-all
: print information about all processes. The default is to print information only for processes in ready queues.

ACCESS REQUIRED

This command requires access to phcs_ or metering_gate_.

NOTES

The following items are printed out by the traffic_control_queue command.

avq
: is the average number of processes in the eligible and priority queues. This is the average number of ready, waiting, or running processes.

elapsed time
: is the time since traffic_control_queue was last called. This equals 0 if it is the first time the program was called for the given process.

active last 15 sec.
: is the number of processes that changed state during the last 15 seconds.

The following items are printed out for each user presently in the ready queue.

flags
: are one-bit indicators in the active process table (APT) entry for the user.

The following flags are printed:

```
E  process is eligible
W  Interprocess Communication (IPC) wakeup pending
S  stop pending
P  process being preempted
L  process is loaded
D  process has descriptor base register loaded
H  process is a hardcore process
I  process is an idle process
```

The flags are preceded by a letter indicating the state of the process. The allowed states are:

```
e  empty or unused
x  running
r  ready
w  waiting
b  blocked
s  stopped
p  waiting for page table lock
```

If the flag is followed by a parenthesized letter, the letter is the CPU tag of the processor on which that process must be run.

dtu
: is the incremental CPU time (in seconds) the process has used since the tcq command was last called.

dpf
: is the incremental number of page faults the process has taken since the tcq command was last called.

temax
: is the value (in milliseconds) of temax of the process. Temax is the maximum amount of CPU time the process may use in the current eligibility quantum.

te
: is the value (in milliseconds) of te of the process. Te is the amount of CPU time used in the current eligibility quantum.

ts
: is the value (in milliseconds) of ts of the process. Ts is the amount of CPU time used since scheduling priority changed.

ti
: is the value (in milliseconds) of ti of the process. Ti is the amount of CPU time used since the process interacted, or the tuning parameter timax, whichever is less.

tssc
: is the real time (in seconds) since the state change of the process.

event
: is the event for which the process is waiting. If this value is 0, the process is not waiting.

d
: is the device identifier of the device containing the page, if the process is waiting for a page. This is not currently used.

ws
: is the modified value of the working set estimate being used for the process.

wc
: is the number of the work class to which the process belongs.

process
: is the name of the user who owns the process.

workclass credits
: is the value (in milliseconds) of CPU time used by the workclass.

EXAMPLES

The following is an example of the information printed when the traffic_control_queue command is invoked with no control argument. (The last column, containing the Person_ids representing the process flag, has been omitted due to space limitations.)

```
avq = 19, elapsed time = 108 sec, 28 active last 15 sec.
flags     dtu     dpf temax   te     ts     ti     tssc event d  ws wc
wWLE        4     760  2097 1273      0      0    0.032 63336 0   0  3
xLED        3     555  1000  162      0      0    0.000     0 0  51  3
wWLE        2     253  2097  543      0      0    0.107 10623 0   0  3
xWLED       0      74  2097  106      0      0    0.004     0 0   0  1
wLE         3     539  1000   30      0      0    0.027 61200 0  28  3

REALTIME QUEUE:

INTERACTIVE QUEUE:
rW          1     129  1000    0      0      0    2.137     0 0  27  3
rW          3     670  1000    0      0      0    2.040     0 0  17  3
rW          5    1093  1000    0      0      0    1.645     0 0  30  1
rW          2     303  1000    0      0      0    0.999     0 0  37  4
rW          1      96  1000    0      0      0    0.948     0 0  28  1
rW         13    2786  1000    0      0      0    0.920     0 0  75  3
rW          2     503  1000    0      0      0    0.459     0 0  24  4
rW          3     485  1000    0      0      0    0.444     0 0  26  1

WORKCLASS   1 QUEUE: credits = 126 ms.
r          15    2260   750    0   2263   3258   28.436     0 0  47  1

WORKCLASS   2 QUEUE: credits = 143 ms.
rW          1     189   750    0    911      0   33.195     0 0  56  2
r          10    1665   750    0   1503   1000   21.679     0 0  44  2

WORKCLASS   3 QUEUE: credits = 1618 ms.
rW          5     694   750    0      0   1000    8.309     0 0  31  3
r           4     433   750    0   3759   3260   35.030     0 0  42  3
rW          8    1672   750    0   2257   8000   23.642     0 0  50  3
rW          6    1311   750    0    753   8000   18.021     0 0  78  3

WORKCLASS   4 QUEUE: credits = 152 ms.
rW          1     292   750    0   1764   1082   33.277     0 0  15  4
```

Name: tty_dump

SYNTAX AS A COMMAND

```
tty_dump channel_name {-control_args}
```

FUNCTION

displays on the user's terminal the contents of the ring zero data bases describing either the current state of selected communications channels managed by the Multics Communication System or the state of such channels at the time of a system crash.

ARGUMENTS

channel_name
    specifies the communications channels for which the state is to be displayed. The star convention is allowed (e.g., b.h202.**). This argument is incompatible with the -user control argument.

CONTROL ARGUMENTS

-ascii
    specifies that the contents of buffers are to be interpreted as ASCII characters in addition to being displayed as octal or hexadecimal values.

