MCR10203
ISOLTS Fixes: error log display, page loading, runaway pages and operator queries
Revision 1.0

Eric Swenson

November 3, 2026

1 Problem Description

Tickets https://multics-trac.swenson.org/ticket/434, https://multics-trac.swenson.org/ticket/435, https://multics-trac.swenson.org/ticket/436, and https://multics-trac.swenson.org/ticket/437 describe issues and an enhancement request for ISOLTS.

ISOLTS (`isolts_`) runs the processor and store test pages (PAS) on a processor taken from the system, records every error report a page sends in `isolts_err_log`, and displays or prints it with the `display_error` request. This MCR fixes four groups of problems.

1.1 A. Error log display

1. The error just reported is not shown. When a page reports an error and then halts at `*options?`, `display_error` shows only the errors before it (with a new log, nothing). The error appears only after the next error, the next console message, or the next page. After a report with register or memory dumps, `display_error` with `print` queues an empty `.ilog`.
2. A `display_error` while a report is still being written (after a BREAK) leaves a garbage entry in the log (seen as sequence numbers 4307574836 and 15461314157, dated 00/00/00), and the entries written after it are threaded wrongly.
3. `display_error -all` does not show the newest entry, and `display_error N` (a count without a limit) shows only $N - 1$ entries, so `display_error 1` shows none.

PAS sends an error report as a series of print requests, the first marked as the start of an error, and never marks its end. `interpret_action` writes each piece to the log with `isolts_err_log_$write` (type 1, then type 2), and `complete_err_mess` finishes the entry (type 3, which also makes it the log's last entry) only at the next console write, program load, or new error. An options request (`enter_options`) did not finish it, and the path that writes a report's dump file prompts for options directly.

The display walks the log with the internal static pointers `mep` and `imep`, the ones the writer uses for the entry in progress, so a display during a report makes the writer finish and thread the wrong

entry.

For problem 3: `isolts_err_log_$display` walks back from the newest entry, counting, and then prints forwards from the entry it reached, with `j` running from `i-2` down to `lim`. With no limit (`-all`, or a count alone) it set `lim = 1`, which prints one entry fewer than it walked back over; since the entries are printed oldest first, the newest is the one left out. Plain `display_error` (count 1, limit 1) computes `lim = 0` and is right.

1.2 B. PA870 repeats its error report for ever

On an L68 test processor, PA870 test 01E reported an error and then printed the report again and again for more than an hour without returning to `*options?`. Each report ended in "UNEXPECTED STORE FAULT AT 045502", in the executive's CU history register printout.

The printout (pas.061 at absolute 045500) indexes the history table with the upper half of word 045545 (decimal 19301), adding 2 per entry until the word is 128 in the upper half and 0 in the lower half, and then stores 0 back. The deck in the T&D deck file does not load word 19301: it is zero in the program, and the deck leaves it out (`ORG 19302` after word 19300). The tape copy of the same executive (pm061) loads it as 0. So the deck relies on memory being zero.

`isolts_` clears its 64K work segment once, when the test processor is set up, before the primitive function tests (PFTs). The PFTs fill unused memory with `DIS` traps, each addressing itself, and the executive is loaded over that. Word 045545 was left as `045545616200` (`DIS 045545`). With a nonzero lower half the index never reaches its end value, so the printout ran on until it addressed 0200000, the 64K limit of the test processor's memory, and took a store fault. The fault's report printed the CU history again, and faulted again.

1.3 C. A runaway page is never stopped

ISOLTS gives up on a page when it makes no request for 600 seconds (1200 for pages 892, 893 and 955), but every request restarts that wait. A page that loops while printing, like PA870 above, sends a request every few seconds and is never stopped; the operator sees nothing (an error report goes only to the log) and has to notice and BREAK.

1.4 D. Operator queries on emulated processors

`test cpu` first asks the operator for permission to use the processor and then for a manual reconfiguration (switch settings, the "cpu test" free-edge connector on an L68). On the DPS8M emulator the configuration is set in the emulator, and the questions only stop the run until someone answers them on the console. Automated ISOLTS runs (the emulator's CI tests) edit `isolts_.pl1` to remove the call.

2 Proposed Fix

2.1 isolts_

1. (A) `enter_options` calls `complete_err_mess` first, before the dump and forced-`seq` paths. PAS asks for options only after it has sent the whole report. With no report in progress, a display cannot disturb one.

