MCR10211
mrds_call retrieve Prints Uninitialized Values
Revision 1.0

Eric Swenson

November 22, 2026

1 Problem Description

Ticket #198 describes an issue where the `mrds_call retrieve` request can print garbage (NUL characters, a database pathname, "exclusive_update") before the retrieved values. In the reported example the caller asked for 6 values (`mrds_call retrieve 6 1 "-range (e Employee) -select e" -all`) from a relation with 3 attributes.

1.1 Cause

`mrds_call` allocates the array that receives the values, `values (n_vals) char (256)`, in its work area and passes its elements to `dsl_$retrieve`; it never initializes the array.

`dsl_$retrieve` (`mrds_dsl_set_user_vals`) takes the values for the selected items from the *end* of its argument list. If there are fewer value arguments than selected items it fails with `mrds_error_$inval_rtrv_expr`; if there are more, it sets the last ones and ignores the extra ones in front, without an error. In the ticket's example (6 values, 3 selected attributes) values 1–3 are never set and keep whatever was in the work area (here, earlier data such as the database pathname and opening mode); values 4–6 receive `name1`, `123` and `comp1`. `mrds_call` prints all six.

2 Proposed Fix

`mrds_call` reports an `nvals` larger than the number of selected items as an error instead of printing values that were never set. It needs no information from MRDS to do so: before the first retrieve it fills every value with a marker that no retrieved value can be (256 copies of `byte (511)`); `dsl_$retrieve` converts each selected value to `char (256)`, so every value it sets is overwritten. After a successful retrieve, the leading values that still hold the marker are the extra ones:

```
unset_value = copy (byte (511), length (unset_value));
values (*) = unset_value;
...
call cu_$generate_call (mrds_dsl_retrieve$retrieve, al_ptr);
...
n_unset = 0;          /* count the leading values not set */
```

```
do i = 1 to n_vals while (values (i) = unset_value);
     n_unset = i;
end;
if n_unset > 0 & n_unset < n_vals then do;
        call retr_cleanup;
        call com_err_ (error_table_$wrong_no_of_args, MRC,
             "^/nvals is ^d, but the selection expression selects ^d
                 value^[s^].",
             n_vals, n_vals - n_unset, (n_vals - n_unset ^= 1));
        go to Exit;
     end;
```

For the ticket's example the output is now:

```
mrds_call: Wrong number of arguments supplied.
nvals is 6, but the selection expression selects 3 values.
```

and no values are printed. The check is made only after the first retrieve; the `-another` retrievals done for `-all` use the same arguments. Requests that give the right number of values are unaffected.

`dsl_$retrieve` itself is not changed: making it reject extra value arguments would change a subroutine interface that existing programs may rely on. An `nvals` smaller than the number of selected items still gets MRDS's "Invalid selection expression for retrieve."; telling that apart from a really invalid expression would need the expression to be parsed in `mrds_call`.

3 Documentation

No documentation changes.

4 Test

1. The ticket's example (`mrds_call retrieve 6 1 "-range (e Employee) -select e" -all`): the error above, no values printed.
2. The same with `nvals` 4: "`selects 3 values`".
3. `mrds_call retrieve 3 1 "..." -all`: unchanged output (`name1`, `123`, `comp1` and the other tuples).
4. A selection of one attribute with `nvals` 2: "`selects 1 value`".
5. `nvals` 2 for the 3-attribute selection: MRDS's "Invalid selection expression for retrieve.", as before.
6. `set_modes no_retrieve_output` with the ticket's example: the error, no tuple count; with `nvals` 3: the tuple count as before.

5 Files Changed

bound_mrds_.1.s.archive	mrds_call.pl1
>sss	bound_mrds_ (rebound)

6 Version History

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