Subject: MCR10086, mbuild Enhancements and Bug Fixes

Author: Gary Dixon

Date: 2021-02-20

Multics Ticket: #213 #217 #219

MTB-1003 describes the new mbuild command. This MCR proposes 3 enhancements to this new tool, and a variety of bug repairs.

The three enhancements make use of library maintenance tools to further automate the code build, audit, and installation process. These tools are:

- archive_sort: used to sort components in source or object archives into alphanumeric order.
- peruse_crossref: used to report on library source modules that reference a given include file.
- verify_info: the new info segment format and content checking tool proposed in MTB-1004.

In addition, several small bugs will be fixed that allow mbuild to be used when deleting files from the library, or moving sources from one bound segment to another.

Proposed Changes

This MCR changes/adds three mbuild requests to use the additional maintenance tools described above. It also recommends repairs for several bugs found within mbuild during past months. These changes are discussed in the next few subsections.

Changed Request: archive_prep (arch)

The **archive_sort** tool will be called from the existing **archive_prep** (**arch**) request. When building a bound segment, a third step is now performed.

1. Update components of the source/object archive(s).
2. **Call the archive_sort tool to sort source and object archive components into ascending alphanumeric order by component name.**
3. Bind the object archive(s) into a bound segment.

Multics coding standards call for sorting of all bound segment archives. Standards require using an Order statement in the bind file to control sequence of object components within the final bound segment.

However, programs which are not following these standards can use the order of object components in their object archive(s) to control position of the object in the final bound segment. To support such programs, a `-no_sort` control argument will be added to the archive_prep request.

See the *Documentation* section below for a description of this control argument. Code changes are primarily in mbuild_archive_.pl1.

New Request: xref

The **peruse_crossref** tool will be called from a new **xref** request. By default, it will call peruse_crossref for each include file in the installation directory. However, the user can give a SEG_NAME argument to get output for individual include files.

See the *Documentation* section below for a description of the xref request. Code primarily resides in the new mbuild_xref_.pl1 source module.

New Request: verify (vi)

The **verify_info** tool will be called from a new **verify (vi)** request. By default, it will call the verify_info tool for each info segment in the installation directory. However, the user can give a SEG_NAME argument to get verify_info output for individual info segments. The new request passes any control arguments to the verify_info tool.

See the *Documentation* section below for a description of the xref request. Code primarily resides in the new mbuild_info_checks_.pl1 source module.

Changes to: bound_mbuild_.bind

The new xref and verify requests are being added as new source modules in the mbuild subsystem: mbuild_info_checks_ contains code for the verify request; mbuild_xref_ contains code for the xref request. The bindfile therefore changes as shown below:

```
Inserted in B:
B53                         mbuild_info_checks_,
B54                         mbuild_xref_,
Preceding:
A48                         mbuild_lib_names_,


Inserted in B:
B105      objectname:       mbuild_info_checks_;
B106
B107      objectname:       mbuild_xref_;
B108
Preceding:
A98       objectname:       mbuild_install_;
```

Changes to: mbuild_request_tables_.alm

A multics_request statement is added to mbuild_request_tables_.alm for each external tool. These requests are hidden from the user (not shown in list_requests or ? (summarize_requests) output). Each tool is invoked by a new or existing mbuild request with the name of an individual segment in the installation directory of the type supported by the tool.

For **archive_sort** and **peruse_crossref**, both long and short names of the tool are given in their multics_request statement; neither of these names collide with other mbuild request names.

```
multics_request  archive_sort,       " Hidden command
                 (as),
                 (Sort archive components into alphanumeric order.),
                 ,
                 flags.allow_command+
                 flags.dont_list+flags.dont_summarize


        request  xref,
                 mbuild_xref_$xref_request,
                 (),
                 (Lists source files that reference each include file.)

multics_request  peruse_crossref,    " Hidden command
                 (pcref),
                 (Prints information from the cross_reference command.),
                 ,
                 flags.allow_command+
                 flags.dont_list+flags.dont_summarize
```

