To: Distribution
From: Betsy
Date: 17 March 1976
Subject: Multics Change Requests

Enclosed are copies of Multics Change Requests which were approved from March 1 - 15, 1976.

Multics Project internal working documentation. Not to be reproduced or distributed outside the Multics Project.
TITLE: Install new MST copy/merge/list/excerpt/write.

AUTHOR: Bernard Greenberg

Coded in: PL/1, AIM, other

Planned for System: 1.0

Category (Check One):
- Lib. Maint. Tools
- Sys. Anal. Tools
- Sys. Prog. Tools

Fixes Bug Number(s): 1.2.5

Document in MF:

User/Operations Visible Interface change:
- Yes
- No

Incompatible change:
- Yes
- No

Performance:
- Better
- Same
- Worse

Replaces MCR:

Objections/Comments: Add -brief control argument, remove abbreviations from tools (except em). Some facility for recording the author and contents of the tape must be developed for MST's.

Use these headings: Summary of Proposal, Reasons for Proposal, Implications, Detailed Proposal.

Summary: Replace copy_mst with a new version, with increased function for editing MST's.

Reasons:
1) The installed copy_mst is in version 1 PL/1, with iot, and has a needlessly difficult interface.
2) The need has often been felt for a tool to replace selected segments on a MST, or extract segments for inspection or other use.

Implications:
1) Vastly improved interface for copy_mst (i.e., copy_mst tape1 tape2 vs. "Type # of tapes to be copied:").
2) The ability to copy multi-reel MST's, which have not happened in several years, will be lost.
3) The ability to copy several tapes in one invocation, which is highly nonstandard, and not particularly useful will be lost.
4) It will be very valuable for debugging and system development to be able to replace single segments on tapes. This implies an increased effectiveness of the CISL Development Site, whereby users can fix tapes made at the MIT Site with the Network Transfer of a single segment, or simply a patch.
5) An ability to create usable MST's with no traceable origin or documentation is introduced. This is a tradeoff in a debugging situation. Like all powerful things, such a tool must be used carefully.

Detailed Proposal: See attached documentation.
Name: **copy_mst**, cpm

*copy_mst* is used to copy a Multics System Tape (either a Multics or BOS bootload tape) onto another reel of tape.

**Usage:**

```
copy_mst tapeid-1 tapeid-2
```

WHERE *tapeid-1* is the tape reel identifier of the tape to be copied, and *tapeid-2* is the tape reel identifier of the tape onto which the copy is to be made.

Name: **merge_mst**, mgm

*merge_mst* is used to copy a Multics System Tape (either a Multics or BOS bootload tape) onto another reel of tape, replacing selected segments with segments from the Storage System.

**Usage:**

```
merge_mst tapeid-1 tapeid-2 : name-1 name-2 ... name-n
```

WHERE *tapeid-1* is the tape reel identifier of the tape to be copied, and *tapeid-2* is the tape reel identifier of the new MST to be made. *name-1* to *name-n* are segment names (possibly star names, see "match_star_name_" in the MPM). Any segment which matches any of the names will be sought in the current working directory, and replaced in the tape copy. If no names are supplied, "**" is the default. If a segment with a separate linkage segment is replaced, a separate linkage segment will be sought in the working directory to replace it. However, if there is none, the linkage segment will be obtained from the object segment in the working directory. If a segment on the tape being replaced is not an object segment, but the matching segment in the working directory is, only the text of the segment will be written to the new tape.

A separate linkage section cannot be replaced without replacing the segment from whence it was separated.

Each name may be preceded by the "-stop" (-sp) control argument, in which case a call to debug is made immediately before the segment is written out, whether or not it has been replaced. This allows the user to inspect or modify the SLT entry in any arbitrary way. Before such a call is made, an informational message giving the address of the **X** segment and the SLT entry is printed out.

An informational message is printed out every time a segment is replaced on the tape with one from the working directory.
This procedure demonstrates the interactions between a user process and the Initializer in order to request an auto call line be assigned to the user. This program performs several functions:

1) Requests the Initializer to call a specified phone number and assign the line to this process.
2) Goes blocked to wait for confirmation from the Initializer.
3) Sets up an event call channel to be invoked if the line is hung up from the other end.
4) Attaches the line to the process through an iox_10 module.
5) Provides an entry to allow the user to disconnect the line.

All of these functions must be performed by any user program which uses the Auto Call Facility.

**Declaration Listing of Segment call_out**

Compiled by: Multics PL/I Compiler of November 24, 1975.
Compiled on: 02/24/76 1530.3 est Tue
Options: list map optimize

```pli
1 call_out: procedure;
2 /* *********************************************** */
3 /* */
4 /* This procedure demonstrates the interactions between a */
5 /* user process and the Initializer in order to request */
6 /* an auto call line be assigned to the user. This program */
7 /* performs several functions:
8 /* */
9 /* 1) Requests the Initializer to call a specified */
10 /* phone number and assign the line to this */
11 /* process.
12 /* 2) Goes blocked to wait for confirmation from the */
13 /* Initializer.
14 /* 3) Sets up an event call channel to be invoked */
15 /* if the line is hung up from the other end.
16 /* 4) Attaches the line to the process through an */
17 /* iox_10 module.
18 /* 5) Provides an entry to allow the user to */
19 /* disconnect the line.
20 /* */
21 /* All of these functions must be performed by any user */
22 /* program which uses the Auto Call Facility.
23 /* */
24 /* *********************************************** */
25
dcl nargs fixed bin,
arg_ptr pointer,
arg_len fixed bin,
arg char(arg_len) based(arg_ptr);

32 dcl ( dialed,
   attached,
   opened ) bit(1) aligned internal static initial("0b");

33 dcl 1 dial_manager_arg internal static aligned,
   version fixed bin initial(1),
   phone_number char(22),
   wakeup_channel fixed bin(71),
   tty_name char(32);

34 dcl 1 reply aligned,
   channel fixed bin(71),
   message fixed bin(71),
   origin bit(36),
   wait_list_index fixed bin;

35 dcl info pointer,
   1 info based(info) like reply;

36 dcl 1 status_flags aligned,
   dialed_up bit(1) unaligned,
   hung_up bit(1) unaligned,
   control bit(1) unaligned;
```

/* Internal static switches used to keep track of */
/* the progress of a call_out attempt */
/* Structure used in calls to dial_manager_, kept */
/* in internal static so that subsequent calls to */
/* call_out will use the same values. */
/* Used in call to ipc$block, contains message */
/* from Initializer when ipc$block returns */
/* Used to reference "hangup" message from ipc_ */
/* event call */
/* Returned by convert_dial_message_ */
/* = line is dialed and assigned */
/* = line has been disconnected */
/* Not used in Auto Call functions */
2 pad bit(33) unaligned;

dcl 1 wait_list aligned, /* Used in calls to lpc_iblock. Note that "chan" */
2 n_channels fixed bin initial(1), /* contains the same value as is kept in */
2 chan(1) fixed bin(71); /* dial_manager_arg唤醒_channel */

my_name char(32), /* For call to lpx_attach_ioname */
stream_name char(32), /* Device name returned by convert_dial_message_ */
dev_name char(32), /* Dim name returned by convert_dial_message_ */
dim_name char(32), /* For call to lpx_attach_ioname */
attach_description char(64);

dcl locb_ptr pointer; /* Number of tty lines attached by this process */

dcl n_dev fixed bin, /* Number of tty lines attached by this process */
code fixed bin(35);

call_outhangup entry (pointer), /* Convert_dial_message$return_io_module entry (fixed bin(71), char(*), char(*), fixed bin, */
convert_dial_message$return_io_module entry (fixed bin(71), char(*), char(*), fixed bin, /* aligned like status_flags, fixed bin(35)), */
dial_manager$dlal_out entry (pointer, fixed bin(35)),
dial_manager$terminate_dial_out entry (pointer, fixed bin(35));

dcl com_err_ entry options (variable), /* Error_table$badcall external fixed bin(35); */
convert_lpc_code_ entry (fixed bin(35)),
cu$arg_count entry (fixed bin),
cu$arg_ptr entry (fixed bin, pointer, fixed bin, fixed bin(35)),
lo$attach_loname entry (char(*), pointer, char(*), fixed bin(35)),
lo$close entry (pointer, fixed bin(35)),
lo$detach_locb entry (pointer, fixed bin(35)),
lo$open entry (pointer, fixed bin, bit(1) aligned, fixed bin(35)),
lpc$block entry (pointer, pointer, fixed bin(35)),
lpc$create_ev_chn entry (fixed bin(71), fixed bin(35)),
lpc$decl_ev_call_chn entry (fixed bin(71), pointer, pointer, fixed bin, fixed bin(35));

dcl error_table$badcall external fixed bin(35);
/* This is the main entry point. It is called with a */
/* telephone number to be called and the name of a stream */
/* that the line, once assigned, should be attached to. */

my_name = "call_out";

if dialled
  then do;
    call com_err_ (error_table$badcall, my_name,
        "Call out already in use, a previously dialled.", dial_manager_arg tty_name);
    return;
  end;

call cu$_arg_count (nargs);
if nargs = 2
  then do;
    call com_err_ (error_table$badcall, my_name,
        "/usage 1st call_out <phone_number> <stream_name>");
    return;
  end;

call cu$_arg_ptr (1, arg_ptr, arg_len, code);
  dial_manager_arg phone_number = arg;

call cu$_arg_ptr (2, arg_ptr, arg_len, code);
  stream_name = arg;

