Subject: MCR10176, Fix bugs in message_facility_.pl1

Author: Jim Lippard

Date: September 21, 2026

Multics Ticket: http://multics-trac.swenson.org/ticket/410

1. Problem Description

There are a number of bugs in message_facility_ which can cause processes to hang or incorrect looping behavior. This MCR proposes fixing those significant bugs as well as other more minor bugs which could lead to incorrect behavior if the right circumstances arose.

The bugs are as follows (line indicates the start of the problem area):

Bug 1a: free_msgf_mbx_ptr, line 424

The loop searching for predecessor mailbox pointers can overshoot by one and leave a freed block on the list. A mid-chain block is freed while still linked; a last-in-chain block is neither unlinked nor freed. This can cause a process hang in the message_status command.

This section of code also returns error_table_$unimplemented_version when it should return error_table_$bad_subr_arg, in the case where a supplied pointer doesn't match message_facility_'s mailbox structures (not a version mismatch).

Bug 1b: get_next_msgf_mbx_ptr, line 290

message_status passes the same variable for both ptr arguments of this entry, so assigning P_next_msgf_mbx_ptr = null () destroys the input pointer before the error recovery on line 304 reads it, causing the recovery to restart at static_msgf_mbx_ptr instead of stepping past the mailbox just freed. The fix makes a copy of the input parameter as input_msgf_mbx_ptr and uses it instead.

These two bugs (1a and 1b) together cause message_status -a to loop; 1a's fix ends the loop but will still report the mailbox twice; 1b's fix removes the duplication but by itself doesn't terminate the loop. (To demonstrate, create and accept messages on three mailboxes, do a message_status -all, then delete the middle mailbox and do message_status -all again.)

Bug 2: wakeup_processor, line 1300

The code tests whether the operation is being done on an "own" mailbox, but calls the non-own entry regardless of the test. The effect is that accept_message wakeups don't work on a mailbox to which the user has d and o permissions but not r, when the o should govern the outcome. (This looks like an accidental omitted fix from MCR7437, which the comments describe as "Modified to correctly figure out when to use mailbox_$own_xxx entries.")

Bug 3: create_message_array, line 1502

Message existence bits are indexed by array positions, but these positions shift when a newly read message is inserted below existing entries. The bit array is also sized from n_messages on entry, so the shifted index can run past its end. The fix sizes it by n_elements, shifts the bits along with the entries, and widens the array deletion pass to also match.

Bug 4: get_message_index, line 1454

The binary search won't work if there are holes in the array from deleted messages; this leads to a process hang and false reports of no message. Newly allocated msg_array elements (between n_messages and n_elements) are not cleared; the fix clears them.

Bug 5: print_message, line 1159

The return P_code parameter is not set when message read fails. (The pm command reports nothing when in some cases it should, when there's a genuine error other than error_table_$no_message.)

Bug 6: set_wakeup_state, line 511

The change_state variable should be initialized at the top; current code uses it before it's initialized.

Bug 7: get_msgf_mbx_ptr, line 379

A failed open leaves the new block linked, leaving a zombie entry which compounds bug 1.

Bug 8: Minor issues

There's a missing return statement in an error code ^= 0 block, there are several places where substr lengths could calculate to a negative number if senders are corrupted or missing, there's an uninitialized mbx_index variable, and an unnecessary array copy when n_messages = 0.

The most significant issues are Bug 1 and Bug 4; Bug 4 is not exposed to any current message commands but is present in message_facility_ and can be demonstrated via a test case.