For **verify_info**, only the long name is used in the multics_request statement. The short name of the tool will instead be used as the short name for the new **verify** request being added to mbuild, as shown below.

```
         request  verify,
                  mbuild_info_checks_$vi_request,
                  (vi),
                  (Check format of info segments.)

multics_request  verify_info,  " Hidden command, used to implement
                 (),           "  verify (vi) request.
                 (Checks format of info segments.),
                 ,
                 flags.allow_both+
                 flags.dont_list+flags.dont_summarize
```

Other mbuild Bug Repairs

Several mbuild bugs have been reported over the past several months. Most are minor in nature; a few prevent unusual installation operations from working correctly. The following repairs proposed for mbuild requests and support subroutines are listed by mbuild source module.

mbuild.pl1: Increment version of the mbuild subsystem (as reported by the . request) to mbuild_1.03; then to mbuild_1.04.

mbuild_Tlist_.pl1: For entry point:

> **mbuild_Tlist_$request_Seg_match_star**: match the star name against added names on the segment only for input structures of type Seg and UNBOUNDOBJ. COMPILE and BOUNDOBJ and other structures do not have added names specified in the build script.
>
> **mbuild_Tlist_$find_BOUNDOBJ**, **$find_COMPILE**, **$find_Seg**, and **$find_UNBOUNDOBJ**: add a parameter to these entry point calling sequences, a pointer to a previous return value. If non-null, the search for a match begins with the thread item following that previous return value. This supports searching a threaded list for multiple items with the same name.

mbuild_Tlist_.incl.pl1: Change the introductory comment to reference correct names for the Tlist_insert_before and Tlist_insert_after functions declared in this include file.

mbuild_data_.pl1: For entry point:

> **mbuild_data_$get_build_progress**: fix bug in gathering/reporting progress toward creation of the installation exec_com.
>
> **mbuild_data_$get_Seg**: distinguish a Seg() for a DELETE operation from an existing Seg created for an ADD or REPLACE operation. Also, distinguish a Seg() which is an archive component in operations which DELETE the Seg() from one bound segment archive, and ADD it to another bound segment archive. The repair for the preceding issue requires adding a new archive_name parameter to the $get_Seg calling sequence. All callers of this entry point must be updated to use this new parameter.
>
> **mbuild_data_$scan_Tb_insert**: allow insertion into a thread for Seg structures having the same Seg.name value (ticket #213). Also, change algorithm to insert aSeg() being DELETEd before other Segs of same type being REPLACEd or ADDed.

mbuild_data_.incl.pl1: Declare several new constants used in the new mbuild_data_$get_Seg calling sequence. These are used in many of the mbuild source modules.

mbuild_analyze_.pl1: Remove a declaration of mbuild_Tlist_$find_Seg; the above change in calling sequence made this unused declaration incorrect. **az_source** (internal procedure): when looking for the BOUNDOBJ containing an object whose source is being DELETEd, map its DELETE operation value into a REPLACE operation for the searched-for BOUNDOBJ. **add_source_arch_components** (internal procedure): change the call to mbuild_data_$get_Seg to specify the containing archive name when looking for a Seg(source) known to be in an archive. That helps distinguish multiple Seg(source) having same name but being moved from one bound segment to another.

mbuild_archive_.pl1: When updating archives, do DELETEs before ADD and REPLACE operations; space in the archive freed by the deletion may be needed to hold REPLACEd components growing in size, and new components being ADDed. Change all mbuild_data_$get_Seg calls to use the new archive_name parameter. **sort_archives** (internal procedure): added to support sorting of all archives after they have been updated (described with the **archive_prep** request changes above); this includes a new -no_sort control argument for the request.

mbuild_clean_.pl1: Remove unused declaration for mbuild_Tlist_$find_request. Update calling sequence for mbuild_Tlist_$find_Seg.