/* We now have the phone number and the stream. Before we */
/* can request that the Initializer place the call we must */
/* set up an InterProcess Communication channel over which */
/* the Initializer will signal the success or failure of the */
/* call. Note that the channel must be passed to the */
/* Initializer so that it will know what channel to use. */

call ipc$_create_ev_chn (wait_list.chan(1), code);
if code = 0
  then do;
    call convert_ipc_code_ (code);
    call com_err_ (code, my_name,
        "Unable to create ipc event channel.");
    return;
  end;

dial_manager_arg wakeup_channel = wait_list.chan(1);
call dial_manager$call_out (addr (dial_manager_args), code);
if code = 0
  then do;
    call com_err_ (code, my_name,
        "Unable to call out to "a., dial_manager_arg$phone_number);
call lpc$block (addr (wait_list), addr (*reply), code);
if code == 0 then do;
    call convert_lpc_code_ (code);
    call com_err_ (code, my_name,
                 "Error calling lpc$block.");
    return;
end;

call convert_dial_message_ (reply_message, dev_name, dim_name, n_dev, status_flags, code);
if "Status_flags.dialed_up
then do;
    call com_err_ (code, my_name,
                 "Error: call out failed.");
    return;
end;
dialed = "1"b;
dial_manager_arg tty_name = dev_name;

/*  */
*/
*/ Now that the line has been assigned, we must prepare a */
*/ handler for a possible hang up at the other end. Note */
*/ that we must use the same IPC channel that we previously */
*/ passed to the Initializer, as this is the only channel */
*/ that is known to the Initializer. */
*/
call lpc_decl_event_call_chn (wait_list.chn(1), addr (call_out$hangup), null(), 1, code);
if code == 0 then do;
    call convert_lpc_code_ (code);
    call com_err_ (code, my_name,
                 "Unable to set hangup handler, line will be attached anyway.");
end;
attach_description = dim_name + " " + dev_name;
call llo_sattach_ioname (stream_name, locb_ptr, attach_description, code);
if code == 0 then do;
    call com_err_ (code, my_name,
                 "Unable to attach stream "a to device "a, line will be hung up.", stream_name, dev_name);
go to term_dial;
end;
attached = "1"b;
call llo_open (locb_ptr, 3, "G"b, code);
if code == 0 then do;
    call com_err_ (code, my_name,
                 "Unable to open stream "a, line will be hung up.", stream_name);
go to detach_locb;
end;

opened = "1"b;

call com_err_ (0, my_name,
    "a attached to a through a", stream_name, dev_name, dim_name);

return;
This entry point is used to terminate an Auto Call connection. Since we allow only one active connection at a time and keep the channel name in internal static, there are no arguments to this entry.

```
my_name = "call_out$end_call_out";

if "dialed"
    then do;
        call com_err_ (0, my_name, "No lines currently attached.");
        return;
    end;

if opened
    then call lox_$close (locb_ptr, code);
opened = "$"b;

if attached
    then call lox_$detach_locb (locb_ptr, code);
attached = "$"b;

if dialed
    then call dial_manager_$terminate_dial_out (addr (dial_manager_arg), code);
dialed = "$"b;

call com_err_ (0, my_name, "Connection terminated, -a hung up.", dial_manager_arg, tty_name);
```

This entry will be invoked as an IPC event call by our user's request. Note that we first check to see that the wakeup is valid, and, if so, we then perform a variety of cleanup operations.

```c
my_name = "call_out$hangup";

if code != 0 then do;
    call com_err (code, my_name,
        "Hangup wakeup received, but unable to convert message from dial_control.");
    return;
end;

if dev_name != dial_manager_arg.tty_name then do;
    call com_err (0, my_name,
        "Hangup wakeup ignored for unknown channel -a.");
    return;
end;

if status_flags.hung_up then do;
    call com_err (0, my_name,
        "Spurious wakeup ignored -a.");
    return;
end;

if opened then call lox$close (locb_ptr, code);
opened = "0"b;

if attached then call lox$detach_locb (locb_ptr, code);
attached = "0"b;

dialed = "0"b;

call com_err (0, my_name,
    "Hangup occurred -a hung up and detached.");
return;
end call_out;
```
II. System Requirements for the Auto Call Facility

In order to make use of the Auto Call Facility, a site must first make changes to the hardware and to several system databases. The rest of this document covers:

1. Hardware Requirements
2. Configuration Cards
3. The Channel Master File (CMF) and the Channel Definition Table (CDT)
4. Channel Access Control
5. System Messages

1. Hardware Requirements for Auto Call

The Multics Auto Call Facility utilizes three hardware components. The first is an HSLA Auto Call board - board number HSC 351, supplied by Honeywell. The second is a Bell system auto call unit - unit number 801A AGU, supplied by the telephone company. The third component is a standard Bell 103A modem.

The HSC 351 board includes installation instructions and one cable for connection to the Bell 801A auto call unit. The Bell 103A modem is also connected to the HSC 351 board, in this case by a standard dataset cable which should be ordered with the modem.

2. Configuration Cards

The HSC 351 auto call board uses an entire HSLA slot, unlike some other HSLA boards which allow two channels/slot. Thus HSLA slot 5, which, with a dual asynchronous HSLA board, provides HSLA channels 5 and 21, will provide only channel 5, an asynchronous 300 baud line, when the HSC 351 is used. Configuration cards should therefore be modified to indicate a single 300 baud channel for the appropriate slot.

3. The Channel Master File and the Channel Definition Table

When the appropriate hardware and configuration card changes have been made the Channel Master File should be modified. The only difference between a standard entry and an auto call entry is that the auto call entry should indicate a service type of "autocall". For example: "service: autocall;". Additional
changes to such items as the charge rate for channel usage may be made but are not required. Once the Channel Master File has been updated it should be converted to a Channel Definition Table using the cv_cmf command and the CDT installed.

4. Channel Access Control

The Channel Master File provides the ability to limit use of a channel by Access Isolation Mechanism access class. This limitation is observed by the Auto Call Facility, but does not provide specific user/project oriented access control. In order to provide this additional control, the Auto Call Facility makes use of the Access Control Segment concept first implemented by the Resource Control Package. Before the Facility may be used, an Access Control Segment must exist in "system_control_dir" for each available auto call channel. The segment(s) must have a name of the form <channel name>.acs, for example: "tty605.acs". When a user process requests that an auto call channel be assigned, the user's access to the Access Control Segment will be checked and the line will be assigned only if the user has at least "r w" access to the segment.

5. System Messages

With the addition of functionality to the Initializer, a number of new messages may appear in the Initializer's log and output. All of the new messages are produced by the procedure dial_ctl_ and are written to both the log and the Initializer console. In the messages listed below, the following generic terms are replaced with values as indicated:

<channel>
  is the ASCII name of the tty channel involved.
<error message>
  is an error message extracted from error_table_.
  because of an error code returned to dial_ctl_.
<process id>
  is the 12 digit octal process id assigned to a
  process when it is created. In the messages
  listed below, the process id is used only when the
  Initializer is unable to locate the process in
  question.
<project>
  is the project id portion of a process identifier.
  (ASCII)
<reason>
  is a standard code for the reason for a process
  termination, for example, "new_proc".
<telephone number> is the ASCII string representing the telephone number being called.

<user> is the user id portion of a process identifier. (ASCII)

"dial_ctl_: now terminating dial_out of <channel> for <user>,<project>"

This message will be outputted whenever an auto call line hangs up for whatever reason. In general, it will be preceded by one of the messages below indicating the reason (line hungup, user requested, etc) the connection is being terminated.

"dial_ctl_: dial_out line being hung, <channel>, reason: <reason>"

This message occurs when a process with an active auto call connection terminates. <reason> is the cause of the process termination.

"dial_ctl_: can't find process <process id>"

This message results when the Initializer receives an auto call request from a process that can not be found in the answer table.

"dial_ctl_: now dialing <channel> to <telephone number> for <user>,<project>"

This message is printed when the Initializer begins to dial an auto call line for a user. Note that this message can be used to determine responsibility for calls placed by the Auto Call Facility for such purposes as billing for long distance time used.

"dial_ctl_:finish_dial_out: can't find process <process id>"

This message appears if the Initializer has initiated an auto call dial for a user and is then unable to locate the appropriate process once the dialing has been completed.

"dial_ctl_: <channel> dialed out for <user>,<project>"

This message is outputted when a successful dial out has occurred and the Initializer is giving the line to the user
process.

"dial_ctl_: <error message> getting dial_out_status or tty info
(<channel> for <user>, <project>)."

This message results when an error is encountered attempting
to get the status of an auto call channel. <error message>
is the ASCII string associated with an error_table_ code
which details the error which occurred.

"dial_ctl_: <error message> dial_out (<channel> for
<user>, <project>) failed."

This message results from any other error returned to
dial_ctl_ while attempting to complete a dial_out request.