Proposed Changes

The proposed changes to message_facility_.pl1 are shown by the compare_ascii output below:

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

A15         1) change(84-05-10,Lippard), approve(), audit(), install():
A16             Written by Jim Lippard.
A17         2) change(84-11-16,Lippard), approve(), audit(), install():
Changed by B to:
B15         1) change(1984-05-10,Lippard), approve(), audit(), install():
B16             Written by Jim Lippard.
B17         2) change(1984-11-16,Lippard), approve(), audit(), install():


A20         3) change(84-11-27,Lippard), approve(), audit(), install():
Changed by B to:
B20         3) change(1984-11-27,Lippard), approve(), audit(), install():


A24         4) change(84-12-18,Lippard), approve(), audit(), install():
Changed by B to:
B24         4) change(1984-12-18,Lippard), approve(), audit(), install():


A27         5) change(85-01-11,Lippard), approve(), audit(), install():
Changed by B to:
B27         5) change(1985-01-11,Lippard), approve(), audit(), install():


A31         6) change(85-01-23,Lippard), approve(), audit(), install():
A32             Modified to fix code calling mlr_.
A33         7) change(85-01-30,Lippard), approve(), audit(), install():
A34             Modified to speed up message selection by using a binary search.
A35         8) change(85-03-26,Lippard), approve(), audit(), install():
A36             Modified to speed up create_message_array and compact_message_array.
A37         9) change(85-04-16,Lippard), approve(), audit(), install():
Changed by B to:
B31         6) change(1985-01-23,Lippard), approve(), audit(), install():
B32             Modified to fix code calling mlr_.
B33         7) change(1985-01-30,Lippard), approve(), audit(), install():
B34             Modified to speed up message selection by using a binary search.
B35         8) change(1985-03-26,Lippard), approve(), audit(), install():
B36             Modified to speed up create_message_array and compact_message_array.
B37         9) change(1985-04-16,Lippard), approve(), audit(), install():


A40        10) change(85-05-31,Lippard), approve(85-11-18,MCR7298),
A41            audit(86-01-10,Spitzer), install(86-01-20,MR12.0-1006):
Changed by B to:
B40        10) change(1985-05-31,Lippard), approve(1985-11-18,MCR7298),
B41            audit(1986-01-10,Spitzer), install(1986-01-20,MR12.0-1006):


A44        11) change(85-09-02,Lippard), approve(85-11-18,MCR7298),
A45            audit(86-01-10,Spitzer), install(86-01-20,MR12.0-1006):
Changed by B to:
B44        11) change(1985-09-02,Lippard), approve(1985-11-18,MCR7298),
B45            audit(1986-01-10,Spitzer), install(1986-01-20,MR12.0-1006):
```

```
A48         12) change(86-04-23,Lippard), approve(86-05-27,MCR7418),
A49             audit(86-06-24,Hartogs), install(86-06-30,MR12.0-1080):
Changed by B to:
B48         12) change(1986-04-23,Lippard), approve(1986-05-27,MCR7418),
B49             audit(1986-06-24,Hartogs), install(1986-06-30,MR12.0-1080):


A52         13) change(86-06-04,Lippard), approve(86-06-24,MCR7432),
A53             audit(86-06-24,Hartogs), install(86-06-30,MR12.0-1080):
A54              Modified to properly zero fields in mail_format when sending messages.
A55         14) change(86-06-05,Lippard), approve(86-06-24,MCR7437),
A56             audit(86-06-24,Hartogs), install(86-06-30,MR12.0-1080):
A57              Modified to correctly figure out when to use mailbox_$own_xxx entries.
A58         15) change(86-08-06,Lippard), approve(87-03-18,MECR0001),
A59             audit(87-03-12,Fawcett), install(87-03-19,MR12.1-1002):
A60              Modified to check for unseen messages in wakeup processor.
A61         16) change(87-01-29,Lippard), approve(87-03-18,MECR0001),
A62             audit(87-03-12,Fawcett), install(87-03-19,MR12.1-1002):
A63              Modified to strip control characters from message comment field.
A64         17) change(87-05-08,Lippard), approve(87-04-20,MCR7669),
A65             audit(87-05-11,Fawcett), install(88-05-04,MR12.2-1045):
Changed by B to:
B52         13) change(1986-06-04,Lippard), approve(1986-06-24,MCR7432),
B53             audit(1986-06-24,Hartogs), install(1986-06-30,MR12.0-1080):
B54              Modified to properly zero fields in mail_format when sending messages.
B55         14) change(1986-06-05,Lippard), approve(1986-06-24,MCR7437),
B56             audit(1986-06-24,Hartogs), install(1986-06-30,MR12.0-1080):
B57              Modified to correctly figure out when to use mailbox_$own_xxx entries.
B58         15) change(1986-08-06,Lippard), approve(1987-03-18,MECR0001),
B59             audit(1987-03-12,Fawcett), install(1987-03-19,MR12.1-1002):
B60              Modified to check for unseen messages in wakeup processor.
B61         16) change(1987-01-29,Lippard), approve(1987-03-18,MECR0001),
B62             audit(1987-03-12,Fawcett), install(1987-03-19,MR12.1-1002):
B63              Modified to strip control characters from message comment field.
B64         17) change(1987-05-08,Lippard), approve(1987-04-20,MCR7669),
B65             audit(1987-05-11,Fawcett), install(1988-05-04,MR12.2-1045):