2. (B) Before each program is loaded (also on a retry after a timeout), the area it loads into is cleared: `wseg.exec` (10000–77777) for the executive (pas.061), `wseg.test_pgm` (100000–177777) for a slave-mode page. Master-mode PFTs are loaded as before. Every word a deck leaves out is then zero, as on a freshly cleared processor.

3. (C) New `test cpu` control argument `-page_time_limit N` (`-ptl N`): a page that has run N minutes ($1 \leq N \leq 100000$) since it was loaded or since the operator last answered an options request is stopped as a BREAK stops it: PAS halts and asks for options, and ISOLTS prints "isolts: pas2 test NNN has run N minutes without a console read; stopping it." Continuing gives the page another N minutes. There is no limit by default: PS955 and others legitimately run for hours. The clock is restarted both when `*options?` is typed and when the answer has been read, so time spent at the prompt does not count: the answer that loads the next page (e.g. `tst955`) still runs under the name of the old page until the new one is loaded, and would otherwise be stopped at once if the operator had taken longer than N minutes to answer.

4. (C) New `test cpu` control argument `-repeat_limit N` (`-rpl N`), default 5, 0 for no check: when a page prints the same error report more than N times in a row with no console read, it is stopped the same way, with "isolts: pas2 test NNN printed the same error report N times without a console read; stopping it." A report is identified by its first non-blank line. A repeat is a new report with the same first line, or that line printed again within the same report (PA870 did both: first one report that never ended, then, after a BREAK, a new report every 13 seconds). The check is off while the operator has set the PAS `loop` option (tracked from the option text, like `run`; `noloop` and `reset` clear it), since looping on an error is then what was asked for. A normal error halts at `*options?`, which resets the count, so a page is stopped only when it repeats without operator input.

5. (D) New `test cpu` control argument `-no_operator_query` (`-noq`): `opr_com` is not called; ISOLTS prints "operator queries skipped" and goes on to the reconfiguration. For emulated processors, where the configuration is already set. Without it, nothing changes.

The new control arguments are reset for each `test cpu` request, like `-trace`.