"dial_ctl_: user requested dial_out term: <channel>,
<user>, <project>"

This message occurs when a user requests that the auto call
connection be terminated normally.
New tty_Control Operations

The following are additions to the Control Operation section of
the tty_write in the MPM (AG93), pages 3-12, 3-13, and 3-14.

dial_out initiates the dialing of a user specified phone number. This operation is allowed only for the process that
originally attached the device (generally the initializer process). The info_ptr must point to the
following structure:

    dcl 1 info_structure aligned,
        2 num_chars fixed bin,
        2 phone_number char(n);

where n is greater than or equal to info_structure.num_chars. The phone number dialed will
be the numeric characters in the first (leftmost) num_chars characters of phone_number (non-numeric
characters are ignored). Up to an 12 digit phone number may be dialed. Upon completion of the dial out
process, a wakeup is issued on the event channel associated with the I/O switch. The dial_out operation
is considered active until either status is obtained using the dial_out_status order (see below) or the I/O
switch is closed and detached. The I/O system status codes this order may return are error_table$big_arg or
error_table$small_arg if the phone number to be dialed contains too many, too few digits respectively,
error_table$undefined_order_request if the channel is
already dialed up, or error_table$io_no_permission if the caller lacks needed permission.

dial_out_status

is used to obtain the status of an active dial_out
operation. Valid status can only be obtained once
(after valid status has been returned the dial_out
operation is no longer active). Note that during the
dial out process this operation may return an indicator
that status is not yet available. Until status is
available the dial out process has not completed. The
process initiating the dial out can wait for a wakeup
on the event channel for the tty channel before issuing
the dial_out_status order. The info_ptr should be a
null pointer. The possible values of code are one of
the following I/O system codes:

error_table$device_not_usable
No telephone auto-call unit attached to channel
(or no power to unit). This implies a physical
difficulty.

error_table$invalid_state
Telephone channel is already being used.

Jay Goldman - 1 - 02/12/76
New tty Control Operations

error_table &no_connection
The phone call was not completed.

error_table &no_operation
The channel can not support dial out. This implies a logical problem with the attempted dial out.

error_table &no_wired_structure
The dial out was successful but Multics was unable to provide the needed wired storage to support use of the channel. The phone line has been hung up. Another attempt may be successful.

event_info
is used to obtain the event channel associated with a tty switch. It is the same event channel returned by the "read_status" and "write_status" orders but this order call can be used even when the channel associated with the I/O switch is hung up. The info_ptr must be to a fixed binary (71) variable.
**TITLE:** AST lock metering  

**AUTHOR:** R. Mullen

<table>
<thead>
<tr>
<th>Category (Check One)</th>
<th>Status</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lib. Maint. Tools</td>
<td>Written</td>
<td>2 Mar, 76</td>
</tr>
<tr>
<td>Sys. Anal. Tools</td>
<td>Expires</td>
<td>3 Jan 76</td>
</tr>
<tr>
<td>DOCUMENTATION CHANGES</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Status:** A 3 Jan 76  
- **Expires:** 3 Jan 76

<table>
<thead>
<tr>
<th>Objections/Comments:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>None (Reason)</td>
<td>Info Segs</td>
</tr>
<tr>
<td>Other (Name)</td>
<td></td>
</tr>
</tbody>
</table>

**Use these headings:**  

**Summary:**

lock$lock Ast to keep the following meters in the SST to be printed by fsm:
- total time the AST is locked
- total time spent waiting for AST to unlock
- total number of AST lockings

**Reasons:**

The AST lock is suspected of being a bottleneck in NSS. If
locked_time / real_time > .3 it will imply a potential problem.
If wait_time / real_time > .5 it will imply that Multiprogramming
depth is increased by a small but intolerable amount.

**Implications:**

AST lock interference may be reduced if need be by:

A. Using per aste or per AST pool locks
B. Changing backup to dump according to dtem
C. Growing AST or critical ASTEPOOLS.
D. Unlocking AST during certain hardcore io operations.
E. Reconfiguration of ASTE pools by salvager
F. Attempt to not deactivate segs with pages on BulkStore
G. Backup to call hphcs_$deactivate_seg.
**TITLE:** Command To Set Sons Volume  

**AUTHOR:** Van Vleck

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Document</td>
<td>Specify One or More</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MCR 1685</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>STATUS</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Written</td>
<td>3/2/76</td>
</tr>
<tr>
<td>Expires</td>
<td>9/13/76</td>
</tr>
</tbody>
</table>

**DOCUMENTATION CHANGES**

- **User/Operations-visible Interface change?** [X] yes [ ] no
- **Incompatible change?** [X] yes [ ] no
- **Performance:** [X] Better [ ] Same
- **Replaces MCR**

**Objections/Comments:**

**Use these headings:** Summary of Proposal, Reasons for Proposal, Implications, Detailed Proposal.

**SUMMARY:** Install highly privileged command to set the sons volume for a directory. The master-dir switch is not affected.

**REASONS:** In order to improve MIT performance it was necessary to set the sons volume of >pdd to the public hierarchy volume. The entry hphcs $set sons lvld already exists for the dumper; this change just supplies a command interface.

**IMPLICATIONS:** This command should only be used on >pdd or in situations where setquota is appropriate, since it falsifies master directory control records and quotas.
Name: set_sons_volume

This highly privileged command sets the sons_lvid for a directory, if the user has modify on the directory and its parent, and the directory is empty. The master directory switch is not affected.

Usage: set_sons_volume path luname

where path is the directory to be modified and luname is the logical volume.

Notes: The normal use of this command is in the system_start-up, etc, to set the sons volume of >pdd to a value other than "root." This command should only be used on >pdd or in situations where setquota is appropriate, since it falsifies master directory control records and quotas.
**TITLE:** Add &rf and &qf features to do

**AUTHOR:** Steve Herbst

<table>
<thead>
<tr>
<th>STATUS</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Written</td>
<td>2/27/76</td>
</tr>
</tbody>
</table>

**Category (Check One):**
- Lib. Maint. Tools
- Sys. Anal. Tools
- Sys. Prog. Tools

**Problem Area:**
- Lib. Maint. Tools
- Sys. Prog. Tools

**Problem Category:**
- Document

**EXPires:** 9/1/76

**DOCUMENTATION CHANGES**
- MPM (Vol, Sect.)
- PLS (AN #)
- MOSN (Sect.)
- MPAM (Sect.)
- MSAM (Sect.)

**Objections/Comments:**
- Info Segs
- Other (Name)
- None (Reason)

**Use these headings:** Summary of Proposal, Reasons for Proposal, Implications, Detailed Proposal.

**SUMMARY:**
Add the &rf and &qf features to the do command, as natural complements to the recently installed &f feature. The string &rfn is replaced by a list of the nth through last arguments, requoted. The string &qfn is replaced by a list of the nth through last arguments with quotes doubled.
The do command has three modes, the long/brief mode, the nogo/go mode, and the absentee/interactive mode. These modes are kept in internal static storage and are thus remembered from call to call within a process. The modes are set by invoking the do command as follows:

```
do mode
```

where mode is one of the following: -long (-lg), -brief (-bf), -nogo, -go, -absentee, or -interactive.

If the long/brief mode is long, then the expanded command line is printed on error_output before it is executed or passed back. If the long/brief mode is brief, then the command line is not printed. The default for this mode is brief.

If the nogo/go mode is nogo, then the expanded command line is not passed to the command processor for execution. If the nogo/go mode is go, then the expanded line is passed to the command processor (if the do command was invoked as a command). If do is invoked as an active function, then the nogo/go mode is ignored. The default for this mode is go.

If the absentee/interactive mode is absentee, then the do command establishes an on unit for the any_other condition during the execution of the expanded command line. This is mainly of use in an absentee environment, in which any invocation of the default any_other on unit terminates the process. In the absentee mode, any signal caught by do merely terminates execution of the command line, not the process. A number of conditions, however, are not handled by dq but are passed on for their standard Multics treatment; they are cput, alrm, quit, program_interrupt, command_error, command_query_error, command_question, and record_quota_overflow. (For a description of these conditions see "List of System Conditions and Default Handlers" in Section VI of the MPM Reference Guide.) If the absentee/interactive mode is interactive, then do does not catch any signals. The default for this mode is interactive.

Quote-Doubling and Requoting

In addition to the parameter designators &1 ... &9, the do command also recognizes two more sets of parameter designators. They are &q1 ... &q9, to request quote-doubling in the actual argument as it is substituted into the expanded command line, and &r1 ... &r9, to request that the actual argument be requoted as well as have its quotes doubled during substitution.