A68         18) change(88-03-31,Lippard), approve(88-04-18,MCR7876),
A69             audit(88-04-26,Parisek), install(88-05-04,MR12.2-1045):
Changed by B to:
B68         18) change(1988-03-31,Lippard), approve(1988-04-18,MCR7876),
B69             audit(1988-04-26,Parisek), install(1988-05-04,MR12.2-1045):


Inserted in B:
B74         19) change(2026-09-20,Lippard), approve(2026-09-20,MCR10176):
B75             Fix eight bugs: (1) free_msgf_mbx_ptr: search for predecessor
overshoots
B76             by one; (2) wakeup_processor: both arms of "own" test call the NON-own
B77             entry; (3) create_message_array: existence bits indexed by positions
that
B78             shift; (4) get_message_index: binary search doesn't converge over
holes,
B79             process hangs; (5) print_message: P_code not set when message read
fails;
B80             (6) set_wakeup_state: change_state not initialized; (7)
get_msgf_mbx_ptr:
B81             failed open leaves new block linked; (8) missing return, possible
negative
B82             substr lengths, uninitialized mbx_index, and no longer copy message
array
B83             in get_msg_array_ptr when n_messages = 0.
```

```
B84       .
Preceding:
A74                                                    END HISTORY COMMENTS */


Inserted in B:
B163               dcl     input_msgf_mbx_ptr     ptr;
Preceding:
A152               dcl     static_msgf_mbx_ptr    ptr internal static init (null ());


A289               msg_facility_mailbox_ptr = P_msgf_mbx_ptr;
Changed by B to:
B301               input_msgf_mbx_ptr = P_msgf_mbx_ptr; /* use a copy of the
parameter */
B302               msg_facility_mailbox_ptr = input_msgf_mbx_ptr;


A292               if msg_facility_mailbox_ptr = null () then next_msgf_mbx_ptr =
static_msgf_mbx_ptr;
Changed by B to:
B305               if input_msgf_mbx_ptr = null () then next_msgf_mbx_ptr =
static_msgf_mbx_ptr;


A304                                       if P_msgf_mbx_ptr = null () then
next_msgf_mbx_ptr = static_msgf_mbx_ptr;
A305                                       else next_msgf_mbx_ptr = P_msgf_mbx_ptr
-> msg_facility_mailbox.next_mbx_ptr;
Changed by B to:
B317                                       if input_msgf_mbx_ptr = null () then
next_msgf_mbx_ptr = static_msgf_mbx_ptr;
B318                                       else next_msgf_mbx_ptr =
input_msgf_mbx_ptr -> msg_facility_mailbox.next_mbx_ptr;


Inserted in B:
B394                                  call message_facility_$free_msgf_mbx_ptr
(msg_facility_mailbox_ptr, (0));
Preceding:
A381                                  return;