mbuild_help_.pl1: Tell help_ to aggregate info seg paragraphs to completely fill terminal page length whenever possible. Change name of default info block (chosen when help request has no TOPIC argument) to select the `summary.topic` block. Call check_star_name_ to validate TOPIC argument; but special-case 1-letter TOPIC argument so info blocks describing the subsystem_version (.) and summarize_requests (?) request can be entered. Change order for locating info segment blocks. New order would: match TOPIC starname against block names in mbuild.info; match ssu.TOPIC.info value against info segs at >doc>subsystem>ssu.TOPIC.info (no starnames allowed); match TOPIC.info found using info_segment search paths (no starnames allowed). The third search location is needed to display help for external tools called by mbuild requests (e.g., history_comment, peruse_crossref, archive, library_fetch, etc.).

mbuild_history_.pl1: Remove segments being DELETEd from list of installation directory segments that can be passed to the history_comment command. A Seg.operation = "DELETE" item has no physical segment in the installation directory, so there is nothing to pass to the history_comment command.

mbuild_install_.pl1: Fix error message reporting that a library name matches several paths. When installing a Seg following the "target_only" paradigm, use the default library associated with that paradigm if the build script does not specify a particular library.directory value. Follow the revised mbuild_data_$get_Seg calling sequence.

mbuild_library_.pl1: For entry point:

> **mbuild_library_$locate**: change code to fill-in Seg.archive_name only if it has a null-string value in the located Seg structure. This change is needed to support moving an archive component from one bound segment to a different bound segment.

mbuild_print_.pl1: Use revised calling sequence for mbuild_Tlist_$find_(BOUNDOBJ COMPILE Seg UNBOUNDOBJ) structures to print items in a threaded list having the same Seg.name but residing in a different bound segment, or being DELETEd from one library and moved to a different library.

mbuild_request_parms_.pl1: update to know about new source modules being added to mbuild: mbuild_info_checks_.pl1 (containing the new **verify** request) and mbuild_xref_.pl1 (containing the new **xref** request).

mbuild_scan_.pl1: Update for the new calling sequences for mbuild_Tlist_$find_Seg, mbuild_data_$get_Seg, and mbuild_data_$scan_Tb_insert.

mbuild_script_.pl1: Add **count_lines** (internal procedure) to count the number of lines in any build script description field (which is not passed to mbuild_script_parse_). Include this line count in a new argument passed to mbuild_script_parse_, so error messages printed by that routine can reference the correct line number in the build script file. Update to use the revised mbuild_data_$get_Seg calling sequence. When mbuild_script_parse_ returns with a nonzero status code, add a blank line between any messages generated by mbuild_script_parse_ and the error message displayed by mbuild_script_.

mbuild_script_parse_.rd: Add a new Alines_preceding_script parameter so line number given in error messages reflects the actual build script line on which the errant statement resides. Follow new calling sequence for mbuild_data_$scan_Tb_insert so files to be DELETEd appear in proper location in the Seg threaded list. Add an unknown_entryname error to diagnose a DELETE operation for a Seg not found in the library. Fix token displayed in error messages for bad_library and bad_path errors (Ticket #219).

mbuild_set_.pl1: Edit introductory comment to correct typo. Actual code is not changed.

Build Scripts:

The changes described in this MCR will be installed in two stages. The first stage installs all changes except for the new mbuild verify request; that request depends on the MTB-1004 changes for verify_info.

The second stage includes changes to implement the new verify request. This stage will be installed after the changes in MTB-1004 are approved and installed.