Quote-doubling can be described as follows. Each parameter designator in the command_string to be expanded is found nested a certain level deep in quotes. If a designator is found to not be within quotes, then its quote-level is zero; if it is found between a single pair of quotes, then its quote-level is one; and so on. If the parameter designator &q1 is found nested to quote-level L then, as control_arg1 is substituted into the expanded command line each quote character found in control_arg1 is replaced by 2**L quote characters during insertion. This permits the quote character to survive the
TITLE: Fix entryname bug in exec_com

AUTHOR: Steve Herbst

Summary:
Fix bug in exec_com that causes the error message "Entry name too long" when the name of an exec_com is 29 characters long & does not end in .ec.
**TITLE:** Fix &attach bug in abs_io  
**AUTHOR:** Steve Herbst

<table>
<thead>
<tr>
<th>Category (Check One)</th>
<th>SHW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lib. Maint. Tools</td>
<td></td>
</tr>
<tr>
<td>Sys. Anal. Tools</td>
<td></td>
</tr>
<tr>
<td>Sys. Prog. Tools</td>
<td></td>
</tr>
</tbody>
</table>

- **Coded in:** [X] PL/I  [ ] ALM  [ ] other  
- **Planned for System MR:** L.0  
- **Fixes Bug Number(s):** X  
- **Documented in MTB:** 355  
- **User/Operations-visible:**  
  - Interface change?  [ ] yes  [X] no  
  - Incompatible change?  [ ] yes  [ ] no  
  - Performance:  [ ] Better  [X] Same  
  - [ ] Worse  
- **Replaces MCR**

**Objections/Comments:**

<table>
<thead>
<tr>
<th>Use these headings:</th>
<th>Summary of Proposal, Reasons for Proposal, Implications, Detailed Proposal.</th>
</tr>
</thead>
</table>

**SUMMARY:** Fix bug in abs_io that causes user_input to remain attached to an exec_com (by &attach) after &quit if &detach is not specified before the &quit.

**REASON:** &quit should restore the old attachment.
MULTICS CHANGE REQUEST

TITLE: Implement new faster tty_read

AUTHOR: Robert S. Coren

Planned for System: MR 4.0
Fixes Bug Number(s): not applicable
Documented in MTB: 262
Incompatible Change: no
User/Operations-visible Interface Change: no
Coded in: ( )PL/I ( )ALM ( )other-see below
Performance: ( )better ( )same ( )worse

DOCUMENTATION CHANGES (specify one or more)
MPM (vol/sect) Refguide 3 MPAM (sect)
MOSN (sect) MSAM (sect)
PLMs (AN#) AN85
Info Segs tty_changes, canonicalization
Other

OBSERVATIONS/COMMENTS:

SUMMARY: Install new version of tty_read described in MTB262, which also implements canonicalization as described in MTB 251.

REASONS: Part of general upgrade of ring-zero typewriter DIM. Improves efficiency of input processing and makes it more consistent; allows for future addition of user-substitutable translation tables and special characters.

IMPLICATIONS: Canonicalization will be consistent, but may not always produce identical results to old software (see MTB 251). "Invisible" control characters will be discarded in "can" and/or "erkl" mode.

DETAILED PROPOSAL: See MTB 262.
A new version of the ring-zero tty DIM will be installed soon which incorporates the following changes:

1. Control characters which do not change the carriage or paper position (i.e., any control character other than backspace, horizontal tab, newline, carriage return, form-feed, vertical tab, or space) cannot be input except as octal escapes if the terminal is in either "can" or "erkl" mode.

2. A lot of bugs are eliminated from canonicalization of input, and well-established rules are established for combinations of overstrikes, erase/kill characters, and escape sequences. Please type "help canonicalization" or dprint >doc>info>canonicalization.info for details.

canonicalization.info will be a non-standard info segment suitable for dprinting, which will contain the list of rules presented in MTB 251, and possibly some of the examples. A slightly revised version of MTB 251 will also be appended to the discussion of canonical form in Section 3 of the MPM Reference Guide.
**Multics Change Request**

<table>
<thead>
<tr>
<th>Title: Fix bugs in system control</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Author: Paul A. Green</td>
<td></td>
</tr>
<tr>
<td>CATEGORY: Lib. Maint. Tools</td>
<td></td>
</tr>
<tr>
<td>Status: A</td>
<td></td>
</tr>
<tr>
<td>Expires: 9/19/76</td>
<td></td>
</tr>
<tr>
<td>User/Operations-visible Interface change? Yes/No</td>
<td></td>
</tr>
<tr>
<td>Performance: Better/Same/Worse</td>
<td></td>
</tr>
<tr>
<td>Replaces MCR</td>
<td></td>
</tr>
</tbody>
</table>

**Objections/Comments:**

Use these headings: SUMMARY, REASONS, IMPLICATIONS, DETAILED PROPOSAL (Optional)

**SUMMARY:**
1. Fix test mode to work.
2. Fix input-wait timeout to work.

**Reasons:** Both features stopped working in the new system control.

**Implications:** None

**Detailed Proposal:** Replace sc_init_pll, system_control_pll, mc_tty_pll
**TITLE:** Fix bugs in mrd.util

**AUTHOR:** Paul A. Green

-Coded in: [✓] PL/I [ ] AIM [ ] Other
-explain in DETAILED PROPOSAL
-Planned for System MR 3.1
-Fixes Bug Number(s) MPRE 9221
-Documented in MTF
-User/Operations-visible [ ] yes [✓] no
-Performance: [ ] Better [✓] Same
-Interface change? [✓] yes [ ] no
-Incompatible change? [✓] yes [ ] no
-Replaces MCR

**STATUS**

Written 02-25-76
Status A 3/9/76
Expires 9/9/76

**SUMMARY:** Fix threading errors in mrd.util that have caused intermittent message coordinator problems at many sites.

**Reasons:** Better reliability, fewer crashes. Slightly better error messages.

**Implications:** None

**Detailed Proposal:** Replace mrd_util.pl1 and syscon_mseg.incl.pl1
**MULTICS CHANGE REQUEST**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MCR</strong></td>
<td>1696</td>
</tr>
<tr>
<td><strong>STATUS</strong></td>
<td></td>
</tr>
<tr>
<td><strong>DATE</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Written</strong></td>
<td>02/25/76</td>
</tr>
<tr>
<td><strong>Expires</strong></td>
<td>06/25/76</td>
</tr>
<tr>
<td><strong>TITLE</strong></td>
<td>Fix bug in print_syserr_log</td>
</tr>
<tr>
<td><strong>AUTHOR</strong></td>
<td>Larry Johnson</td>
</tr>
<tr>
<td><strong>Planned for System</strong></td>
<td>4.0</td>
</tr>
<tr>
<td><strong>Fixes Bug Number(s):</strong></td>
<td>not applicable</td>
</tr>
<tr>
<td><strong>Documented in MTB:</strong></td>
<td>not applicable</td>
</tr>
<tr>
<td><strong>Incompatible Change:</strong></td>
<td>no</td>
</tr>
<tr>
<td><strong>User/Operations-visible Interface Change:</strong></td>
<td>no</td>
</tr>
<tr>
<td><strong>Coded in:</strong></td>
<td>( )PL/I ( )ALM ( )other-see below</td>
</tr>
<tr>
<td><strong>Performance:</strong></td>
<td>( )better ( )same ( )worse</td>
</tr>
<tr>
<td><strong>DOCUMENTATION CHANGES</strong></td>
<td>MPM (vol,sect)</td>
</tr>
<tr>
<td></td>
<td>MPAM (sect)</td>
</tr>
<tr>
<td></td>
<td>MOSN (sect)</td>
</tr>
<tr>
<td></td>
<td>MSAM (sect)</td>
</tr>
<tr>
<td></td>
<td>PLMs (AN#)</td>
</tr>
<tr>
<td></td>
<td>Info Segs</td>
</tr>
<tr>
<td></td>
<td>Other</td>
</tr>
<tr>
<td></td>
<td>None (reason)</td>
</tr>
<tr>
<td><strong>CATEGORY</strong></td>
<td>( )Lib. Maint. Tools</td>
</tr>
<tr>
<td></td>
<td>( )Sys. Anal. Tools</td>
</tr>
<tr>
<td></td>
<td>( )Sys. Prog. Tools</td>
</tr>
<tr>
<td></td>
<td>( )355</td>
</tr>
<tr>
<td></td>
<td>( )BOS</td>
</tr>
<tr>
<td></td>
<td>( )Salvager</td>
</tr>
<tr>
<td></td>
<td>( )Ring Zero</td>
</tr>
<tr>
<td></td>
<td>( )Ring One</td>
</tr>
<tr>
<td></td>
<td>( )SysDaemon/Admin</td>
</tr>
<tr>
<td></td>
<td>( )Runtime</td>
</tr>
<tr>
<td></td>
<td>( )User Command/Subr</td>
</tr>
<tr>
<td><strong>OBJECTIONS/COMMENTS:</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

The -exclude and -match control arguments do not work properly if the syserr message is an "=".

**Detailed Proposal**

Change the control arguments to check the expanded text.
VER. 3
MULTICS CHANGE REQUEST

TITLE: Fix bug in get_process_id_
AUTHOR: Larry Johnson

Planned for System: 4.0
Fixes Bug Number(s): not applicable
Documented in MTB: not applicable
Incompatible Change: no
User/Operations-visible Interface Change: no
Coded in: (B)PL/I ( )ALM ( )other-see below
Performance: (l)better ( )same ( )worse

DOCUMENTATION CHANGES (specify one or more)
MPM (vol,sect) MPAM (sect)
MOSN (sect) MSAM (sect)
PLMs (AN#)
Info Segs
Other
None (reason)

OBJECTIONS/COMMENTS:

Headings are: SUMMARY, REASONS, IMPLICATIONS, DETAILED PROPOSAL (optional)

Summary

The module get_process_id_ (which contains get_group_id_, get_lock_id_, get_pdir_, get_authorization_) keeps the values to be returned by these entries in internal static. All of these values are initialized with the first call to any of the entry points. The initialization check is incorrect, however, so that all of the values are initialized with every call to any of the entries. This results in two extra hcs_calls every time this subroutine is called.