The `compare_ascii` comparison of the current and proposed `isolts_.pl1` appears below:

```
cpa [lpn isolts_.pl1] ==
Inserted in B:
B99      dcl  no_opr bit (1) init ("0"b);                    /* -no_operator_query: skip
    opr_com */
B100     dcl  page_limit fixed bin (71) init (0);            /* -page_time_limit,
    microseconds; 0 = none */
B101     dcl  page_start fixed bin (71) init (0);            /* page loaded or last
    console read */
B102     dcl  repeat_limit fixed bin init (5);               /* -repeat_limit; 0 = no
    check */
B103     dcl  err_repeat fixed bin init (0);                 /* times err_line was
    reported in a row */
B104     dcl  err_line char (136) varying init ("");         /* first line of the last
    error report */
B105     dcl  out_line char (136) varying;                   /* a line of the current
    error report */
B106     dcl  head_pending bit (1) init ("0"b);       /* next non-blank line is a report
    head */
```

```
B107     dcl  loop_opt bit (1) init ("0"b);                    /* PAS loop option entered
    */
B108     dcl  n fixed bin (35);
Preceding:
A99
A100     /* Constants */


A122     dcl  (addr, addrel, fixed, hbound, index, length, ltrim, mod, null, ptr, rel,
A123          rtrim, search, string, substr, time) builtin;
Changed by B to:
B132     dcl  (addr, addrel, clock, divide, fixed, hbound, index, length, ltrim, mod, null,
    ptr, rel,
B133          rtrim, search, string, substr, time, verify) builtin;


Inserted in B:
B288                no_opr = "0"b;                             /* and the per-test control
    args */
B289                page_limit = 0;
B290                repeat_limit = 5;
Preceding:
A278                if cmd_cnt < 3 then                        /* not enough args */


Inserted in B:
B308                        else if args (i) = "-no_operator_query"
B309                         | args (i) = "-noq" then no_opr = "1"b; /* don't ask the
    operator (opr_com) */
B310                        else if args (i) = "-page_time_limit"
B311                         | args (i) = "-ptl" then do;      /* minutes a page may run
    without a console read */
B312                           if i = cmd_cnt then go to bad_rsp;
B313                           i = i + 1;
B314                           n = cv_dec_check_ ((args (i)), code);
B315                           if code ^= 0 | n < 1 | n > 100000 then go to bad_rsp;
B316                           page_limit = n * 60000000;
B317                        end;
B318                        else if args (i) = "-repeat_limit"
B319                         | args (i) = "-rpl" then do;      /* same error this many
    times in a row stops the page */
B320                           if i = cmd_cnt then go to bad_rsp;
B321                           i = i + 1;
B322                           n = cv_dec_check_ ((args (i)), code);
B323                           if code ^= 0 | n < 0 | n > 100000 then go to bad_rsp;
B324                           repeat_limit = n;
B325                        end;
Preceding:
A295                        else go to bad_rsp;               /* tell user about his
    typing problem */
```

```
A339               if opr_com (cpu_tag, scu_tag) then         /* if permission denied */
Changed by B to:
B370               if no_opr then                             /* -no_operator_query: the
    operator */
B371                  call ioa_ ("operator queries skipped"); /* doesn't need to do
    anything (emulator) */
B372               else if opr_com (cpu_tag, scu_tag) then    /* if permission denied */


A382          dump_in_prog, idump = "0"b;
Changed by B to:
B415          dump_in_prog, idump, loop_opt = "0"b;
B416          err_repeat = 0;
B417          err_line = "";


A424     tout_retry: call gload_ (t_ptr, addrel (wseg_p, pgm_offset), 0, addr
    (gload_data), code);
Changed by B to:
B459     /* Clear the area the program loads into. A deck leaves out words that the
B460        program expects to be zero (pas.061 leaves out 19301, the index of its
B461        CU history printout, which the tape's pm061 loads as 0), and the work
B462        segment is cleared only once, before the PFTs, which fill memory with
B463        DIS traps. Left as DIS 19301, that index never reached its end value, so
B464        the printout ran up to the 64K limit and took a store fault, whose report
B465        printed the history again: PA870 01E repeated its report for ever. */
B466
B467     tout_retry:
B468             if nxt_tst = pas_exec then                   /* the pas2 exec, 10000 -
    77777 */
B469                wseg.exec = "0"b;
B470             else if pgm_offset = fixed (rel (addr (wseg.test_pgm)), 18) then
B471                wseg.test_pgm = "0"b;                     /* a slave program, 100000
    - 177777 */
B472             call gload_ (t_ptr, addrel (wseg_p, pgm_offset), 0, addr (gload_data),
    code);


Inserted in B:
B492             page_start = clock ();                       /* start the
    -page_time_limit clock */
B493             err_repeat = 0;                              /* and the -repeat_limit
    count */
B494             err_line = "";
Preceding:
A444     continue_pas:                                        /* target of non local go
    tos */


Inserted in B:
B553             if page_limit > 0 & pas_sw & nxt_tst ^= pas_exec & ^dump_in_prog &
    ^action.read then
```

```
B554                  if clock () - page_start > page_limit then do;
B555                                                          /* the timeout above is
    reset by every request */
B556                     call ioa_ ("^/^a: pas2 test ^a has run ^d minutes without a
    console read; stopping it.",
B557                       pname, nxt_tst, divide (page_limit, 60000000, 17));
B558                     call halt_page;
B559                  end;
Preceding:
A502               call interpret_action;                     /* do what the test wants */


Inserted in B:
B641               call check_repeat;                         /* stop a page that repeats
    its report */
Preceding:
A583               call isolts_err_log_$write (mptr, mlen, mtype, cpu_tag, scu_tag);


Inserted in B:
B667          page_start = clock ();                          /* a console read restarts
    the */
B668          err_repeat = 0;                                 /* -page_time_limit clock
    and the */
B669          err_line = "";                                  /* -repeat_limit count */
B670          call complete_err_mess;                         /* PAS has finished any
    error report before it
B671                                                             asks for options; without
    this an error
B672                                                             followed by a halt was
    not in the log for
B673                                                             display_error, and a
    display during the
B674                                                             report garbled the entry.
    Done before the
B675                                                             dump case, which goes
    straight to opt_mess */
Preceding:
A608          if dump_in_prog then do;                        /* dumping and we get here,
    means we are ready to dump */


Inserted in B:
B739               if index (com_string, "noloop") ^= 0 then loop_opt = "0"b; /* loop on
    the subtest */
B740               else if index (com_string, "loop") ^= 0 then loop_opt = "1"b; /* (no
    -repeat_limit check) */
Preceding:
A671               if index (com_string, "reset") ^= 0 then do; /* if reset option */


Inserted in B:
```

```
B744                  loop_opt = "0"b;
Preceding:
A674               end;


Inserted in B:
B754          page_start = clock ();                          /* restart the
    -page_time_limit clock after
B755                                                             the read too: time spent
    at the prompt is
B756                                                             not page run time, and
    the next request
B757                                                             (e.g. loading the tstNNN
    page) still runs
B758                                                             under the old page's name
    */
Preceding:
A683          go to cont_pas;                                 /* perform non_local goto */


Inserted in B:
B828     /* check_repeat - internal procedure to stop a page that prints the same error
B829        report again and again without a console read (-repeat_limit). The
B830        head of a report is its first non-blank line. A repeat is a new report
B831        with the same head, or the head printed again inside the same report:
B832        PA870 01E (dps8m, 2026-11-02) did both, first as one message that never
B833        ended, then (after a BREAK) as a new error every 13 seconds. Not done
B834        when the operator asked for the PAS loop option. */
B835
B836     check_repeat: proc;
B837
B838          if repeat_limit = 0 | loop_opt then return;
B839          if mtype = 1 then head_pending = "1"b;          /* new report: look for its
    head */
B840          bcd_chars = mlen * 6;
B841          call tolts_util_$bci_to_ascii (bcd_str, out_line, bcd_chars);
B842          if verify (out_line, " " || NL) = 0 then return; /* blank line */
B843          if head_pending then do;                        /* head of this report */
B844            head_pending = "0"b;
B845            if out_line = err_line then err_repeat = err_repeat + 1;
B846            else do;
B847               err_line = out_line;
B848               err_repeat = 1;
B849            end;
B850          end;
B851          else if out_line = err_line then                /* head again in the same
    report */
B852            err_repeat = err_repeat + 1;
B853          else return;
B854          if err_repeat > repeat_limit then do;
B855            call complete_err_mess;
B856            call ioa_ ("^/^a: pas2 test ^a printed the same error report ^d times
```

```
    without a console read; stopping it.",
B857                pname, nxt_tst, err_repeat);
B858               call halt_page;
B859             end;
B860
B861          end check_repeat;
B862
B863       /* halt_page - internal procedure to stop the running page as a BREAK does
B864          (-page_time_limit, -repeat_limit): PAS halts and asks for options */
B865
B866       halt_page: proc;
B867
B868             page_start = clock ();                          /* a full period again
    after continuing */
B869             err_repeat = 0;
B870             err_line = "";
B871             head_pending = "0"b;
B872             ntype = "0"b;                                   /* force options type out */
B873             string (action_codes) = "0"b;                   /* drop the request we were
    given */
B874             action_codes.halt = "1"b;                       /* set halt imediate flag */
B875             go to cont_pas;                                 /* perform non local go to
    */
B876
B877          end halt_page;
B878
Preceding:
A752       /* set_survey - internal procdure to set up system survey table */

Comparison finished: 15 differences, 137 lines.
```