The stage one build script is shown below:

```
Installation_directory:  >udd>m>gd>w>mb03;

 Build_script: mb03.mb;

 Bound_obj:    bound_mbuild_                IN: tools  UPDATE;
     bindfile: bound_mbuild_.bind            REPLACE;
     source:   mbuild.pl1                    REPLACE  compiler: pl1 -ot;
     source:   mbuild_Tlist_.pl1             REPLACE  compiler: pl1 -ot;
     source:   mbuild_analyze_.pl1           REPLACE  compiler: pl1 -ot;
     source:   mbuild_archive_.pl1           REPLACE  compiler: pl1 -ot;
     source:   mbuild_clean_.pl1             REPLACE  compiler: pl1 -ot;
     source:   mbuild_compile_.pl1           REPLACE  compiler: pl1 -ot;
     source:   mbuild_data_.pl1              REPLACE  compiler: pl1 -ot;
     source:   mbuild_help_.pl1              REPLACE  compiler: pl1 -ot;
     source:   mbuild_history_.pl1           REPLACE  compiler: pl1 -ot;
     source:   mbuild_install_.pl1           REPLACE  compiler: pl1 -ot;
     source:   mbuild_library_.pl1           REPLACE  compiler: pl1 -ot;
     source:   mbuild_print_.pl1             REPLACE  compiler: pl1 -ot;
     source:   mbuild_request_parms_.pl1     REPLACE  compiler: pl1 -ot;
     source:   mbuild_request_tables_.alm    REPLACE  compiler: alm;
     source:   mbuild_scan_.pl1              REPLACE  compiler: pl1 -ot;
     source:   mbuild_script_.pl1            REPLACE  compiler: pl1 -ot;
     source:   mbuild_script_parse_.rd       REPLACE  compiler: rdc -ot;
     source:   mbuild_set_.pl1               REPLACE  compiler: pl1 -ot;
     source:   mbuild_xref_.pl1              ADD      compiler: pl1 -ot;

 Include:      mbuild_Tlist_.incl.pl1        REPLACE IN: incl;
 Include:      mbuild_data_.incl.pl1         REPLACE IN: incl;

 Info:         mbuild.info                 IN: priv.info  REPLACE;
                 add_name:
                   mb.info
                   build_script.gi.info;
```

The stage two build script is shown below:

```
Installation_directory:  >udd>m>gd>w>mb04;

 Build_script:  mb04.mb;

 Bound_obj:     bound_mbuild_                IN: tools  UPDATE;
     bindfile:  bound_mbuild_.bind            REPLACE;
     source:    mbuild.pl1                    REPLACE  compiler: pl1 -ot;
     source:    mbuild_info_checks_.pl1       ADD      compiler: pl1 -ot;
     source:    mbuild_request_parms_.pl1     REPLACE  compiler: pl1 -ot;
     source:    mbuild_request_tables_.alm    REPLACE  compiler: alm;

 Info:          mbuild.info                  IN: priv.info  REPLACE;
                  add_name:
                    mb.info
                    build_script.gi.info;
```

Documentation

The following documentation describes the new control argument for the **archive_prep** request. Changed or new documentation is shown in **bold** font.

```
2021-02-14  archive_prep, arch

Syntax:  archive_prep {-control_arg}
         arch         {-control_arg}

Function: updates bound segment source/object archives.  Sorts updated
archive components into alphanumeric order within their archive.
Binds object archives to produce a bound segment.

Control arguments:
-list, -ls
   produces a listing segment whose name is derived from the name of
   the bound object segment plus a suffix of list.  The listing segment
   is generated to dprint; it contains the bound segment's bind control
   segment (see "Notes on bindfile"), its bind map, and that
   information from the bound object segment printed by the
   print_link_info command.  You can't invoke -list with -map.  In the
   absence of -list or -map, no listing segment is generated.
-map
   produces a listing segment (with the suffixes list and map) that
   contains only the bind map information.  It is incompatible with
   -list.  In the absence of -list or -map, no listing segment is
   generated.
-no_sort, -ns
   Most bind files have an Order statement controlling order in which
   object components appear in the output bound segment produced by
   the bind command.  For bound segments having no bind file or a bind
   file not using the Order statement, the bind command sequences
   object components by their order of appearance in input archives.
   Therefore, -no_sort should be used to suppress sorting of archives
   for such bound segments.
```

The following documentation describes the new **xref** request.

```
2021-02-15  xref

Syntax:  xref {SEG_NAME} {-pcref_control_args}

Function: displays crossref items (source files) that reference
include files found in the installation directory.

Arguments:
SEG_NAME
   name the segment(s) to check.  The .incl.* suffix (or a more
   specific suffix like .incl.pl1 or .incl.alm) must be given.
   The star convention may be used to select several segments.
   If SEG_NAME is not given, all include segments are checked.
```

```
Control arguments:
   Any control argument accepted by the peruse_crossref (pcref)
   command may be given.  For a full list of controls, type:
      help pcref -brief
```