-all, -a
    specifies that, for each channel selected by the above arguments, information is to be displayed from the data bases of the channel, its parent multiplexer, its grandparent multiplexer, etc., up to the top level multiplexer channel. For example, for b.h202.prt1, all information from the data bases of b.h202.prt1, b.h202, and b that is related to b.h202.prt1 would be displayed.

-brief, bf
    suppresses display of the buffer contents. Only the addresses, size, and flags for each buffer are displayed.

-ebcdic8
    specifies that the contents of buffers are to be interpreted as EBCDIC (8-bit byte) characters in addition to being displayed as octal or hexadecimal values.

-ebcdic9
    specifies that the contents of buffers are to be interpreted as EBCDIC (9-bit byte) characters, in addition to being displayed as octal or hexadecimal values.

-erf N
    specifies that information about the channels is to be taken from the system dump associated with error report form (ERF) N located in the dumps search list. If this control argument is omitted, information about the currently running system is displayed. This control argument is incompatible with the -user control argument.

−hex8
: specifies that the contents of buffers are to be displayed as hexadecimal values, in addition to any character interpretation. Each 8−bit byte in a word is displayed (nine hexadecimal digits).

−hex9
: specifies that the contents of buffers are to be displayed as hexadecimal values, in addition to any character interpretation. The low order 8 bits of each 9−bit byte in a word is displayed as two hexadecimal digits.

−lcte
: specifies that the logical channel table entries (LTE) for the selected channels are to be displayed in addition to the other information normally displayed. If −all is specified, the LCTEs of all parent multiplexers are also displayed.

−long, −lg
: specifies that the contents of any input and output buffers for the channels are to be displayed. This is the default.

−octal
: specifies that the contents of buffers are to be displayed as octal values in addition to any character interpretation. Octal is the default numeric mode for buffer contents display.

−subchan, −sbc
: specifies that information from the data base of the parent multiplexer related only to the selected channels is to be displayed.

−user STR
: specifies that the state of all communications channels attached by the specified user(s) is to be displayed. STR is a starname used to identify the users and is matched against the Person_id.Project_id of each logged in user. For example, "*Smith.M*" would match any user whose Person_id ends with "Smith" that is logged in on a project that starts with "M". This control argument is incompatible with the channel_name argument and the −erf control argument.

ACCESS REQUIRED

Use of the tty_dump command without the −erf control argument requires access to the gate phcs_.

NOTES

The description of the dump_segment command in the *Multics Commands and Active Functions* manual, Order No. AG92, provides detailed information on the various buffer display formats.

The default mode for buffer displays is to display their contents as octal values without any character interpretation.

There are two sets of conflicting control arguments in tty_dump: three with which to specify the base of numeric display (-octal, -hex8, and -hex9), and three with which to specify character code interpretation (-ascii, -ebcdic8, and -ebcdic9). If conflicting control arguments are given on the command line, the last one specified will be used.

EXAMPLES

The command line:

```
   tty_dump b.h202.** -all
```

displays the state of multiplexer b.h202 and its subchannels. Displayed information includes the WTCBs/TCBs of the subchannels, multiplexer-specific data for the subchannels, global data for the multiplexer, and the PCB of the multiplexer's physical FNP channel.

By comparison, the command line:

```
   tty_dump b.h202 -all
```

displays only global multiplexer data and the PCB.

Name: tty_lines

SYNTAX AS A COMMAND

```
tty_lines {arg}
```

FUNCTION

prints information about communications channels defined in the channel definition table (CDT). An optional argument can be used to print information about a subset of channels.

ARGUMENTS

id STR
: prints information about channels on which the terminal most recently dialed up has an identification code specified by STR. The string STR is four characters long.

dlN
: prints information about each channel that has been dialed up N times or more.

d=N
prints information about each channel that has been dialed up exactly N times.

stN
prints information about each channel whose current state code is N (see "Notes" below).

acN
prints information about each channel whose activity code is N (see "Notes" below).

slN
prints information about the Nth entry in the CDT.

-type STR
prints information about each channel on which the most recently dialed terminal was of the terminal type specified by STR.

NOTES

If the tty_lines command is given with no argument, it prints information about all channels in the CDT. For each channel, a line is printed in the following format:

```
NAME   TYPE          D S  W A  BAUD Person_id Project_id (ID) C
```

NAME
is the channel name, e.g., a.1006.

TYPE
is the terminal type that has most recently dialed the channel, or NU if the channel has not been used.

D
is the number of times the channel has been dialed up.

S
is the current state of the channel. It may have one of the following values:

```
1   hung up
2   listening (ready for dialup)
5   dialed
```

W
is an internal variable indicating what the answering service expects to happen next to the channel.

A
is the activity code for the channel. It may have one of the following values:

```
1   hung up
2   listening (ready for dialup)
3   dialed up but not logged in
4   user is logged in but process not yet created
5   user process has channel
6   auto_call line is in process of dialing out
7   auto_call line is in use (dial complete)
```

BAUD
is the baud rate of the channel

Person_id
is the Person_id of the current user of the channel. If A is not 4 or 5, this field is omitted.

Project_id
is the Project_id of the current user of the channel. If A is not 4 or 5, this field is omitted.

