

MCR10187

initiate and copy_dir Fail With “Allocation could not be performed”

Revision 1.0

Eric Swenson

October 10, 2026

1 Problem Description

[Ticket #358](#): `initiate -all` sometimes fails with

`initiate`: Allocation could not be performed. Unable to get names of >udd>m>gd>w>ect>token

`copy_dir` fails the same way, often, in long-lived processes.

1.1 Cause

As Gary Dixon explains in the ticket, `hcs.$status_` returns the entry’s names, and a link’s target pathname, as 18-bit *offsets* (`names_relp`, `pathname_relp`) from the base of the area the caller supplies. That works only if they are allocated in the segment that holds the area header. `status_` (in ring 0) checks for this after allocating:

```
if (return_names_ptr ^= null & baseno (return_names_ptr) ^= baseno
    (return_area_ptr))
    | (return_pathname_ptr ^= null
    & baseno (return_pathname_ptr) ^= baseno (return_area_ptr))
then call fatal_error (error_table_$notalloc);
```

and fails with `error_table.$notalloc` (“Allocation could not be performed”) after freeing what it allocated.

The callers pass the system free area (`get_system_free_area_`). That area is extensible: when its first segment has no free block large enough, the area package (`alloc_`, through `get_next_area_ptr_`) adds another segment in the process directory and allocates there, even when called from ring 0. So once a process has used enough of its system free area, every `hcs.$status_` call that asks for names can fail, depending on where the allocator happens to place the names. The failure is intermittent, grows more likely the longer a process runs, and disappears after `new_proc`.

`copy_dir` meets it far more often than `initiate` because it calls `hcs_$status_` for every entry it copies: `copy_dir_` calls `copy_names_` for each entry, which gets the entry's names with `hcs_$status_` and the system free area, and `copy_dir_`'s `link` procedure does the same for each link's names and target. A large copy also uses a lot of the system free area itself.

Calls that return full pointers, such as `hcs_$star_` and `hcs_$list_acl`, are not affected: they work in any segment of an extensible area.

2 Proposed Fix

At each affected call, `hcs_$status_` is given an automatic area of 2048 words, as the ticket suggests. An automatic area is not extensible, so the names are always in the segment that holds it (the stack). 2048 words hold the names of an entry with over 200 names. If the area is too small, `status_` returns `error_table_$noalloc`; the call is then repeated with the system free area, as before. So no case is worse than now.

```

areap = addr (status_area);          /* dcl status_area area (2048); */
...
call hcs_$status_ (dir1, en1, 0, eptr, areap, code);
if code = error_table_$noalloc | code = error_table_$notalloc
then do;
    areap = get_system_free_area_ ();
    call hcs_$status_ (dir1, en1, 0, eptr, areap, code);
end;

```

The existing `free` statements are kept; they free in whichever area was used. Freeing in the automatic area is harmless, and the area disappears when the procedure returns. A comment at each call explains why the automatic area is used.

Changed:

- `initiate.pl1` (`bound_ti_term_`): the `-all` path.
- `copy_names_.pl1` (`bound_fscom2_`): used by `copy_names`, `move_names`, `copy_dir`, `move_dir` and others.
- `copy_dir_.pl1` (`bound_fscom2_`): the `link` procedure, which uses a new local pointer (`status_area_ptr`) and area (`link_area`) for the names and the link pathname.

2.1 Not included

About 55 other programs call `hcs_$status_` or `hcs_$status_long` and also use the system free area, among them `status`, `list`, `delete_`, `nd_handler_`, `copy_`, `where` and `get_shortest_path_`. Those that pass the system free area to `hcs_$status_` have the same problem and can be fixed the same way; the list of candidates (not yet checked individually) is below:

```

bound_absentee_com_.s.archive/lar_util_.pl1
bound_archive_.s.archive/archive_aux_.pl1
bound_dm_firstref_.2.s.archive/fm_get_last_ci_num.pl1
bound_fs_util_.s.archive/fs_standard_object_.pl1.pmac
bound_fscom1_.s.archive/delete_.pl1