Detailed Proposal

Fix the initialization check so that all values are initialized once per process (per ring).

Page 1 of 1
SUMMARY: While writing the tape_ansl_PLM, a number of minor bugs have been discovered, none of which have ever been reported. The majority of these bugs will never be encountered except under highly unlikely combinations of errors; nevertheless, they should be fixed.

REASONS: Increased reliability

IMPLICATIONS: Improved reliability
list_mst

list_mst is used to find out what segments are on a Multics System tape (either a Multics or BOS bootload tape).

Usage: list_mst tapeid name-1 name-2 ... name-n

WHERE tapeid is the tape reel identifier of the Multics System Tape to be listed. name-1 through name-n are names of segments to be listed if they appear on the tape. Note that they may contain stars (see "match_star_name_" in the MPM), and the default is "**" (list everything) if no names are given.

A summary line giving the length of each segment, and its primary name is printed out for each segment listed. A special name is printed out for segments written with the mst generator "first_name" keyword, for the proper name of such segments does not appear on the tape.

excerpt_mst

excerpt_mst is used to excerpt given segments from a Multics System Tape (either a Multics or BOS bootload tape).

Usage: excerpt_mst tapeid name-1 name-2 ... name-n

WHERE tapeid is the tape reel identifier of the Multics System Tape to be excerpted. name-1 through name-n are the names of the specific segments to be extracted. These names may contain stars (see "match_star_name_" in the MPM), and the default is "**" (divulge entire tape) if no names are given. If a given segment has a separate linkage segment on the tape, and has been extracted, the separate linkage segment will be extracted as well. Segments extracted are created in the current working directory. Bit counts are set from the SLT entry on the tape, as opposed to the actual length of the segment on the tape.

An informational message is printed whenever a segment is extracted.
Name: `write_mst`

`write_mst` is used to write short Multics System Tapes. This is usually useful only for writing tapes for BOS, either complete BOS tapes, or tapes to be used with the BOS LOADDM command (see the Multics Operators Handbook).

Usage: `write_mst tapeid names-and-control-args-

WHERE `tapeid` is the tape reel identifier of the `M` tape to be written. `-names-and-control-args-` are the names of segments to be written, and control arguments specifying collection marks and optional text/link decoding. The arguments may be any intermixture of

1) Segment specifications
2) Collection mark specifications.

A collection mark specification is the control argument `-col` or `-collection`. It causes sequential collection marks to be written every time it appears in the command line.

A segment specification is the (relative pathname of a segment to be written on the tape, possibly preceded by the optional control arguments `-text`, `-link`, `-stop` (short names `-tx`, `-lk`, and `-sp`). The segment is written with the entry name supplied as its name on the tape. Both its bit count/current length and actual length are derived from its bit count in the storage system. Other than the bit count and current length, its SLT entry is given as zero.

For each segment, if the `-link` argument is supplied, a separate segment, consisting of the separated linkage of the segment, is written on the tape, with the name `<name>.link`.

If the `-text` argument is supplied, only the text of the segment is written, as opposed to the whole segment. This, as `-link`, requires that the segment specified be a standard object segment. Note that `-text` and `-link` neither contradict nor interact with each other.

If the `-stop` argument is supplied, a call is made to debug immediately before the segment is written out, so that the user may modify the SLT entry in any desired way. A message giving the location of the segment and the SLT entry is printed out before such a call is made.
TITLE: Add per-process segfault counter to system

AUTHOR: R. Bratt

SUMMARY:

Add per-process segfault counter to pds. Add new entry to hcs_get_process_usage return structure to return segfault count.

Note: This is compatible since the return structure already includes an entry specifying the number of items to return.

Also, add per process vtoc read and write meters.

REASON:

This may prove to be a useful meter. It costs very little to provide.
Name: hcs$_{get\_process\_usage}$

This subroutine returns information about a process's usage of Multics since it was created. It provides data about processor and memory usage.

Usage

```plaintext
declare hcs$_{get\_process\_usage}$ entry (ptr, fixed bin(35));
call hcs$_{get\_process\_usage}$ (info_pointer, code);
```

1) `info_pointer` is a pointer to the structure in which process information is returned (see Notes below). (Input)

2) `code` is a standard status code. (Output)

Notes

The format of the structure based on `info_pointer` is:

```plaintext
declare 1 process_usage,
  2 number_wanted fixed bin,
  2 cpu_time_used fixed bin(71),
  2 memory_usage fixed bin(71),
  2 number_of_page_faults fixed bin(35),
  2 amount_of_prepaging fixed bin(35),
  2 process_virtual_time fixed bin(71)
```

1) `number_wanted` is set by the calling program to specify the number of other entries in the structure to be filled in. The entry itself (the numbers wanted) is not included in this count. The value of `number_wanted` would cause `number_wanted` entries listed below to be filled in. A smaller number, `n`, will cause the first `n` entries to be filled in. (Input)

2) `cpu_time_used` is set to the amount of processor time (in microseconds) used by the calling process. (Output)
3) memory_usage

Is a measure of the primary (core) memory used by this process. The units of memory usage are page-seconds, normalized to account for the size of primary memory actually in use. (Output)

4) number_of_page_faults

Is set to the number of demand page faults this process has taken. (Output)

5) amount_of_prepaging

Is the number of pages prepaged for this process. (Output)

6) process_virtual_time

Is the amount of processor time (in microseconds) used exclusive of page fault and system interrupt processing time. (Output)

7) number_of_segment_faults

Is set to the number of segment faults this process has taken. (Output)

8) number_of_bounds_faults

Is set to the number of bounds faults this process has taken. (Output)

9) number_of_vtoc_reads

Is set to the number of VTOC read io operations the process has done (output).

10) number_of_vtoc_writes

Is set to the number of write VTOC io operations the process has done (output).
Provide new entry point, teco$macro, which allows the initial macro executed to be changed. (For compatibility reasons we must use a new entry point rather than a command line argument.)

REASONS:

teco's inability to start anything but a user-oriented editing session greatly reduces its usefulness. The addition of a mechanism for specifying the initial command buffer to be executed allows teco to be called from command level to perform specific tasks, e.g., abbrev editing.
1. Loops cannot cross macro boundaries, i.e. a loop cannot start in one macro and end in another. This does not, however, prohibit the M command from being used within a loop.

2. A macro can modify itself if it is in a Q-register. Note, however, that the current invocation of the macro is not affected; only future accesses to the Q-register. If the macro is invoked by the EM command, the results of modifying the file are hard to predict: TECO reads the command string directly from the file.

3. When a macro is invoked by the EM command, it should be noted that the name of the macro is found in the Q-register named "." Thus several macros can be put in one segment with the first command in the segment being OQ. (Don't forget to put all the appropriate names on the segment).

4. If an M or EM command is given as the last command in one macro, the command is interpreted as a goto rather than a call. Thus, unlimited M's can be done in this manner although there is an implementation defined limit to the depth of calls.

5. When the TECO command is entered, a macro named start_up is searched for. If it is found, the arguments to TECO are put onto the pushdown stack and the start_up macro is executed. If no start_up macro is found, the string EI/filename/J is executed, where filename is the first argument to TECO. At the present time, there is a start_up macro in the TECO library. When the start_up macro is called, the first thing on the pushdown list is the number of arguments TECO was called with. The remaining items in the list are the actual string arguments to TECO going from left to right on the command line.

6. If TECO$macro is invoked as a command, then TECO behaves as described above except that the first argument is not placed in the pushdown list. Instead, TECO looks for a macro named by the first argument and executes it in place of the usual start_up macro. If an error is encountered before TECO command level is reached (and the macro need not ever allow TECO to reach command level), then TECO simply returns.

CODING CONVENTIONS FOR MACROS

Since there are only a small number of Q-registers (95), each with a one-character name, there are serious problems in writing a set of macros that are compatible. A set of macros become incompatible if one macro uses a Q-register for long-term storage that any other macro uses at all. There are two ways this effect can be combatted. First, by establishing certain coding conventions, and second, by use of a documented macro library. Probably the most important coding convention is the specification of which Q-registers can be used inside a macro for
TITLE: COBOL Bug Fixing

AUTHOR: William K. O'Neill (CEO-B)

-Coded in [X]PL/I [X]ALM [ ] other-explain in DETAILED PROPOSAL
-Planned for System MR 3.1
-Fixes Bug Number(s) 143, 150, 162
-Documented in MTB

Category (Check One)

Lib. Maint. Tools
Sys. Anal. Tools
Sys. Prog. Tools

355

Document Specify One or More

Bos
Salvager
Ring Zero
Ring One
SysDaemon/Admin.
Runtime
User Cmd/Subr.

Objections/Comments:

Use these headings: Summary of Proposal, Reasons for Proposal, Implications, Detailed Proposal.