ID
is the identification of the terminal currently using the channel. If S is not 5, this field is omitted.

C
is the comment field from the CDT entry.

Name: tune_disk

SYNTAX AS A COMMAND

```
tune_disk DRIVE_NAME IO_TYPE -load N -response N
or
tune_disk reset_max
or
tune_disk reset_sys
or
tune_disk stagnate N
or
tune_disk system IO_TYPE {-max n} {-map IO_TYPE}
```

FUNCTION

alters disk tuning parameters. A description of disk tuning techniques can be found in the *Multics System Maintenance Procedures* manual, Order No. AM81.

ARGUMENTS

DRIVE_NAME
is the name of the disk drive to be tuned. (For example dska_05).

IO_TYPE
identifies one of the I/O types tunable by tune_disk, where IO_TYPE can be one of the following:

```
page_read
page_write
vtoce_read
vtoce_write
test_read
test_write
```

reset_max
requests that all queue maximum depth meters be reset in the disk_seg database. The time and date at which the meters were last reset is also maintained in the database. This argument is useful to permit a new/lower max depth to be seen after altering tuning parameters, or after an Allocation Lock has occurred.

reset_sys
requests that all system depth counters be reset to 0. This is useful after altering system depth counter mapping. If counter mapping has been changed while requests were in the queue, the counter which had been used may be left artificially high. Resetting back to 0 lets the system correct the value.

stagnate N
specifies a change of the system wide stagnation time period to the specified number of seconds. Tune_disk sets a maximum stagnation time period of 6 minutes.

system
indicates modification of a system-wide optimization factor. The maximum depth and/or mapping for the specified io_type will be latered. If neither a maximum depth value, nor a mapping is altered an error message is issued.

CONTROL ARGUMENTS

-load N, -ld N
: defines the optimization maximum queue loadpoint for the specified drive. The value N is stated in terms of queue elements. For blocking I/O, this value would typically reflect a point which preserves sufficient multiprogramming. For non-blocking I/O, this would typically reflect a point before resource saturation would occur and cause the I/O type to become blocking. The -load value is one of the two points (along with -response) that define the optimization line. If -load 1 is specified, the initial response value is the optimizing multiplier and no load optimization is performed.

-map IO_TYPE
: specifies that the current depth counting for the specified system-wide optimization entry should be done using the counter for io_type. For example:

```
        tune_disk system PageRead -map PageWrite
```

Would have the depth counter for PageWrite used to accumulate the number of PageRead IO's currently outstanding.

-max N
: indicates that the maximum depth for the specified system-wide optimization entry should be set to n. If this depth is reached then full optimization of this IO type will be done system wide for all drives.

-response N, -rsp N
: defines the optimization maximum response value. This value is the multiplier to be used for an IO_TYPE queue load of a single request.

ACCESS REQUIRED

This command requires access to the hphcs_ gate.

NOTES

Refer to the *Multics System Maintenance Procedures* manual, Order No. AM81 for a description of disk tuning techniques.

Name: tune_work_class, twc

SYNTAX AS A COMMAND

```
twc -work_class N -control_args
```

FUNCTION

sets or changes the scheduling parameters for a single work class.

ARGUMENTS

-work_class N, -wc N
    specifies the work class for which scheduling parameters are to be set.

CONTROL ARGUMENTS
    are the parameters to be set, and can be chosen from the following (at least one must be specified):

-governed STR, -gv STR
    controls the limitation of CPU resources to the work class. STR can be "off," in which case there is no limitation for the work class; or STR can be a number between one and 100, which represents a percentage of total system CPU time. In this case, the work class is limited to the specified percentage of total system CPU time.

-int_response N, -ir N
    is the desired response time, in decimal seconds, after an interaction.

-int_quantum N, -iq N
    is the quantum (time slice), in decimal seconds, given after an interaction.

-int_queue STR
    controls the use of the interactive scheduler queue by users in the work class. STR can be "on", in which case users in the work class who have interacted recently are given priority over users in all work classes who have not interacted recently. STR can also be "off", in which case users in the work class who have interacted recently do not receive priority. The default is "off" for governed work classes and "on" for all other work classes.

-response N, -r N
    is the time, in decimal seconds, between successive quanta.

-quantum N, -q N
    is the quantum, in decimal seconds, given when an interaction has not just occurred.

-pin_weight N, -pw N
    sets the pin weight of the work class to N. The default is 3 for the Initializer, and 0 for all other work classes.

-post_purge STR, -pp STR
: controls post purging of processes in the work class, where STR can be "on" or "off." If on, processes are post purged if post purging is enabled for the system; if off, processes are not post purged.

-realtime STR, -realt STR
: places the work class in realtime mode if STR is "on"; removes the work class from realtime mode if STR is "off."

-wc_max_eligible N
: applies eligibility constraints to processes in the work class, where N is an integer. If N is nonzero, no more than N processes are eligible at one time; if N is zero, only system-wide eligibility constraints are applied.

ACCESS REQUIRED

You need access to hphcs_.

NOTES

If the system scheduler is in percent mode, and the specified work class is not in realtime mode, the values of int_response, int_quantum, response, quantum, and wc_max_eligible have no effect on the system's operation.