A421
A422      /* set msg_facility_mailbox_ptr to point to mbx in list just BEFORE the one
we're freeing */
A423                          found = FALSE;
A424                          do msg_facility_mailbox_ptr = static_msgf_mbx_ptr
A425                               repeat (msg_facility_mailbox.next_mbx_ptr)
A426                               while (msg_facility_mailbox.next_mbx_ptr ^= null ()
A427                               & ^found);
A428                               if msg_facility_mailbox.next_mbx_ptr =
freed_msgf_mbx_ptr then found = TRUE;
A429                          end;
A430
A431                          if msg_facility_mailbox.next_mbx_ptr = null () then
do; /* this msgf structure isn't in our list! */
A432                                  P_code = error_table_$unimplemented_version;
A433                                  return;
A434                             end;
A435                     end;
Changed by B to:
B435      /* set msg_facility_mailbox_ptr to point to mbx in list just BEFORE the one
we're freeing */
```

```
B436                     if static_msgf_mbx_ptr = null () then do; /* nothing in our
list at all */
B437                          P_code = error_table_$bad_subr_arg;
B438                          return;
B439                     end;
B440                     do msg_facility_mailbox_ptr = static_msgf_mbx_ptr
B441                          repeat (msg_facility_mailbox.next_mbx_ptr)
B442                          while (msg_facility_mailbox.next_mbx_ptr ^= null ()
B443                          & msg_facility_mailbox.next_mbx_ptr ^=
freed_msgf_mbx_ptr);
B444                     end;
B445                     if msg_facility_mailbox.next_mbx_ptr ^= freed_msgf_mbx_ptr
then do;
B446                          /* this msgf structure isn't in our list! */
B447                          P_code = error_table_$bad_subr_arg;
B448                          return;
B449                     end;
B450                end;


Inserted in B:
B496                change_state = FALSE; /* initialize */
Preceding:
A481                msg_facility_mailbox_ptr = P_msgf_mbx_ptr;


Inserted in B:
B532                allow_switch = ""b; /* initialize */
Preceding:
A516                substr (allow_switch, 1, 1) = substr
(msg_facility_mailbox.wakeup_state, 1, 1);


A800                array_length = MSG_ARRAY_ELEMENT_LENGTH * n_messages;
A801                destination_string_ptr = addr (msg_array (1));
A802                source_string_ptr = addr (internal_msg_array (1));
A803                destination_string = source_string;
Changed by B to:
B817                if n_messages > 0 then do;
B818                     array_length = MSG_ARRAY_ELEMENT_LENGTH * n_messages;
B819                     destination_string_ptr = addr (msg_array (1));
B820                     source_string_ptr = addr (internal_msg_array (1));
B821                     destination_string = source_string;
B822                end;


Inserted in B:
B1027               mbx_index = 0; /* initialize */
B1028
Preceding:
A1008               on cleanup begin;


A1159                         if code ^= 0 then return;
Changed by B to:
B1180                         if code ^= 0 then do;
B1181                              if code ^= error_table_$no_message then P_code =
code;
B1182                              return;
B1183                         end;


A1204                         message_sender = substr (local_mi.sender, 1, length
(rtrim (local_mi.sender)) - 2);
```

```
Changed by B to:
B1228                         message_sender = substr (local_mi.sender, 1, max (length
(rtrim (local_mi.sender)) - 2, 0));


A1302               else call mailbox_$incremental_read_index
(msg_facility_mailbox.index, area_ptr, READ_CURRENT, message_id,
Changed by B to:
B1326               else call mailbox_$own_incremental_read_index
(msg_facility_mailbox.index, area_ptr, READ_CURRENT, message_id,


Inserted in B:
B1340                         return;
Preceding:
A1316                    end;


Inserted in B:
B1370                                    unspec (msg_array) = ""b; /* clear new
elements for scan in get_message_index */
Preceding:
A1345                                    if msg_facility_mailbox.messages_ptr ^= null
() then do;


A1439               dcl     (max_idx, high_idx, low_idx, high_temp, low_temp) fixed
bin;
Changed by B to:
B1465               dcl     (max_idx, high_idx, low_idx) fixed bin;