Summary

(a) Bug 143: A READ into a file that has an occurs may store the data in the wrong location. It is difficult to program around this problem.

(b) Bug 150: The compiler will abort if the number of data names, plus procedure names, plus section names exceeds approximately 800. Many large programs will not compile.

(c) Bug 162: The compiler will abort if bad syntax is encountered in the organization clause in the environment division. This is a frequent error for a new user of Multics COBOL to make.
TITLE: Add information-only entry points to 
dir_control_error

AUTHOR: Lee Scheffler

Summary: Add two new entries, dir_control_error$contents_info and 
dir_control_error$attributes_info. These entries will 
operate exactly as do the current dir_control_error$contents 
and dir_control_error$attributes, but will not generate 
auditing messages.

Several places in hardcore directory control (both oss and nss) 
call the contents and attributes entries solely to determine 
whether or not they may return a detailed error code to their 
callers. Change these calls (in append, del dir tree, 
find entry, initiate and truncate) to call the contents_info 
and attributes_info entry points.

Reasons: Bug fix required for Air Force acceptance of AIM.

Eliminate 90% of the (erroneous) auditing messages in the 
syserr log, particularly the message generated each time 
hcs$make_seg is called on a segment that already exists.

Implications: There will be a better distinction between the two 
different uses of dir_control_error, namely, to determine 
the proper error code to return on a real error, and 
determining whether the user has sufficient access to get 
a detailed status code.
**Title:** Fix erroneous audit calls from set_privileges  

**Author:** Lee Scheffler

- Coded in XPL/I  
- Planned for System MR 4.0  
- Fixes Bug Number(s) unreported  
- Documented in MTR  
- User/Operations-visible Interface change? yes  
- Performance: Better

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sys. Prog. Tools</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>355</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BOS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Salvager</td>
<td></td>
<td></td>
<td>MPM (Vol, Sect.)</td>
</tr>
<tr>
<td>X Ring Zero</td>
<td></td>
<td></td>
<td>PLMS (AN #) AN 75</td>
</tr>
<tr>
<td>Ring One</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SysDaemon/Admin.</td>
<td></td>
<td></td>
<td>MOSN (Sect.)</td>
</tr>
<tr>
<td>Runtine</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Cmd/Subr.</td>
<td></td>
<td></td>
<td>MSAM (Sect.)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Document</th>
<th>Specify One or More</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Objections/Comments:**

- Info Segs

**Use these headings:** Summary of Proposal, Reasons for Proposal, Implications, Detailed Proposal.

**Summary:** Change set_privileges to call protection audit only when there is an actual change to the system privileges of the calling process. (It currently calls protection audit on every call, whether or not an actual change to privileges occurs.)

**Reasons:**
1. Bug fix required for Air Force acceptance of AIM.  
2. Remove excessive audit messages from the syserr log.

**Implications:** None
TITLE: Add check for valid entry pointer to protection_audit $access_denied

AUTHOR: Lee Scheffler

- Coded in [ ] PL/I [x] AIM [ ] other-explain in DETAILED PROPOSAL
- Planned for System MR 4.0
- Fixes Bug Number(s) [ ] unreported
- Documented in MTB [ ]
- Interface change? [ ] yes [x] no
- Incompatible change? [ ] yes [x] no
- Performance: [ ] Better [x] Same [ ] Worse
- Replaces MCR

Summary: Change protection_audit $access_denied to check that its entry pointer parameter is a valid entry pointer (offset A = 0) before blindly charging ahead picking up attributes out of the branch pointed to.

Reasons:
1. Bug fix required for Air Force acceptance of AIM.
2. Auditing messages for cases where dir control error is passed a directory pointer (offset = 0) instead of an entry pointer contain garbage.
**TITLE:** Change convert authorization to properly convert null access class strings

**AUTHOR:** L. Scheffler

<table>
<thead>
<tr>
<th>Category (Check One)</th>
<th>Document</th>
<th>Specify One or More</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lib. Maint. Tools</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sys. Anal. Tools</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sys. Prog. Tools</td>
<td>355</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DOCUMENTATION CHANGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOS</td>
</tr>
<tr>
<td>Salvager</td>
</tr>
<tr>
<td>Ring Zero</td>
</tr>
<tr>
<td>Ring One</td>
</tr>
<tr>
<td>SysDaemon/Admin.</td>
</tr>
<tr>
<td>MOSN (Sect.)</td>
</tr>
<tr>
<td>X Runtime</td>
</tr>
<tr>
<td>X User Cmd/Subr.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Objections/Comments:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Info Segs</td>
</tr>
<tr>
<td>Other (Name)</td>
</tr>
<tr>
<td>None (Reason)</td>
</tr>
</tbody>
</table>

Use these headings: Summary of Proposal, Reasons for Proposal, Implications, Detailed Proposal.

**Summary:** Change convert authorization $ from string to reject a null access class string if the name for level 0 has not been set to null.

**Reasons:**

1. Bug fix required for Air Force acceptance of AIM.

2. For compatibility with sites not using AIM, a convention was established that a null access class character string would correspond to "system low", defined to be level 0, no categories. This should only be true if level 0 has a null name. If a site chooses a non-null name for level 0, a null access class string should be rejected as invalid.

**Implications:** Greater consistency
TITLE: Add audit message for illegal returns

AUTHOR: Lee Scheffler

- Coded in: PL/I □ AIM □ other
   - explain in DETAILED PROPOSAL
- Planned for System MR 4.0
- Fixes Bug Number(s): □
- Documented in MBT 355
- User/Operations-visible Interface change? □ yes □ no
- Incompatible change? □ yes □ no
- Performance: □ Better □ Same □ Worse
- Replaces MCR

Category (Check One):
- Lib. Maint. Tools
- Sys. Anal. Tools
- Sys. Prog. Tools

Expires: 3/12/76

DOCUMENTATION CHANGES

- Fix(es) Bug Number(s): □
- Documented in MBT 1355
- Planned for System MR 4.0
- Prog. Tools
- User/Operations-visible BOS
- Incompatible change? □ yes □ no
- Performance: □ Better □ Same □ Worse
- Replaces MCR

Doc. Type Specify One or More
- MPM (Vol, Sect.)
- PLMS (AN #) AN75
- MOSN (Sect.)
- MPAM (Sect.)
- MSGM (Sect.)

Objections/Comments:

Use these headings:
- Summary of Proposal
- Reasons for Proposal
- Implications
- Detailed Proposal

Summary: Change restart fault to call protection audit if an attempt is made to restart with a set of invalid machine conditions.

Reasons: This is an auditable protection event that has not previously been audited.

Implications: Auditing mechanism is more complete.
TITLE: Modify map355 to increase assembly limits

AUTHOR: M. Grady


SUMMARY:
Modify the map355 command to produce a $limit card for the GCOS assembly of 355map source which specifies 128k core limit for the assembly.

REASONS:
The latest changes to dia_man caused the memory usage during assembly to exceed the current 32k limit. This caused the assembler to bomb.
TITLE: Install FAST subsystem

AUTHOR: S. Barr

SUMMARY:

Install FAST as described in MTB 248 with two modifications.

1. If a path is given with the run command, it can specify a source segment or an object segment.

2. Change the ready message from "r" to "r hhmm".

DETAILED PROPOSAL:

See attached description of the run command.
run

**name:** run

The run command executes a BASIC or FORTRAN program. After execution it closes all input/output files and frees common blocks.

**usage**

```
run {path}
```

where path is the pathname of an object segment. If path is not given, the run command compiles the temporary text and executes it. In order for the command to work, the entryname used with the last old, new, or save command must have a language suffix. If path is specified and the entryname has a language suffix, the run command expects the segment to contain the source program and compiles and executes it. If path is specified and the entryname does not contain a language suffix, the run command expects the segment to contain object code and executes it.

**example**

```plaintext
! old test.basic
r

! run (program execution)
r

! run std /object segment "std" in working directory (program execution)
r

! run /temporary text is not changed by run (program execution of test.basic)
r
```
<table>
<thead>
<tr>
<th>TITLE:</th>
<th>Improve delete's error messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTHOR:</td>
<td>S. Herbst</td>
</tr>
<tr>
<td></td>
<td>JWG</td>
</tr>
<tr>
<td>STATUS</td>
<td>DATE</td>
</tr>
<tr>
<td>Written</td>
<td>2-23-76</td>
</tr>
<tr>
<td>Expires</td>
<td>3-12-76</td>
</tr>
<tr>
<td>Category (Check One):</td>
<td>Lib. Maint. Tools</td>
</tr>
<tr>
<td>Fix Bug Number(s):</td>
<td>Sys. Prog. Tools</td>
</tr>
<tr>
<td>Documented in MTB:</td>
<td>355</td>
</tr>
<tr>
<td>User/Operations-visible:</td>
<td>BOS</td>
</tr>
<tr>
<td>Interface change?</td>
<td>Yes [x] No</td>
</tr>
<tr>
<td>Incompatible change?</td>
<td>Yes [x] No</td>
</tr>
<tr>
<td>Performance:</td>
<td>Better [x] Same</td>
</tr>
<tr>
<td>Replaces MCR:</td>
<td>X</td>
</tr>
<tr>
<td>DOCUMENTATION CHANGES:</td>
<td></td>
</tr>
<tr>
<td>Document</td>
<td>Specify One or More</td>
</tr>
<tr>
<td>MPM (Vol, Sect.):</td>
<td></td>
</tr>
<tr>
<td>MOSN (Sect.):</td>
<td></td>
</tr>
<tr>
<td>MSAM (Sect.):</td>
<td></td>
</tr>
<tr>
<td>Objections/Comments:</td>
<td></td>
</tr>
<tr>
<td>Into Segs</td>
<td>info segs</td>
</tr>
<tr>
<td>Other (Name):</td>
<td></td>
</tr>
<tr>
<td>None (Reason)</td>
<td>doc ok</td>
</tr>
</tbody>
</table>

Use these headings: Summary of Proposal, Reasons for Proposal, Implications, Detailed Proposal.