If the system scheduler is in deadline mode, or the specified work class is in realtime mode, the values of governed have no effect on the system's operation.

This command is useful for setting scheduler parameters on a temporary basis. Parameters set by it are overridden by the values in the master group table (MGT) at shift change time, if a new MGT is installed, or if the operator issues the command line "maxu auto."

Name: unlock_mca

SYNTAX AS A COMMAND

```
unlock_mca mca_number
```

FUNCTION

unlocks (enables) input to the maintenance channel adapter (MCA) specified by the argument.

ARGUMENTS

mca_number
    is the decimal number of the MCA to be unlocked.

Name: unwire_pages

SYNTAX AS A COMMAND

```
unwire_pages path
```

FUNCTION

reverses the effect of the wire_pages command. All pages that were wired are unwired (i.e., no longer guaranteed to remain in main memory).

ARGUMENTS

path
    is the pathname of the segment whose pages are to be unwired.

This page intentionally left blank.

Name: up_ctr

SYNTAX AS A COMMAND

```
up_ctr
```

FUNCTION

is called by biller.ec at the end of a billing period. For each requisition in the reqfile, the up_ctr command adds "charges this month" to "charges this requisition" and resets "charges this month" to zero. It prints one number, the total of all "charges this month" fields.

Name: update_mail_table_entry

SYNTAX AS A COMMAND

```
update_mail_table_entry name {address} {-control_args}
```

FUNCTION

modifies an entry in the mail table.

ARGUMENTS

name
: specifies the name of the mail table entry which is to be updated. The name must be enclosed in quotation marks if it contains blank spaces. If the name is an alias, the associated normal entry is updated.

address
: is a destination specifier, that is, a mailing address in the form used by the -mailbox or -user control arguments (see below). If this is given, it becomes the new mail address of the entry.

CONTROL ARGUMENTS

-acs_path path
: specifies the Access Control Segment (ACS) which controls who may update the entry's mail address. rw access to the ACS indicates that a user may update the entry. If path is the null string there will be no ACS, and only system administrators (anyone with access to mail_table_priv_) may update the entry.

-alias name
: specifies an alternate name for the entry. If this is specified multiple times, each alias is added.

-delete_alias name
: causes the specified alternate name to be removed from the entry. In order to maintain consistency with the person name table (PNT), this alias may not be the user's login alias.

-log path
: specifies the pathname of a logbox and is equivalent to:

```
-mailbox >udd>Project_id>Person_id>Person_id.sv.mbx
```

If this used, the pathname specified becomes the new mail address of the entry.

-mailbox path, -mbx path
: specifies the pathname of a mailbox. The suffix "mbx" is added if necessary. If this is used, the pathname specified becomes the new mail address of the entry.

-mailing_list path, -mls path
: specifies the name of a mailing list. The suffix "mls" is added if necessary. The archive component pathname convention is accepted. If this control argument is used, the pathname specified becomes the new mail address of the entry.

-meeting path, -mtg path
: specifies the pathname of a Forum meeting. The suffix "control" is added if necessary. If this is used, the pathname specified becomes the new mail address of the entry.

-save path, -sv path
: specifies the pathname of a savebox. The suffix "sv.mbx" is added if necessary. If this is used, the pathname specified becomes the new mail address of the entry.

STR -at FSystem {-via RelayN...-via Relay1}
: specifies an address on another computer system. STR identifies the user (or group of users) to receive the message and is not interpreted in any way by the local system. FSystem is the name of the foreign system where the address is located. If the optional -via control arguments are not present, FSystem must be one of the names of a foreign system in the local system's network information table (NIT). If, however, the -via control arguments are specified, the foreign system name does not need to be known to the local system.

If the -via control arguments are specified, they identify an explicit route to be used to reach the foreign system. In this case, Relay1 must be one of the names of a foreign system in the local system's NIT. Mail destined for this foreign address is forwarded to the system identified as Relay1. From there it is forwarded to the system identified as Relay2, etc. until it reaches the system identified as RelayN, where it is delivered to the system on which the foreign address actually resides. When the NIT is queried for either FSystem or Relay1, the query is performed in a case insensitive manner.

–user Person_id.Project_id
    specifies the given user's default mailbox under the specified project. This control argument is equivalent to:

```
-mailbox >udd>Project_id>Person_id>Person_id.mbx
```

    If this is used, the pathname indicated becomes the new mail address of the entry.

ACCESS REQUIRED

The user must have e access to the gate mail_table_priv_.

EXAMPLES

The command line

```
update_mail_table_entry "John Jones" -alias JJ
```

adds the name JJ to the entry John Jones.

The command line

```
update_mail_table_entry "John Jones" JJones.Sales
```

changes the mailing address asociated with John Jones to JJones.Sales.

Name: usage_and_revenue

SYNTAX AS A COMMAND

```
usage_and_revenue control data {old_data}
```

FUNCTION

prints out a report of system usage and revenue broken down by groups of users.