The following documentation describes the new **verify** request.

```
2021-02-14  verify, vi

Syntax:  vi {SEG_NAME} {-verify_info_control_args}
         vi -rules AREA

Function: checks format of info segments; or display rules and
guidelines for info segment formatting and section titles.

Arguments:
SEG_NAME
   name the segment(s) to check.  The .info suffix must be given.
   The star convention may be used to select several segments.
   If SEG_NAME is not given, all info segments are checked.
   No SEG_NAME is permitted if -rules is given.

Control arguments:
   Any control argument accepted by verify_info (vi) command may
   be given.  Commonly-used controls are described below.  For a
   full list of controls, type:  help verify_info -brief
-names, -nm
  add names required by guidelines if missing from an info segment.
  Reorder names following guidelines.  If names exist that conflict
  with guidelines, ask user for permission to remove those names.
  If removal not granted, move conflicting names to end of name list.
  Name changes are listed in informational messages.

-no_names, -nnm
  suppress name additions, removals, and reordering.  Error messages
  report names that are missing, conflict with guidelines, or need to
  be reordered.  (default)
-force_names, -fnm
  change names to follow all guidelines without user queries.
-rules {AREA}
  displays verify_info guidelines and rules for info segment structure
  and format.  See "List of rule areas" below.

List of rule areas:
   The -rules AREA operand may be any of the following.
all, a
  displays all rules and guidelines used by verify_info. (default)
file, f
  displays guidelines for supported file structures (allowed order of
  :Info: :[Info]: :Entry: and :hcom: block divider lines within an
  info segment).
```

```
block, bk
  displays rules used to determine the kind of each info block
  (e.g., command or active function, subroutine, subsystem request,
  etc.).

section, scn
  displays Multics standard info segment section titles, and a list of
  obsolete or deprecated titles with a preferred replacement.
KIND_OF_INFO_BLOCK
  displays section titles typical for a particular kind of info block
  including the usual order of appearance within the block of each
  title.  See "List of info block kinds" below.

List of info block kinds:
  The -rules KIND_OF_INFO_BLOCK value may be any of the following.
all_kinds, ak
  display section titles typical of all of the following block kinds.
command, cmd
  display section titles used in info blocks describing commands
  and/or active functions.
general_info, gi
  display section titles used in info blocks describing general
  information.

subsystem, ss
  display section titles used in info blocks describing subsystem
  requests and/or active requests; subsystem request summary; or other
  subsystem topic.
request, req
  display section titles used in info blocks describing subsystem
  requests and/or active requests.

topics, topic
  display section titles used in info blocks describing subsystem
  topics (information not a request description or summary of
  requests).
subroutine, subr
  display section titles used in info blocks describing a subroutine
  or function.  This includes a subroutine introduction block, and
  blocks describing a subroutine entry point.
io_module, io
  display section titles used in info blocks describing an I/O module;
  or one of its setup operations (open_file, close_file, or detach);
  or one of its control orders.
```

Additional changes were made to mbuild.info including:

- Correct errors found by verify_info.
- Document the quit (q), ?, and . requests which were omitted from earlier versions of this info segment. That documentation is nearly identical to the >doc>subsystem documentation for these standard ssu_ requests.

Testing

These changes to mbuild code have been tested by: preparing multiple versions of bound_mbuild_ containing the changes; by preparing multiple version of bound_info_rtns_ containing verify_info, info_seg_, and revised help command and help_ subsystem.

A test version of the changes was also made available to Eric Swenson for use in other software updates on the GHM system.

Finally, individual changes to the software were fully tested to prove that the reported symptoms were actually remedied by each change. Most of the reported symptoms were not documented as Multics Tickets. So each problem was briefly described in the history comment for the source containing the main repair of the bug; and these bugs were summarized in the build script descriptions. For reasons of brevity, those build script description fields are not shown in this MCR, but will be available to the code auditor.

Version History

Date	Revision	Author	Comment
2021-01-07	0.1	Gary Dixon	Pre-release versions of document
2021-02-05	1.0	Gary Dixon	Preliminary review copy of document.
2021-02-20	1.1	Gary Dixon	Add repair for history (hcom) request bug relating to DELETEd Seg items.