Subject: MCR10164, Correct problems in compare_entry_names Tool

Author: Gary Dixon

Date: 2026-07-11

Multics Ticket: https://multics-trac.swenson.org/multics-trac/ticket/392

Ticket 392 describes the following problems in the compare_entry_names tool.

PROBLEM 1: The compare_entry_names tool is documented to compare the names on two file system entries.

```
help compare_entry_names -all

>doc>info>compare_entry_names.info   (14 lines in info)

1986-01-28  compare_entry_names, cen

Syntax as a command:  cen path1 path2

Function: compares the names on two entries in the storage system and
prints information about the differences.

Arguments:
path1
   is the pathname of the first entry.  You can't use the star
   convention.
path2
   is the pathname of the second entry.  You can use the equal
   convention.
```

However, because it chases links when locating the two "entries", it fails to compare names on a branch with names on a link to that branch that is located in a different directory. Instead, it compares unique IDs of the two entries and complains they are the same segment.

The function it is actually performing should be called compare_branch_names.

To correct this problem, the following changes are needed.

A. Calls to hcs_$status_long should pass a NO_CHASE argument value.

B. The code should check if either entry is a Link before comparing unique IDs of the two branches. When a link is involved, the tool must skip unique ID check because Link entries do not have a unique ID.

PROBLEM 2: To avoid a well-known flaw in the design of the hcs_$status_ and hcs_$status_long interfaces, the names on the target entry may not be allocated in an extensible area. If such areas locate the name array in one of the extension components, then the returned names are not accessible.

Both hcs_$status_ and $status_long interface base the name array on a pointer to the first component of the area plus an offset to the name array location within that area component. Their interface has no way to return a pointer to the actual area component in which the name array gets allocated.

From status_structures.incl.pl1:

```
dcl 1 status_branch aligned based (status_ptr),
    2 short aligned,
      3 type fixed bin (2) unaligned unsigned,      /* seg, dir, or link   */
      3 nnames fixed bin (16) unaligned unsigned,   /* number of names     */
      3 names_relp bit (18) unaligned,              /* see entry_names dcl */
         ...

dcl  status_entry_names (status_branch.nnames) character (32) aligned
          based (pointer (status_area_ptr, status_branch.names_relp));
```

compare_entry_names uses the area provided by get_system_free_area_ which is an extensible area. Therefore, it is susceptible to this problem.

The recommended way to avoid this defect is to allocate the name arrays in a non-extensible area large enough to hold these allocations. An automatic area of 1700 words is large enough to hold two 100-element name arrays, and the area would occupy only 1.7 pages of storage in the stack segment.

PROBLEM 3: compare_entry_names tries to rejoin the directory and entryname portions of a pathname back together by just separating the two names with a greater-than (>) character. For example:

```
call ioa_ ("^/Names unique to ^a>^a", dname1, ename1);
```

This method fails for entries in the root (>) directory. It should be joining the two portions using the pathname_ subroutine which properly handles joining such pathname components.

Proposed Changes

Replace the current compare_entry_names source with a simpler source which applies the error remedies suggested above.

```
Changes --

 Bound_obj:    bound_lib_utility_                IN: tools  UPDATE;
   source:     compare_entry_names.pl1            REPLACE  compiler: pl1 -ot;
```

However, the solution proposed for PROBLEM 1 represents an incompatible change to the user interface for compare_entry_names. Instead of always chasing links when getting status for the two entries named by path1 and path2 arguments, links would never be chased.

To avoid this incompatible change, I propose to add -chase and -no_chase control arguments to the command, allowing the user to choose how links will be handled. And the default operation for the command would change in a compatible fashion by supporting a third chase setting: -chase_if_possible, -cip.

These new control arguments are described in the revised info segment shown below.

```
2026-07-12  compare_entry_names, cen

Syntax as a command:  cen PATH1 PATH2 {-control_args}

Function: compare names on two entries in the storage system and print
names which appear on only one of the entries.

Arguments:
PATH1
   gives the pathname of the first entry.  Neither the star convention
   nor the archive pathname convention are supported.
PATH2
   gives the pathname of the second entry.  The equal convention may be
   used to incorporate name components from PATH1 in the second
   pathname.

Control arguments:
-chase
   If either PATH argument is a link, follow that link to reach its
   branch target and compare names on that target with those on the
   other PATH argument.
-no_chase
   If either PATH argument is a link, compare names on that link with
   those on the other PATH argument.
-chase_if_possible, -cip
   If either PATH argument is a link, follow that link to reach its
   branch target.  If both chased paths identify the same branch
   target, then compare names on the original PATH entries without
   chasing links. (default)
```

This change involves significant enhancements to the compare_entry_names code, so no code changes will be included in this MCR. Code review will be left to the developer and his auditor.

Testing

The testing of these revisions focused on:

- Correct operation of the name comparison code.
    - Ability to compare names on link entries with names on target segment or directory entries. Compare names on >sss with names on a link to >sss.
- Ability to handle entries having 100 or more entry names.
    - Check that a proper error message is displayed when asked to handle entries with 150 names (a highly unusual entry). Create segments A and B, and add 100 names to each; then check output of new cen command. Then add 50 more names to each test segment, and check for error_table_$no_alloc being returned by hcs_$status_long and displayed.
- Proper handling of pathnames located in the Root directory (>).
    - Run the command to check entry names on >ldd and >sss, ensuring both pathnames are displayed properly.

Documentation

The revised documentation for compare_entry_names was shown earlier in this MCR.

Version History

INSTRUCTIONS: End with a version table including details shown below.

Date	Revision	Author	Comment
2026-07-11	1.0	Gary Dixon	Initial draft of this MCR.
2026-07-12	1.1	Gary Dixon	Add -chase, -no_chase and -chase_if_possible control arguments.