The following lines must be added to master.ec and biller.ec before this command can be invoked. To master.ec add:

```
file_output usage_and_revenue.report
usage_and_revenue usage_and_revenue.control today.use_totals
   yesterday.use_totals
console_output
```

To biller.ec add:

```
file_output monthly_usage_and_revenue.report
usage_and_revenue usage_and_revenue.control today.use_totals
console_output
```

ARGUMENTS

control
: is the pathname of an ASCII file that defines the groups of users for the report.

data
: is the pathname of a copy of the system use_totals month-to-date statistical data base.

old_data
: is the pathname of an earlier copy of the system use_totals month-to-date statistical data base.

NOTES

If both the data and old_data arguments are given, a daily report is produced showing the incremental system usage between the time old_data was created and the time data was created. If only the data argument is given, a monthly report is produced showing the month-to-date usage at the time data was created.

The control file may specify up to nine groups of users to be shown in the report. Each line of the control file specifies one group; each group consists of one or more of the usage bins from the use_totals data base. The format of a control file line is:

```
Group title:user1,user2,...,userN
```

Lines beginning with an asterisk (*) are ignored. The groups appear in the report in the same order in which they appear in the control file.

where:

group title
: is the title used to identify the group in the report. The maximum length is 24 (spaces are allowed, see "Example" below).

useri
: are the names of the usage bins in the use_totals data base that are to be included in this group. These are the names specified in the control file for reset_use_totals. If useri is enclosed in quotes, it specifies a group that is already defined in the control file and all the bins that go into that group also go into this group. (See "Example" below.)

EXAMPLES

If use_totals has bins labeled "Staff," "Users," and "Other," the control file:

```
Staff Use:Staff
Non-Staff Use:Users,Other
Total:"Staff Use","Non-Staff Use"
```

results in a report with three groups, one containing one usage bin (Staff), one containing two usage bins (Users+Other), and one containing three usage bins (Staff+Users+Other).

A facility is provided to add notes or messages to the report. If a segment named usage_and_revenue_footnote exists in the caller's working directory, its contents are printed at the end of the report.

Name: usage_total

SYNTAX AS A COMMAND

```
usage_total sat_path pdt_dir projfile use_totals
```

FUNCTION

scans all PDTs and places month-to-date system usage figures in a statistical data base.

ARGUMENTS

pdt_dir
: is the pathname of the directory in which the PDTs are located.

projfile
: is the pathname of the project file.

sat_pat
: is the pathname of the SAT.

use_totals
: is the pathname of the statistics data segment.

NOTES

This command is executed by the crank (in master.ec). The use_totals segment is today.use_totals, which is subsequently used as input to the system_daily_report program.

Name: vtoc_buffer_meters

SYNTAX AS A COMMAND

```
vtoc_buffer_meters {-control_arg}
```

FUNCTION

provides information about the utilization of volume table of contents (VTOC) buffers.

CONTROL ARGUMENTS

-report_reset, -rr
  generates a full report and then performs the reset operation.

-reset, -rs
  resets the metering interval for the invoking process so that the interval begins at the last call with -reset specified. If -reset has never been given in a process, it is equivalent to having been specified at system initialization time.

ACCESS RQUIRED

This command requires access to phcs_ or metering_gate_.

NOTES

If the vtoc_buffer_meters command is given with no control argument, it prints a full report.

The following are brief descriptions of the metering variables printed out by the vtoc_buffer_meters command.

The first section of the output (labeled "Routine") displays the number of calls to VTOC buffer management routines. In each of these, the data is presented as number of calls and as average time between calls (ATB). The routines included are the following:

get_vtoce
  called to read a VTOC entry, waiting for the I/O if the desired VTOCE entry is not already in a VTOC buffer.

put_vtoce
called to write a VTOC entry without waiting for the I/O to complete.

alloc_and_put_vtoce
called to obtain a free VTOC entry, read the VTOC entry, and write it back to disk.

free_vtoce
called to return a VTOC entry to the free pool after nulling it and writing the nulled entry to disk.

await_vtoce
called to await the completion of a VTOC write I/O generated by a previous call (e.g., to put_vtoce).

GET_BUFFERS
an internal routine to obtain a VTOC buffer for a specified VTOC entry. It first checks whether a buffer is already assigned to the VTOC entry (if so, this is considered a "Hit"). If a buffer is not assigned, the routine selects one for the VTOC entry.

WAIT
an internal procedure to wait for completion of all I/Os for a given VTOC buffer. "TC Waits" are the number of times it was necessary to relinquish the CPU to wait for I/O completion.

The next section of the output (labeled "Buffer Allocation") displays data on the performance of the buffer selection algorithm (the routine GET_BUFFERS).

Steps
is the number of times a buffer was examined by the selection algorithm.

Skips
is the number of times a buffer was skipped by the selection algorithm. This is refined by the reason for the skip, as follows:

os - an I/O was in progress for the buffer.

hot - the buffer contained a VTOC entry that could not be written to disk because of irrecoverable write errors.

wait - some process was waiting for completion of I/O activity to this buffer.

The last section of the output (labelled "Disk I/Os") lists the number of disk reads and writes to the VTOC.