A1453               low_idx = 1;
A1454               if ^msg_facility_mailbox.msg_array_compacted then max_idx =
msg_facility_mailbox.n_elements;
A1455               else max_idx = msg_facility_mailbox.n_messages;   /* faster */
Changed by B to:
B1479               /* The binary search below cannot cope with deleted entries, so
when the
B1480                  array has holes in it, scan instead. This is the transient
case,
B1481                  between a delete and the next compaction. The search can
therefore
B1482                  assume every entry it sees is live, which is why it no longer
has a
B1483                  case for deleted ones.  */
B1484
B1485               if ^msg_facility_mailbox.msg_array_compacted then do;
B1486                    do idx = 1 to msg_facility_mailbox.n_elements;
B1487                         if internal_msg_array.message_id (idx) ^= ""b then
B1488                              if substr (internal_msg_array.message_id (idx), 19,
54) = message_id then do;
B1489                                   P_message_index = idx;
B1490                                   return;
B1491                              end;
B1492                    end;
B1493                    P_code = error_table_$no_message;
B1494                    return;
B1495               end;
B1496
B1497               low_idx = 1;
B1498               max_idx = msg_facility_mailbox.n_messages;        /* faster */
```

```
A1461                    if internal_msg_array.message_id (idx) = ""b then do; /*
deleted */
A1462                              high_temp = min (idx + 1, max_idx);
A1463                              low_temp = max (idx - 1, 1);
A1464                              do while (high_temp < max_idx &
internal_msg_array.message_id (high_temp) = ""b);
A1465                                   high_temp = high_temp + 1;
A1466                              end;
A1467                              do while (low_temp > 1 &
internal_msg_array.message_id (low_temp) = ""b);
A1468                                   low_temp = low_temp - 1;
A1469                              end;
A1470                              if substr (internal_msg_array.message_id
(high_temp), 19, 54) > message_id
A1471                                   | internal_msg_array.message_id (high_temp) =
""b then high_idx = low_temp;
A1472                              else low_idx = high_temp;
A1473
A1474                              if idx > high_idx then idx = high_idx;
A1475                              else if idx < low_idx then idx = low_idx;
A1476                         end;
A1477
A1478                    else if substr (internal_msg_array.message_id (idx), 19, 54)
< message_id then do;
Changed by B to:
B1504                    if substr (internal_msg_array.message_id (idx), 19, 54) <
message_id then do;


A1502               dcl     msg_existence_array    (msg_facility_mailbox.n_messages)
bit (1) aligned;
Changed by B to:
B1528               dcl     msg_existence_array    (msg_facility_mailbox.n_elements)
bit (1) aligned;
B1529               dcl     existence_valid        bit (1) aligned;
B1530               dcl     kdx                    fixed bin;


Inserted in B:
B1537               existence_valid = TRUE;
Preceding:
A1509               old_msg_count = msg_facility_mailbox.n_messages;


A1547                                   msg_existence_array (idx) = TRUE;
Changed by B to:
B1576                                   if existence_valid then msg_existence_array
(idx) = TRUE;


Inserted in B:
B1598                                        unspec (msg_array) = ""b; /* clear new
elements for scan in get_message_index */
Preceding:
A1569                                        if msg_facility_mailbox.messages_ptr ^=
null () then do;


Inserted in B:
B1608                                        existence_valid = FALSE; /* array outgrew
msg_existence_array */
Preceding:
A1578                                   end;
```