**SUMMARY:**

Fix the delete command to include the entire pathname with its error messages.

**REASONS:**

Currently, only the entry name is printed. For example, if `x` does not exist, the command "delete xy" produces the misleading error message:

"Some directory in pathname does not exist. y"
TITLE: Implement star convention in adjust_bit_count

AUTHOR: S. Herbst

STATUS DATE
Written 2-23-76
Expires 5-21-76

Objections/Comments:

Use these headings: Summary of Proposal, Reasons for Proposal, Implications, Detailed Proposal.

SUMMARY:

Change the adjust_bit_count command to implement the star convention.

REASONS:

Now that bit counts are more important, this change makes it easier for users to maintain correct bit counts on their segments.
**TITLE:** Correct bug in detection of eof in ntape

**AUTHOR:** R. Bratt

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Coded in:</td>
<td>X</td>
<td>PL/I</td>
<td>ALM</td>
</tr>
<tr>
<td>Planned for System MR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fixes Bug Number(s)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Documented in MTB</td>
<td>355</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**STATUS DATE**

- Written: 2-24-76
- Status: A 8/24
- Expires: 9/21/76

**DOCUMENTATION CHANGES**

- Fix Bug Number(s) Sys. Prog. Tools
- Document Specify One or More

**User/Operations-visible Interface change?**
- Yes [X] No

**Incompatible change?**
- Yes [X] No

**Performance:**
- Better [X] Same

**Replaces MCR**
- Yes [X] No

**Objections/Comments:**

- None (Reason) fix bug

**Use these headings:** Summary of Proposal, Reasons for Proposal, Implications, Detailed Proposal.

**SUMMARY:**

Correct eof detection in ntape by changing incorrect constant.
TITLE: Implement dump_fnp and patch_fnp Commands

AUTHOR: Robert S. Coren

Planned for System: not applicable 4.0
Fixes Bug Number(s): not applicable
Documented in MTE: not applicable
Incompatible Change: no
User/Operations-visible Interface Change: no
Coded in: (E)PL/I ( )ALM (E)other-see below
Performance: ( )better ( )same ( )worse

DOCUMENTATION CHANGES (specify one or more)
MPM (vol,sect) MPA* (sect)
MPHS (sect) MSAM (sect)
PLMs (AN#) AN85
Info Segs
Other

OBJECTIONS/COMMENTS:

Headings are: SUMMARY, REASONS, IMPLICATIONS, DETAILED PROPOSAL (optional)

SUMMARY: 1. Implement command, dump_fnp, and control operation in ring-zero typewriter DIM, "dump_fnp", to allow a user to print the contents of specified locations of FNP memory on his/her terminal.

2. Implement command, patch_fnp, and corresponding control operation, to allow a user to modify specified locations in FNP memory from Multics command level.

REASONS: It has frequently been found to be useful, when investigating actual or suspected bugs in FNP software, to be able to examine the contents of FNP memory. The inability to do this without access to an FNP console has often been inconvenient. It is very occasionally useful to be able to modify the contents of a running FNP, either to test assumptions about or to correct known software bugs.

IMPLICATIONS: None.

DETAILED PROPOSAL: See attached command documentation. The two control operations will make use of a term-wired buffer (using 108/b) to pass the required data between ring zero and the FNP. Accordingly, a new segment, fnp_dump_data, will be added to the system header for this purpose. The "dump_fnp" and "patch_fnp" operations will be privileged and highly privileged, respectively; new rate entries, GENERIC and GENERIC, will be added. It is not proposed to make these two operations available through iox_%control.
**Command Descriptions**

**Name:** dump_fnp

This privileged command enables a user to print the contents of specified locations in FNP memory on his/her terminal, in a format similar to that used by dump_segment.

**Usage**

```
dump_fnp fnp_id address nwords -control_arg-
```

- **fnp_id** is the identifier of the FNP whose memory is to be dumped. It must be either "a", "b", "c", or "d".
- **address** is an octal number indicating the starting address from which memory is to be dumped.
- **nwords** is an octal number giving the number of 10-bit words that are to be dumped. It must not be greater than 4000(8).
- **control_arg** may be -character (or -ch), and indicates, if present, that the contents of the specified memory locations are to be printed as ASCII characters as well as octal numbers. It may appear anywhere on the command line.
name: patch_fnp

This highly-privileged command enables a user to modify specified contiguous locations in FNP memory.

usage

patch_fnp fnp_id address values

fnp_id is the identifier of the FNP to be patched. It must be "a", "b", "c", or "d".

address is an octal number identifying the first location to be patched.

values are one or more octal values to replace the contents of consecutive words of FNP memory starting at the location specified by address; each value may be a maximum of 6 digits.

Note: The user is asked to verify the correctness of the patches before they are made; a record of the patches is entered in the syserr log and printed on the syserr console.
CONTROL OPERATIONS

dump_fnp is used to dump the contents of FNP memory.

Usage

declare phcs_tty_order entry (fixed bin, char (*), ptr, fixed bin,
fixed bin(35));
call phcs_tty_order (fnp_no, "dump_fnp", info_ptr, ignore, code);

fnp_no is the number of the relevant FNP, such that 0 < fnp_no <= 4. (Input)

info_ptr points to the following structure: (Input)

dcl 1 dump_fnp_info aligned,
   2 fnp_address fixed bin,
   2 fnp_length fixed bin,
   2 bufp ptr;

fnp_address is the address of the first word of FNP memory to be dumped.

fnp_length is the number of 18-bit words to be dumped.

bufp points to a buffer in which the dumped words are to be returned to the caller.

code is a standard system status code. (Output)

patch_fnp is used to modify specified locations of FNP memory.

Usage

declare phcs_tty_order entry (fixed bin, char (*), ptr, fixed bin,
fixed bin(35));
call phcs_tty_order (fnp_no, "patch_fnp", info_ptr, ignore, code);

Arguments are as above; the buffer pointed to by patch_fnp_info.bufp contains the values to be patched into FNP memory.
TITLE: Change format of pl1_version_

SUMMARY: Change the format of pl1_version_ to include a release number.

REASONS: pl1_version_ contains an ASCII string that the PL/I compiler places in each object segment to identify the compiler used to create the segment. This string, pointed at by std_symbol_header.gen_version, presently contains the date the compiler was installed, which differs at MIT and IPCO. To uniquely identify a compiler, therefore, we need a release number.

DETAILED PROPOSAL: A unique release number will be used with the installed compiler. Since we have been numbering the releases in our conversations with Toshiba, we will use that scheme in pl1_version_. pl1_version_ for MR4.0 might say, for example:

Multics Change Request

| TITLE: | Fix bug in propagate |
| AUTHOR: | S. Herbst |

- Coded in [ ] PL/I [] ALM [ ] other-
- explain in DETAILED PROPOSAL
- Planned for System MR
- Fixes Bug Number(s)
- Documented in MTB
- User/Operations-visible
- Interface change? [x] yes [ ] no
- Incompatible change? [ ] yes [x] no
- Performance: [ ] Better [x] Same
- Replaces MCR

<table>
<thead>
<tr>
<th>Category (Check One)</th>
<th>Lib. Maint. Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sys. Anal. Tools</td>
</tr>
<tr>
<td></td>
<td>Sys. Prog. Tools</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STATUS</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Written</td>
<td>2-23-76</td>
</tr>
<tr>
<td>Status</td>
<td>63217</td>
</tr>
<tr>
<td>Expires</td>
<td>9-2-76</td>
</tr>
</tbody>
</table>

DOCUMENTATION CHANGES

<table>
<thead>
<tr>
<th>Document</th>
<th>Specify One or More</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MPM (Vol, Sect.)</td>
</tr>
<tr>
<td></td>
<td>Ring Zero</td>
</tr>
<tr>
<td></td>
<td>Ring One</td>
</tr>
<tr>
<td></td>
<td>SysDaemon/Admin.</td>
</tr>
<tr>
<td></td>
<td>Runtime</td>
</tr>
<tr>
<td></td>
<td>X User Cmd/Subr.</td>
</tr>
</tbody>
</table>

Objections/Comments:

Use these headings: Summary of Proposal, Reasons for Proposal, Implications, Detailed Proposal.

SUMMARY:

Fix bug in iox_propagate that causes a modes operation on a re-attached I/O switch to return error_table_not_attached.