EXAMPLES

The following is an example of the information printed when the vtoc_buffer_meters command is invoked with no control argument: (See Appendix A for a representation of the configuration deck used to create the system from which the metering sample was taken.)

```
Total metering time:          0:15:35

Routine                   # calls ATB(sec)

get_vtoce                   7367     0.13
put_vtoce                      0     0.00
alloc_and_put_vtoce          732     1.28
free_vtoce                   642     1.46
await_vtoce                  805     1.16
GET_BUFFERS                17187     0.05   9820 Hits
                                        ( 57.1% of calls)
WAIT                        6291     0.15   6291 TC Waits
                                        (100.0% of calls)

Buffer Allocation
                           # ATB(sec)

Steps                      10106     0.09
Skips                       2739     0.34  27.1% of steps
  os                        2516     0.37  91.9% of skips
  hot                          0     0.00   0.0% of skips
  wait                       223     4.19   8.1% of skips

Disk I/Os
                           # ATB(sec)

Reads                       5218     0.18
Writes                      4601     0.20
```

Name: vtoc_pathname

SYNTAX AS A COMMAND

```
vtoc_pathname volname vtocx {-control_arg}
```

```
vtoc_pathname pvtx vtocx {-control_arg}
```

FUNCTION

determines the pathname of a segment from the location of its VTOC entry (VTOCE). Specify the location of the VTOCE by giving its volume name (or physical volume table index, if known) and an index into the VTOC of that volume.

ARGUMENTS

volname
  is the physical volume name of the volume on which the VTOCE resides. This volume must be mounted and must be part of a mounted logical volume.

pvtx
  is the physical volume table index of the volume on which the VTOCE resides, if known. is the VTOC index of the VTOCE. You must give it in octal.

vtocx
  is the VTOC index of the VTOCE. You must give it in octal.

CONTROL ARGUMENTS

-brief, -bf
  suppresses the printing of an error message when the VTOCE is free.

ACCESS REQUIRED

You need access to the phcs_ gate, since it must copy directories.

NOTES

Your process must have status access to each of the containing directories of the segment. The command supplies "-NO-ACCESS-" as the entryname at the level at which further access is necessary, if needed; it gives "-NOT-LISTED-" as the entryname at that level if one of the containing directories specified in the VTOCE does not exist in its containing directory; and it provides "-????-" as the entryname at any level below that at which either of these problems occurs.

Name: vtocx_to_record

SYNTAX AS A COMMAND

```
vtocx_to_record vtoc_index {device_name}
```

FUNCTION

converts an octal VTOCE to index a Multics record number and sector offset.

ARGUMENTS

vtoc_index
    is the octal VTOCE index.

device_name
    is a valid device name (e.g., "m400", "m451").

Name: wire_pages

SYNTAX AS A COMMAND

```
wire_pages path {first_page n_pages} {-control_arg}
```

FUNCTION

wires all, or selected, pages of a segment into main memory. Such pages are not subject to removal by the page replacement algorithm. Wired pages remain in memory until shutdown, or until unwired (see the unwire_pages command).

ARGUMENTS

path
    is the pathname of the segment to be wired. Supervisor segments cannot be wired by this command.

first_page
    specifies the page number of the first page to be wired. The first page of a segment is page zero. If you give first_page, give n_pages. If first_page ends with a decimal point, it is treated as a decimal number; otherwise it is treated as an octal number. (Default: to wire all pages)

n_pages
    specifies the number of pages to be wired. If n_pages ends with a decimal point, it is treated as a decimal number; otherwise it is treated as an octal number.

CONTROL ARGUMENTS

-text
: to specify that path is an object segment and that only the text section should be wired. The default is to wire the whole segment.

ACCESS REQUIRED

Use of this command requires re access to the hphcs_ gate.

Name: work_class_meters, wcm

SYNTAX AS A COMMAND

```
wcm {-control_arg}
```

FUNCTION

prints certain information from the tc_data segment about each work class currently defined.

CONTROL ARGUMENTS

-report_reset, -rr
: generates a full report and then performs the reset operation.

-reset, -rs
: resets the metering interval for the invoking process so that the interval begins at the last call with -reset specified. If -reset has never been given in a process, it is equivalent to having been specified at system initialization time.

ACCESS REQUIRED

This command requires access to phcs_ or metering_gate_. Additionally, in order for the command to print the names of the work classes, access to both the Master Group Table (MGT) and the answer table is required. These tables are located in the directory >system_control_1.

NOTES

If the work_class_meters command is given with no control argument, it prints a full report.

When the scheduler is operating in percent mode, percentages are computed against two different base CPU quantities. It is necessary to understand the differences between these quantities in order to interpret the output of work_class_meters.

One base quantity is the total system CPU time. This is simply the total realtime all CPUs have been active doing anything (including running an idle process). In any interval of time when there was no reconfiguration of CPUs, the total system CPU time is the product of the length of the interval and the number of CPUs. Another base quantity is nonidle CPU time. This is the total CPU time expended by all CPUs except when running an idle process. It is given by the total system CPU time minus the sum of all idle time reported by total_time_meters (MP Idle, Non-MP Idle, and Zero Idle).