2.2 isolts_err_log_

1. (A) `display` saves `mep` and `imep` on entry and restores them on return and in its cleanup handler (which is now established before the print file is attached).
2. (A) With no limit, `lim = 0`, so all the entries walked back over are printed: `-all` shows every entry, `display_error N` the N newest, `display_error N L` (unchanged) L entries starting N back, and `display_error` (unchanged) the newest.

A log already damaged by an earlier display must be deleted (ISOLTS makes a new one).

The `compare_ascii` comparison of the current and proposed `isolts_err_log_.pl1` appears below:

```
cpa [lpn isolts_err_log_.pl1] ==

Inserted in B:
B64        dcl (save_mep, save_imep) ptr;                       /* writer's pointers, kept
    across display */
Preceding:
A64        dcl (term, first_printed) bit (1);
```

```
A302              s_att, s_open = "0"b;                    /* reset flags */
Changed by B to:
B303              save_mep = mep;                          /* display walks the log
    with mep and */
B304              save_imep = imep;                        /* imep, which are the
    writer's static */
B305                                                       /* pointers; a display
    while a message is */
B306                                                       /* being written (after a
    BREAK) used to */
B307                                                       /* leave them moved, so the
    writer then */
B308                                                       /* completed and threaded
    the wrong entry */
B309              s_att, s_open = "0"b;                    /* reset flags */
B310              on cleanup begin;                        /* establish a cleanup
    handler */
B311                   call clean_up;
B312                   mep = save_mep;
B313                   imep = save_imep;
B314              end;


A306              on cleanup call clean_up;                /* establish a cleanup
    handler */
Deleted by B, preceding:
B318              if cnt = 0 then                          /* set defaults */


A311              if lim = 0 | lim = -1 then
A312                   lim = 1;
Changed by B to:
B322              if lim = 0 | lim = -1 then               /* no limit: all cnt
    entries; lim = 1 */
B323                   lim = 0;                            /* dropped the newest
    (-all, count only) */


Inserted in B:
B373              mep = save_mep;                          /* give the writer its
    pointers back */
B374              imep = save_imep;
Preceding:
A362              return;

Comparison finished: 5 differences, 21 lines.
```