REASONS:

In some cases, a modes operation is valid on an attached but closed switch. In others, the error code error_table_no_operation should be returned. Bug is caused by improper initialization of an attached iocb.
Summary

The following bugs have been found in the syserr mechanism:

1. Messages could be logged out of order due to an incorrect sequence of locking the wired and paged logs to copy messages out.

2. Zero length messages could cause a crash when logged (on some hardware), or a crash when the log is validated at start up.

3. When equal messages are written on the console, they are padded with blanks out to the full length of the original message.

Detailed Proposal

Problem 1 will be fixed by changing the way in which the logs are locked. Currently, a syserr interrupt is handled as follows.

1. lock the wired log
2. copy messages into wired stack
3. unlock wired log
4. lock paged log
5. copy messages in
6. unlock paged log
It is possible that while one syserr log interrupt is between steps 3 and 4, another processor could complete steps 1 thru 4 and log a message out of order. The sequence will be changed to the following.

1. lock paged log
2. lock wired log
3. copy messages into wired stack
4. unlock wired log
5. copy messages into paged log
6. unlock paged log
**Title:** Answering Service Changes for Auto Call  
**Author:** David M. Jordan  
**TVV:**

<table>
<thead>
<tr>
<th>Category (Check One)</th>
<th>Document</th>
<th>Specify One or More</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lib. Maint. Tools</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sys. Anal. Tools</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sys. Prog. Tools</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lib. Maintain. Tools</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Reported fixes:**
-bug number(s)

**User/Operations-visible Interface change:** yes

**Performance:**
- Better
- Same
- Worse

**Replaces MCR:**

**Objections/Comments:** Info Segs

**Other (Name):** SRB

**None (Reason):**

Use these headings: SUMMARY, REASONS, IMPLICATIONS, DETAILED PROPOSAL (Optional)

**Summary:**
Install the Answering Service portion of the Multics Auto Call Facility as described in MTB 247 and in the accompanying user and external site documentation.
To: Whom It May Concern

From: David M. Jordan

Date: February 27, 1976

Subject: The Multics Auto Call Facility

Auto call hardware and associated software allow a computer to place telephone calls to other computers or terminals over standard telephone lines. This allows a relatively inexpensive alternative to more formal networks for special purpose low speed communications.

The Multics Auto Call Facility provides the necessary support software to allow controlled and accountable use of auto call hardware by arbitrary user processes. The facility relies on the Initializer process to perform such functions as controlling access to the hardware, establishing a communications link, and allowing for resource accounting.

The remainder of this document is divided into three sections:

User Documentation
System Requirements
New tty Control Operations
I. Use of the Multics Auto Call Facility

The Multics Auto Call Facility allows the user access to pre-dialed auto call lines. All control operations (line assignment, telephone dialing, and hangup processing) are performed by the Initializer in response to user generated Inter Process Communication requests or in response to hardware conditions. After a line has been assigned, the user may perform input and output over the line using the standard Multics I/O system iox_.

The remainder of this documentation describes three entry points of interest to the user:

- `dial_manager_$dial_out` to request that the Initializer assign an auto call line and place a call to a specified telephone number.
- `dial_manager_$terminate_dial_out` to request that the Initializer hang up an auto call line and deassign it from the requesting process.
- `convert_dial_message_$return_io_module` to interpret status returned by the Initializer to determine such information as channel name and line status.

Because all requests are handled through Inter Process Communications, various functions peripheral to auto call must be performed by the user process. In order to demonstrate the sequence of calls required and data structures needed, a sample program which performs the basic required functions is also provided.
**Name**: dial_manager_$dial_out

This entry point is used to request that an auto call line be dialed to a given telephone number and, if the line is successfully dialed, that the line be assigned to the requesting process.

**Usage**

declare dial_manager_$dial_out entry (pointer, fixed bln(35));
call dial_manager_$dial_out (addr(dial_manager_arg), code);

1. **dial_manager_arg**
   
is as described in [Notes](#) below. Note also that `dial_manager_arg.version` must be 1, and that `dial_manager_arg.phone_number` and `dial_manager_arg.wakeup_channel` must be supplied. Note also that `dial_manager_arg.tty_name` may be either a null string (in which case the system will attempt to assign any available auto call tty channel) or a specific tty channel (in which case the system will attempt to assign that channel only). (Input)

2. **code**
   
is a standard status code. (See [Notes](#) below.) (Output)
**Name:** dial_manager_$\_terminate\_dial\_out

This entry point is used to request that the Initializer hang up an auto call line and unassign it from the requesting process.

**Usage**

```plaintext
declare dial_manager_$_terminate_dial_out entry (pointer,
fixed bin(35));

call dial_manager_$_terminate_dial_out
(addr(dial_manager_arg), code);

1. dial_manager_arg
   is as described in Notes below. Note also that
dial_manager_arg.version must be 1, that
dial_manager_arg.tty_name must be supplied, and
that dial_manager_arg.wakeup_channel must be
supplied and have the same value as was used when
the original dial_manager_$_dial_out was made.
(Input)

2. code
   is a standard status code. (See Notes below.)
(Output)
```
**Name**: convert_dial_message$\_return\_io\_module

This entry point is used to process Inter Process Communication messages from the Initializer regarding the status of an auto call line. In addition to returning line status, this entry also returns the device name and IO module name for use in attaching the line through the lox_ I/O system.

**Usage**

```plaintext
declare convert_dial_message$\_return\_io\_module entry (fixed
type bin(71), char(*), char(*), fixed bin, 1 aligned like
status_flags, fixed bin(35));
call convert_dial_message$\_return\_io\_module (info.message,
dev_name, dim_name, n_dev, status_flags, code);
```

1. **info.message**
   - is the message portion of the structure filled in by ipc$_block or supplied as an argument to an IPC event call procedure. For more information see the description of ipc_ in the MPM Subsystem Writers' Guide, AK92. (Input)

2. **dev_name**
   - is the name of the tty device assigned or null if no device was assigned. (Output)

3. **dim_name**
   - is the name of the lox_ IO module to be used with the assigned device. (Output)

4. **n_dev**
   - is the number of tty devices currently assigned to this process. (Output)

5. **status_flags**
   - is a structure with the following declaration:
     ```plaintext
declare 1 status_flags aligned,
     2 dialed_up bit(1) unaligned,
     2 hung_up bit(1) unaligned,
     2 control bit(1) unaligned,
     2 pad bit(33) unaligned;
```
   The "dialed_up" bit will be on if the line is currently dialed up. The "hung_up" bit will be on if the line has hung up. The "control" bit is used only in reference to calls to dial_manager_ entries other than dial_out and terminate_dial_out. (Output)
6. code

is a standard status code. (See Notes below.) (Output)

Notes

1. `dial_manager_arg`

dial_manager_arg is a structure used to pass a variety of information about an auto call request to the procedure `dial_manager_`. It has the following declaration:

```c
declare 1 dial_manager_arg Internal static aligned,
  2 version fixed bin initial (1),
  2 phone_number char(22),
  2 wakeup_channel fixed bin(71),
  2 tty_name char(32);
```

1. version

indicates the version of the structure that is being used. Currently this must be 1.

2. `phone_number`

for calls to `dial_manager_${dial_out}`, this is the telephone number to be called. Note that non-numeric characters will be ignored, so the user need not remove them from a telephone number string.

3. `wakeup_channel`

is an InterProcess Communication channel to be used in communications from the Initializer. Note that the channel should be the same for all calls used in reference to the same auto call session.

4. `tty_name`

is used for calls to `dial_manager_${terminate_dial_out}` to indicate which channel should be disconnected. In calls to `dial_manager_${dial_out}`, it must be either a null string (in which case the Initializer will attempt to assign any available auto call channel) or a specific tty channel to be used for the auto call attempt.

2. Status codes.

dial_manager_${dial_out} and dial_manager_${terminate_dial_out} may return any of the following status codes:

- `status_code_1`
- `status_code_2`
- `status_code_3`
1. error_table_$bad_arg, if dial_manager_arg.version != 1.
2. Any codes returned by message_segment_$add_file, if dial_manager_ was unable to place the details of a request into the message segment provided by the Initializer.
3. Any codes returned by ipc_$block.
4. Any codes returned by convert_dial_message_.
5. error_table_$action_not_performed for any other errors.

convert_dial_manager_$return_to_module will return
error_table_$action_not_performed if any error is encountered.
In order to help the user understand the sequencing of calls required for use of the Auto Call Facility a sample program has been provided on the next few pages. The main entry point "call_out" accepts as arguments a telephone number and stream name. It calls dial_manager_dial_out to request that a call be placed to the specified number, and, if the attempt is successful, it then attaches the assigned line through the appropriate lox_IO module.

The second entry point "call_out$end_call_out" performs the functions required for a user to terminate an auto call session. In addition to requesting that the Initializer hang up the line, it also makes the appropriate lox_calls to close and detach the stream.

The last entry point "call_out$hangelog" is designed to act as an IPC event call procedure to be invoked if the line hangs up for reasons other than a user request. It also performs the various functions required to clean up the lox_attachment.

Note that this procedure is designed to handle only one auto call connection at a time and thus has various flags in internal static to insure that it will not be re-entered while a previous auto call session is still in progress.