When the scheduler is operating in percent mode, it distributes CPU resources among contending work classes according to their guaranteed percentages. These percentages are percentages of total nonidle CPU time. So, if there are two work classes, each with a guarantee of 50 percent, and the system is 50 percent idle, each work class gets 25 percent of total system CPU time (assuming that there is enough demand for this to be possible). In this example, each work class is getting 50 percent of the nonidle CPU time, but only 25 percent of the total system CPU time. Another way of viewing this is that the guaranteed percentages define a relationship among work classes according to the ratio of percentages. That is, a work class with a guaranteed percentage of 10 percent gets about half as much CPU time as a work class with a guaranteed percentage of 20 percent, assuming sufficient demand by both. Further, this ratio is independent of the system load.

The system administrator can limit the CPU resources consumed by a work class to a fixed percentage of the total system CPU time. The scheduler enforces this limitation, even at the expense of going idle. That is, a work class with a maximum percentage of 10 percent gets no more than 10 percent of the total CPU time in any interval, regardless of load. Excess CPU time is distributed among work classes with no maximum percentage, according to their guaranteed percentages. If this cannot be done, the excess CPU time becomes idle time.

At any time one or more work classes may be a realtime work class with specified response time and quanta. A process in such a work class is low priority until its deadline arrives, at which time it is made eligible regardless of any other constraints. The remainder of the work classes are scheduled either by percentage of CPU time (percentage mode) or by soft deadlines (deadline mode).

The following parameters are always displayed for each work class.

WC
is the number of the work class.

%GUAR
is the percentage of nonidle CPU time guaranteed to the work class if the scheduler is being operated in percent mode and if there is sufficient demand by the work class for this to be possible.

%MAX
is the maximum percentage of total CPU time allowed by the system administrator to be consumed by this work class. This field is blank if the work class has no limitation on CPU consumption.

%TCPU
is the percentage of total CPU time actually received by this work class in the metering interval.

V/ELIG
is the average amount of CPU time used per eligibility quantum.

PW
is the pin weight, or number of free laps for pages brought into memory.

The following parameters are always displayed for realtime work classes, and are displayed for other work classes only if the scheduler is operating in deadline mode.

IRESP
is the response time (in seconds) specified for the work class after an interaction.

IQUANT
is the initial quantum (in seconds) for the work class after an interaction.

RESP
is the specified delay (in seconds) between subsequent quanta.

QUANT
is the value (in seconds) of subsequent quanta.

The following parameters are displayed when the scheduler is operating in either deadline or percentage mode.

P
if printed, members of the work class are post purged.

M
is the max_eligible limit per work class. A zero means the work class has no particular limit.

R2
if printed, the members of the work class are scheduled in realtime mode. They are made eligible at or before their deadlines.

I
if printed, members of the work class are given scheduler priority after interactions.

LCG
are the load control groups that are placed in the work class. If the LCG name is parenthesized, only the absentee processes in the LCG are placed in the work class.

EXAMPLES

The following is an example of the information printed when the work_class_meters command is invoked with no control argument. The scheduler is operating in percentage mode. (See Appendix A for a representation of the configuration deck used to create the system from which the metering sample was taken.)

```
Total metering time     0:15:35

WC %GUAR %MAX %TCP V/ELIG PW IRESP IQUANT RESP QUANT P M R I LCG

 0              2.  0.06   3  0.26  2.10  0.26  2.10 P O R I Init
 1              3.  0.07   1  0.25  0.75  0.50  1.00 P O R I RTime
 2   7.         6.  0.35   1                         P O   I System
                                                        SysAdm
                                                        OPR FED
 3  32.        45.  0.32   1                         P O   I SysProg
                                                        SysDev
 4   9.  14.   12.  0.43   1                         P O     SEngr
 5  20.         8.  0.20   1                         P O   I HEngr
 6  12.         5.  0.34   1                         P O   I MktUS
                                                        MktFor
                                                        MktEd
 7   3.   7.    5.  0.48   1                         P O     DS-CC
 8   6.         0.  0.10   1                         P O   I OffAuto
 9   4.   8.    1.  0.53   1                         P O     Misc Mfg
10   3.         3.  0.37   1                         P O   I Other
11   4.         1.  0.10   1                         P O   I Special

TCPU percents (%GUAR) control non-realtime work_classes.
```

Name: write_acct_bill

SYNTAX AS A COMMAND

```
write_acct_bill mm yy
```

FUNCTION

produces a bill for each external account number. Each bill contains one line for each Multics account.

ARGUMENTS

mm
: is a two-digit month designation.

yy
: are the last two digits of the year.

NOTES

The input to this program is the reqfile segment in the working directory.

The output is on the switches named bill and mailing_copy, which must have been previously attached.

This command is used by biller.ec to create the bill and mailing_copy segments. The two segments differ in that mailing_copy has a page containing the mailing address of the intended recipient of the bill.

The operation of this command depends on the proper attachment of the I/O switches (by biller.ec).

Name: write_billing_summary

SYNTAX AS A COMMAND

```
write_billing_summary mm yy
```

FUNCTION

this program is used to produce a one-line-per-account summary of the billing for external accounts.

ARGUMENTS

mm
: is a two-digit month.

yy
: is a two-digit year.

NOTES

Input to the program is the reqfile segment in the working directory.

Output is on the sumry I/O switch, which must have been previously attached.

This command is used by biller.ec to create the msum segment.