```
A1585                                          found = FALSE;
A1586                                          do while (^found & jdx > 1);
A1587                                               if internal_msg_array.message_id
(jdx - 1) < message_id then found = TRUE;
A1588                                               else jdx = jdx - 1;
A1589                                          end;
A1590
A1591                                          if jdx ^= msg_facility_mailbox.n_messages
then do;
A1592                                               array_length =
MSG_ARRAY_ELEMENT_LENGTH * (msg_facility_mailbox.n_messages -
\c jdx);
A1593                                               call mrl_ (addr
(internal_msg_array (jdx)), array_length,
A1594                                                    addr (internal_msg_array
(jdx + 1)), array_length);
A1595                                          end;
A1596
A1597                                     end;
Changed by B to:
B1616                                     found = FALSE;
B1617                                     do while (^found & jdx > 1);
B1618                                          if internal_msg_array.message_id (jdx -
1) < message_id then found = TRUE;
B1619                                          else jdx = jdx - 1;
B1620                                     end;
B1621
B1622                                     if jdx ^= msg_facility_mailbox.n_messages then
do;
B1623                                          array_length = MSG_ARRAY_ELEMENT_LENGTH *
(msg_facility_mailbox.n_messages - jdx);
B1624                                          call mrl_ (addr (internal_msg_array
(jdx)), array_length,
B1625                                               addr (internal_msg_array (jdx + 1)),
array_length);
B1626                                          if existence_valid then /* keep the bits
with their entries */
B1627                                               do kdx =
msg_facility_mailbox.n_messages to jdx + 1 by -1;
B1628                                                    msg_existence_array (kdx) =
msg_existence_array (kdx - 1);
B1629                                               end;
B1630                                     end;
B1631                                end;


Inserted in B:
B1634                                if existence_valid then msg_existence_array (jdx) =
TRUE;
Preceding:
A1600                                d_permission = substr (access_mode, 2, 1);


A1629                    if msg_count < old_msg_count then do;
Changed by B to:
B1664                    if msg_count < old_msg_count & existence_valid then do;


A1632                              do idx = 1 to old_msg_count while (n_deleted <
old_msg_count - msg_count);
Changed by B to:
B1667                              do idx = 1 to msg_facility_mailbox.n_messages while
(n_deleted < old_msg_count - msg_count);
```

```
A1710                         person_id = before (substr (local_mra.sender_id, 1,
length (rtrim (local_mra.sender_id)) - 2), ".");
A1711                         project_id = after (substr (local_mra.sender_id, 1,
length (rtrim (local_mra.sender_id)) - 2), ".");
Changed by B to:
B1745                         person_id = before (substr (local_mra.sender_id, 1, max
(length (rtrim (local_mra.sender_id)) - 2, 0)
\c), ".");
B1746                         project_id = after (substr (local_mra.sender_id, 1, max
(length (rtrim (local_mra.sender_id)) - 2, 0)
\c), ".");


A1763                     message_sender = substr (local_mi.sender, 1, length (rtrim
(local_mi.sender)) - 2);
A1764                  else message_sender = substr (local_mi.sender, 1, length (rtrim
(local_mi.sender)) - 2)
Changed by B to:
B1798                     message_sender = substr (local_mi.sender, 1, max (length
(rtrim (local_mi.sender)) - 2, 0));
B1799                  else message_sender = substr (local_mi.sender, 1, max (length
(rtrim (local_mi.sender)) - 2, 0))

Comparison finished: 40 differences, 243 lines.
```

Mbuild script file in >udd>m>jjl>s>mcr>MCR10153.mb

Source code in >udd>m>jjl>s>mcr>message_facility_.pl1

Testing

>udd>m>jjl>s>msgf_test.pl1 is code that tests bugs 1a/b, 4, 5, and 7 before and after the changes. It also contains a regression test for bug 3, though it is not a test for bug 3 (reachability is hard to demonstrate). The code touched by bug 3 runs on every wakeup, every get_msg_array_ptr, so use of pm, dlm, am, dm all use it and their continuing to function properly is indication that the fix doesn't break anything; msgf_test.pl1 has a case 3 to test the functionality in the fixed version but doesn't demonstrate error in the current version.

Bug 2 is demonstrated with the following sequence of commands:

```
sa test1.mbx adosw   (no r)
am –mbx test1
sm –mbx test1 hello
pm –mbx test1
```

Unfixed: wakeup read calls wrong entry, fails with wakeup error, but pm works.

Fixed: message prints (you have o access) from am and pm.

Documentation

No changes to documentation.

Version History

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

Date	Revision	Author	Comment
2026-09-20	0.1	Jim Lippard	Pre-release version of document
2026-09-21	1.0	Jim Lippard	Version released for initial review/ comment.