```

bound_fscom1_.s.archive/delete.pl1
bound_fscom1_.s.archive/list.pl1
bound_fscom2_.s.archive/check_file_system_damage.pl1
bound_fscom2_.s.archive/copy_.pl1
bound_fscom2_.s.archive/copy_dir_.pl1
bound_fscom2_.s.archive/copy_names_.pl1
bound_fscom2_.s.archive/nd_handler_.pl1
bound_fscom2_.s.archive/status.pl1
bound_full_cp_.s.archive/get_shortest_path_.pl1
bound_help_.s.archive/check_info_segs.pl1
bound_help_.s.archive/info_seg_.pl1
bound_info_rtns_.s.archive/print_motd.pl1
bound_info_rtns_.s.archive/where.pl1
bound_io_commands_.s.archive/enter_output_request.pl1
bound_io_commands_.s.archive/pool_manager_.pl1
bound_l6_satellite_.s.archive/l6_tran_.pl1
bound_l6_satellite_.s.archive/l6_tran_overseer_.pl1
bound_mrds_.4.s.archive/mu_temp_segments.pl1
bound_mseg_.s.archive/mseg_utils_.pl1
bound_tape_archive_.s.archive/ta_filesys_util_.pl1
bound_tape_archive_.s.archive/ta_replace_.pl1
bound_ti_term_.s.archive/initiate.pl1
execute_search_rules_.pl1
bound_admin_rtnes_.s.archive/set_tpp.pl1
bound_amu_.s.archive/amu_deadproc_.pl1
bound_amu_.s.archive/amu_fdump_mgr_.pl1
bound_amu_.s.archive/amu_get_name_.pl1
bound_carry_facility_.s.archive/carry_dump.pl1
bound_carry_facility_.s.archive/carry_load.pl1
bound_carry_facility_.s.archive/enter_carry_request.pl1
bound_dir_info_.s.archive/save_dir_info.pl1
bound_dn355_tools_.s.archive/manipulate_pathname_.pl1
bound_lib_utility_.s.archive/date_deleter.pl1
bound_lib_utility_.s.archive/get_primary_name_.pl1
bound_rcp_tools_.s.archive/reconstruct_registry.pl1
save_previous_system.pl1
bound_executive_mail_.s.archive/xmail_dir_manager_.pl1
bound_executive_mail_.s.archive/xmail_review_defers_.pl1
bound_forum_.s.archive/forum_add_meeting.pl1
bound_forum_.s.archive/forum_create.pl1
bound_forum_.s.archive/forum_find_v1.pl1
bound_imft_.s.archive/enter_imft_request.pl1
bound_imft_.s.archive/imft_cir_mir_commands_.pl1
bound_imft_.s.archive/imft_transmit_object_.pl1
bound_imft_.s.archive/list_imft_requests.pl1
bound_multics_emacs_.s.archive/list_emacs_ctls.pl1
bound_ted_support_.s.archive#1
bound_v2_forum_mgr_.s.archive/forum_conversion_.pl1
bound_v2_forum_mgr_.s.archive/forum_salvager_.pl1
bound_v2apl_.1.s.archive/apl_directory_commands_.pl1
bound_v2apl_.1.s.archive/apl_file_system_.pl1

They are left to a follow-up MCR so that this change stays small.

A change to `hcs_$status_` itself would not help: an 18-bit offset cannot point into another segment, and the area package gives the caller no way to require the first segment.

3 Documentation

No documentation changes.

4 Test

1. **Confirm the cause.** In a process that has produced the error, list the process directory (`list [pd]>**.area.**`): more than one area segment means the system free area has been extended. In that process `status -names` on an entry with several names should fail the same way some of the time; after `new_proc` it does not.
2. **Reproduce.** Extend the system free area by allocating most of a segment's worth of storage in it (a small test program that allocates and keeps, say, 250 blocks of 1024 words with `get_system_free_area_`), then run `initiate PATH -all`, `copy_names`, and `copy_dir` on a directory with segments, multi-named entries and links. With the old programs they fail with "Allocation could not be performed"; with the new ones they succeed, and the copies have all names and correct link targets.
3. **Many names.** `copy_names` and `initiate -all` on an entry with more than 250 names (falls back to the system free area) work as before in a new process.
4. **Normal use.** `copy_dir`, `move_dir`, `copy_names`, `move_names` and `initiate -all` in a new process give the same results as before.

5 Files Changed

<code>bound_ti_term_.s.archive</code>	<code>initiate.pl1</code>
<code>bound_fscom2_.s.archive</code>	<code>copy_names_.pl1</code> , <code>copy_dir_.pl1</code>
<code>bound_ti_term_.archive</code>	recompiled <code>initiate</code>
<code>bound_fscom2_.archive</code>	recompiled <code>copy_names_</code> , <code>copy_dir_</code>
<code>>sss</code>	<code>bound_ti_term_</code> , <code>bound_fscom2_</code> (rebound)

6 Version History

2026-10-10	1.0	Initial version of the MCR.
------------	-----	-----------------------------