Name: write_notify_test

SYNTAX

```
write_notify_test {cpu_str} {-control_args}
```

FUNCTION

test the ability of the DPS 8 CPU and SCU interface to selectively clear CACHE when another active unit writes into main memory.

ARGUMENTS

cpu_str
: this is a character string of CPUs to use for testing. At least two CPUs are required, and one must be a DPS 8. The default is to select all the DPS 8 CPUs on the system. If only one DPS 8 CPU is found online, a L68 is used as the second processor. If there is only one CPU on the system or there are no DPS 8 CPUs, the test will not run.

CONTROL ARGUMENTS

-brief -bf
: displays the brief form of errors and meters. This control argument is the default.

-deactivation_count -dc N
: deactivates the data segment N times. This deactivation changes the location of the test page in main memory. The default is 1% of the total online pages.

-loop_count -lc N
: loops N times before deactivating the segment. The default is 10.

-long -lg
: displays the long form of errors and meters.

-meter
: displays the memory utilization percentages. If the -long control argument has been given, the actual pages and address lines are displayed.

-word_increment, -wi N
: uses N as the word increment for the test page. N is a decimal number greater than 0 and less than 1023. The default is 52.

-work_class
: runs the test in N work class. N is a decimal number from 0 to the highest work class number (16 max). The default is to run in the highest numbered realtime work class. Exiting from the test causes the work class to revert back to the original work class.

ACCESS REQUIRED

This command requires phcs_ and hphcs_ access.

NOTES

When invoked as an active function, write_notify_test returns the results of the test. If no errors are found, the return string is the word "passed". If the test detects errors, the first word of the return string is "failed". It is followed by the failing unit or units:

```
failed CPU_A
```

or

```
failed CPU_A SCU_B
```

This test assumes that a write notify problem is address line dependent rather than memory dependent. Not all addresses or memory addresses are checked. The defaults listed cause a random selection of address lines. The use of the -meter and -long control arguments display the address lines (0 through 13) used for the test page. The -word_increment control argument controls the address lines 14 through 23.

The test consists of three embedded loops. The outermost loop is controlled by the number of CPUs selected. The CPUs selected are divided into pairs and each pair is iterated through the next two sub-loops. For example: CPUs a, b, c, and d have been selected. The first pair is ab, the next is bc, and the last is cd. This aids in diagnostics.

The -deactivation_count control argument controls the second loop, the deactivation loop. This loop deactivates the test page in main memory, enabling the system to assign a new page address for testing.

The -loop_count control argument controls the inner most loop, the read_write loop. The first CPU in the pair writes the test page and the second CPU reads the test page. Next, the second CPU in the pair writes the test page and the first CPU reads the test page. Meters are displayed for each pair of CPUs.

Name: write_user_usage_report

SYNTAX AS A COMMAND

```
write_user_usage_report sat_path pdt_dir reqfile projfile miscfile
```

FUNCTION

produces one or more pages for each project, giving the usage for the month for all users.

ARGUMENTS

sat_path
: is the pathname of a copy of the SAT. This data base contains a list indicating the valid projects and the projects deleted during the month.

pdt_dir
: is the pathname of the directory where the copies of the PDTs are found. These segments contain per-user virtual CPU, memory units, and terminal usage for each project.

reqfile
: is the pathname of the reqfile segment. This segment contains per-account requisition information.

projfile
: is the pathname of the projfile segment. This segment contains per-project information.

miscfile
: is the pathname of the miscfile segment. This segment contains miscellaneous charges and detailed information for the month for each project.

NOTES

The command writes its output on four switches, named: long_bill, mailing_copy, short_bill, and both_bills. These switches must have been previously attached (all I/O attachments are the responsibility of the caller). Output intended for both the long and short versions of the bill is written on the both_bills switch. The long version of the bill is written on the long_bill switch. This version has from one to four sections per project to include:

```
summary of total charges (one line per user)
summary of interactive charges (one line per shift per user)
summary of interactive charges (one line per queue per user)
summary of I/O daemon charges (one line per queue per user)
```

The mailing copy version of the bill (written on the mailing_copy switch) includes all of the above, plus the following:

1. Page containing the mailing address of the intended recipient precedes the section for each project.

2. Current billing rates at the site are printed at the end of the total-charge summary.

3. Contents of the billing_footnote segment (if it exists) are printed in the total-charge summary, after the billing rates. The billing_footnote segment contains announcement material for all project supervisors.

The short version of the bill (written on the short_bill switch) consists of the per-project total-charge summary followed by a grand total page.

If a user entry appears in a PDT for which there is no reqfile or projfile entry, the command types out an error message and aborts.

If the calculated total dollar figure for an account (the sum of the user dollar figures in the PDT entries) does not match the total dollar figure kept in the reqfile, an error message is printed giving both totals, and the command continues using the total dollar figure from reqfile.

It should be noted that, since the charges for each user are computed and stored as floating-point binary numbers, some rounding occurs during the conversion of each user's usage figures to dollars and cents for printing. Thus, discrepancies occur between the sum of the printed user usage figures and the total usage figure for the account. The total figure is the one that is billed.

This command is used by biller.ec to create the long_bill, short_bill, and mailing_copy segments.

The operation of this command depends on the proper attachment of the I/O switches (by biller.ec).