3 Documentation

ISOLTS has no info segment. The three new `test cpu` control arguments should be added where the ISOLTS requests are documented:

```
test cpu TAG {-memory TAG} {-brief} {-trace} {-no_operator_query}
           {-page_time_limit N} {-repeat_limit N}

-no_operator_query, -noq
   don't ask the operator for permission to test the processor or
   to reconfigure it (for emulated processors, whose configuration is
   already set).
-page_time_limit N, -ptl N
   stop a test page (as a BREAK does) when it has run N minutes since
   it was loaded or since the last options request was answered.
   The default is no limit.
-repeat_limit N, -rpl N
   stop a test page when it prints the same error report more than N
   times in a row with no options request. The default is 5; 0 turns
   the check off. Not done while the PAS loop option is set.
```

4 Test

On a test system (DPS8M emulator) with a private copy of `bound_tolts_`, testing CPU B, starting with a new error log. Step 1 has been run (2026-11-01) and passed with an earlier version of the `enter_options` change. With revision 1.1 (2026-11-02), step 5 (PA870 reported once) and step 8 (`-noq`) passed, and `-ptl` stopped PS955 as in the first part of step 7. The rest, and the second part of step 7 (revision 1.2), are still to be run.

1. Run the processor test sequence (`tst700`, `seq`). When PA887 reports its error (test 09A) and halts, `display_error` at the first `*options?` shows the PA887 test 09A report. Before the fix it showed nothing; with an older log it showed only earlier errors. The same after a report with dumps (on an L68: PM776 06A or PM842 16A with an earlier emulator), and `display_error` with `print` queues a non-empty `.ilog`.

2. Continue (new line) until a further error (e.g. PM893), then `display_error -all`: every error is listed once, in order, with no garbage entry.

3. BREAK during an error report, `display_error`, continue, let the page report another error, then `display_error -all`: no garbage entry, all reports intact.

4. With several entries in the log: `display_error -all` shows all of them, newest included; `display_error 2` the two newest; `display_error 1` the newest; `display_error 3 1` the third newest. `display_error` with no error since the last display, and `display_error -all` with an empty log ("Error log file is empty"): unchanged.

5. (B) On an L68 test processor, run PA870 (`tst870`): test 01E reports its error (an accepted CU-overlap difference on the emulator), including the CU history registers, and halts at `*options?`. Before the fix the report repeated for ever. The rest of the sequence runs as before.

6. (C) With the fix of step 5 removed (or with a page that loops), the page is stopped after the

sixth identical report with the `-repeat_limit` message, at `*options?`. With `-rpl 0`, it is not stopped. After `loop` at `*options?` on a failing subtest, the page loops until BREAK (no message); after `noloop` the check applies again.

7. (C) `test cpu b -ptl 1`, then `tst955`: PS955 is stopped about a minute after it starts, at `*options?`; continuing (new line) runs it for another minute. Without `-ptl`, PS955 runs until data switch bit 1 is set, as before. Then, with `-ptl 2`, wait more than two minutes at `*options?` before typing `tst955`: PS955 is loaded and runs for two minutes (before the fix in revision 1.2 the old page was stopped at once, "pas2 test 700 has run 2 minutes without a console read").

8. (D) `test cpu b -noq`: no operator messages or queries, "operator queries skipped", and the test environment is set up. Without `-noq` the queries are asked as before, and `deny` still refuses the test.

5 Files Changed

`bound_tolts_.s.archive`	`isolts_.pl1`
`bound_tolts_.s.archive`	`isolts_err_log_.pl1`

6 Version History

2026-11-01	1.0	Initial version of the MCR (error log display).