

MCR10183

Reloader Cannot List Large Directories for Trimming

Revision 1.0

Eric Swenson

October 10, 2026

1 Problem Description

[Ticket #226](#): reloading the 12.7LDD_STANDARD tape with `reload.system.release` creates a reload error file containing:

```
hcs_$list_dir_fast: Allocation could not be performed. >library_dir_dir>include
backup_load_dir_list: Allocation could not be performed. >library_dir_dir>include
```

As Gary Dixon explains in the ticket, `reload.system.release` reloads with trimming (`-trim`): after the entries of a directory have been reloaded from tape, `backup_load_dir_list` calls `hcs_$list_dir_fast` to list everything now in the directory, and deletes the names, and entries, that did not come from the tape. When `hcs_$list_dir_fast` fails, trimming of that directory is skipped. So an include file deleted from a release is never deleted from `>ldd>include` at a site that installs the release with `reload.system.release`.

1.1 Cause

`backup_load_dir_list` passes `hcs_$list_dir_fast` an area in a segment in the process directory, created with the obsolete *buddy-system* area package:

```
backup_load_dir_list:
    procedure (hp1, code2) options (rename ((area_, old_area_), (alloc_, old_alloc_),
        (freen_, old_freen_)));
...
dcl listp ptr static initial (null),          /* Pointer to area for list_dir. */
    list_area (65536) based (listp);          /* Area for list_dir. */
...
    call old_area_ (65536, listp);            /* Clear area and get contents of system
        directory. */
    call hcs_$list_dir_fast (hp -> h.dname, listp -> list_area, sys_bp, sys_bc,
        sys_lp, sys_lc, code);
```

A buddy-system area rounds every block up to a power of two, and the area itself is the largest

power of two that fits (`buddy_area_`: “the actual size of this style area is the largest block which is a power of $2 \leq \text{size}$ ”). Since the area also holds its header, the largest block a 65536-word buddy area can hold is 32768 words.

`hcs_$list_dir_fast` (`star_$list_dir_fast`) allocates in that area one `dcpack_branch` (20 words) per branch, one `dcpack_ename` (10 words) per name, and the link arrays. For a directory of about 2500 segments, such as `>ldd>include`, the branch array alone is about 50,000 words, rounded up to a 65536-word block, which can never be allocated. Any directory of more than about 1600 branches fails the same way; `>ldd>include` is simply the largest directory on the release tapes.

The 64K limit and the buddy area date from 1972 (“Modified: 20 December 1972 by A. Downing to make calls to old area package”); `list_dir_fast` was adopted in 1976, and `>ldd>include` has since outgrown the area.

2 Proposed Fix

Give `hcs_$list_dir_fast` a standard area that fills the whole segment:

- The `rename` option is removed; it was there only to make the call to `area_` reach `old_area_`, and nothing else in the program uses `area_`, `alloc_` or `free_`. The declaration of `old_area_` is removed.
- When the segment `reload_area` is created (once per process), `hcs_$get_max_length_seg` gets its maximum length, saved in the new static `list_area_size`. If that fails, the error is returned as for a failure to create the segment, and the next call tries again.
- The area is declared with that size, both as the array of words that the gate declaration takes and as an area:

```

dcl listp ptr static initial (null),          /* Pointer to area for list_dir. */
    list_area_size fixed bin (19) static initial (0), /* Its size: the segment's max
        length. */
    list_area (list_area_size) fixed bin based (listp), /* Area for list_dir, as
        the gate declares it. */
    reload_area area (list_area_size) based (listp); /* The same, as an area. */

```

- Before each listing it is initialized with `reload_area = empty ()`; in place of `call old_area_ (65536, listp)`; . The segment is still truncated after each use.

With the default maximum segment length (255K words) the area holds the listing of a directory of over 8000 single-named entries, more than three times the size of `>ldd>include`. A comment at the change explains why the buddy area failed.

2.1 Not changed

- The ticket suggests a new hardcore entry returning only names. It is not needed: the problem is the size and format of the area, not the amount of information returned.
- The ticket also notes that the reload error file could not be read until the system was up, without forcing access and setting its bit count. That is a separate problem and is not addressed here.

3 Documentation

No documentation changes.

4 Test

1. **The listing.** A small test program calls `hcs.$list_dir_fast` on `>ldd>include`, first with a 65536-word area made by `old_area_`, then with a standard area filling a temporary segment. The first returns `error_table.$notalloc`; the second returns the full listing (branch count equal to `list -dir >ldd>include`).
2. **reload_system_release.** On a test system, add a segment `test.trim.incl.pl1` and an extra name on an existing include file in `>ldd>include`, then reload the LDD.STANDARD tape with `reload_system_release` and the new `bound_reloader_`. No error file entry is produced for `>ldd>include`; the test segment and the extra name are gone; all include files from the tape are present.
3. **Other directories.** The same reload of the other release tapes produces no new errors, and directories are trimmed as before.

5 Files Changed

<code>bound_reloader..s.archive</code>	<code>backup_load_dir_list.pl1</code> : standard area of the segment's size for <code>hcs.\$list_dir_fast</code>
<code>bound_reloader..archive</code>	<code>backup_load_dir_list</code> (recompiled)
<code>>s11</code>	<code>bound_reloader_</code> (rebound). It is on the hard-core list, so it is installed with a new system tape.

6 Version History

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