Subject: MCR10126: mbuild Version 2.00

Author: Gary Dixon

Date: 2022-09-12

Multics Ticket: http://multics-trac.swenson.org/ticket/290#

Multics Ticket #290 identifies several improvements suggested for the mbuild user interface and runtime environment.

mbuild needs to be changed as follows.

- No longer store name of build script, or pathname of build directory in the build script segment.
- Track across mbuild invocations:
    - progress through major phases of the build: developing, auditing, installing
        - Use start-up requests tailored to needs of the current phase.
    - library descriptor associated with each build directory.
    - update_seg log directory associated with each build directory. This is often tied to the library descriptor, but not always.
    - MCR number and installation ID associated with the build directory.
    - names of developer, auditor, installer in the build script file.
    - broader development status (detailed notes) in a new section of the build script.
- Each mbuild invocation needs to:
    - monitor changes in the build environment occurring while mbuild is running (working directory, translator search paths, library descriptor, and updates to the build script segment done via an external editor).

This wish list stems from conversations with Eric Swenson about typical problems faced by mbuild auditors and installers.

PROBLEM A: It is too difficult to move a build directory from one location in the file system to another location, or to simply rename a build script. Difficulties arise because:

- Name of the build script is tied to shortest name on the build directory created by the developer. However, the auditor or installer may want to move each build directory to a new location (for safe-keeping, or to collect audit work or installs in one location for ease of reference), and may want to impose a different naming convention on the build directory and on its contained build script segment.
- Pathname of the build directory is embedded in the build script Installation_directory: statement.

- Entryname of the build script is embedded in Build_script: statements contained inside the build script segment.
- Entryname of the install exec_com (created by the mbuild install_ec request) also shares this naming convention, so again the name on the build directory gets tied to name of the install exec_com in a Build_exec_com: statement.
- Entryname of the build script is also tied to the update_seg installation object (.io) and installation log (.il) segments created by the build exec_com. These get embedded in the build script as Build_io: and Build_log: statements.

When a build directory is renamed or moved, much renaming of dir, build script, and items in build script statements is required. This extra work is undesirable.

PROBLEM B: As development, auditing, and installing operations proceed across many weeks of time, mbuild provides no place for storing details pertinent to a particular build directory, or its required build environment, etc. The developer and auditor are forced to remember these details across many invocations of mbuild, and sometimes must remember to issue several environment setup commands prior when switching from one development effort to another. Setup details include the following:

- Change to build directory containing the new or revised source segments.
- Set appropriate library descriptor describing target library for the changeset.
- Ensure location of include files for that target library appears in “translator” search paths.

This information needs to be documented as properties of the build so it can be easily communicated between developer(s), auditor(s), and the installer. Proper setup needs to happen automatically when mbuild is invoked by a user, based upon these recorded properties.

PROBLEM C: While the mbuild subsystem is running, it is often necessary to escape out of the mbuild environment: if development or auditing are interrupted by unrelated issues; or if the developer or auditor must change from the build directory to another working directory to run tests, or investigate related code. However, correct mbuild operation depends upon working directly in the build directory as mbuild requests are issued; having an unchanging library descriptor and unchanging “translator” search paths setup for that build.

mbuild should check after each request line is processed whether it remains in the required operating environment.

PROBLEM D: While the mbuild subsystem is running, it is often necessary to escape to an external editor to make additions or corrections to the build script segment. Resolving the problems above will require adding additional types of data to that script, thereby increasing frequency of external editing. After each editing excursion, the user must remember to read those changes back into mbuild's internal data so mbuild remains informed of the changes to its build instructions and environmental setup requirements.

mbuild's post-request line checks should look for changes to build script content, and automatically read modifications sections of the build script into mbuild's internal environment. But reading changes to Build Script Language statements must be delayed while compile, archiving and binding operations are progressing because changes to these statements (or to the library descriptor identifying the target library named in those statements) may require re-scan of the build directory (looking for additional segments to be changed), as well as re-reading the changed Build Script Language statements. And the revised data must then be re-analyzed before resuming compile, archive and bind operations.

Therefore, mbuild's post-request line checks should automatically read sections of the build script not affecting segments being changed or their target libraries; and remind the user to manually read in changes impacting build operations when appropriate to get updates to the original-content segment list.

PROBLEM E: Large changes often involve mods or additions to many source modules in one or more bound segments, changes/additions to include files, changes or additions to info segments, etc. While the Build Script Language statements track attachment of such segments to the build effort, the developer has no place to record goals of the effort, or progress in design, coding, documenting, or testing tasks required to integrate these tasks into a well-implemented development project.

The mbuild build script should include a free-format section in which such status information can be managed by the developer, communicated to auditor and installer, and captured as part of the installation record for development project.

Proposed Changes

This MCR proposes changes to the user interface of the mbuild command, to the build script segment format and contents, and to mbuild requests that deal with contents of the build script. It also includes a small bug fix to the translator_temp_ subsystem. These changes are detailed in the subsections below.

Build Script Segment

The current build script segment begins with an optional description section followed by required statements in the Build Script Language. If present, the optional description begins with the keyword **Description:** followed by one or more free-format lines of text. Any description section ends with the first statement in Build Script Language, which is always:

Installation_directory: <pathname of build directory>;

This MCR proposes a new build script segment layout with two new sections added to the current Description and Build Script Language areas. Each of the four sections is always present in every build script, and each begins with a section title line (shown in **bold** font below).

Description -- a section containing free-form lines giving a brief English-language description of the change at a history_comment level of detail. This replaces the Description: section of current scripts.

Changes -- a section containing Build Script Language statements describing segments being REPLACEd, ADDed, or DELETEd as part of the build operation. The Changes -- section replaces the Build Script Language portion of current scripts.

Information -- a new section containing statements in a syntax:

keyword: value;

giving property values for the build operation. These are retained in the build script across invocations of mbuild, and used by the mbuild to setup the build environment and set defaults used by the hcom request. mbuild provides requests for setting the value of each property. The properties to be supported are listed in the Information -- discussion (see below).

Status -- a new section containing free-form lines which the developer can use to state development goals, and track progress in design, coding, documenting, testing of modules being replaced, added or deleted, etc. mbuild provides a template suggesting the kind of data that might be provided. However, each developer, auditor, or installer can edit lines in this section and format, expand or remove data as needed.

Each of the sections above begins with a line containing only the section title separated by whitespace from two consecutive dash (-) characters; the title starts in column 1 of that line. Lines of the section continue until the next section title line is encountered, or end-of-segment is reached.

The user can replace the entire **Description --** or **Status --** section using the mbuild `set` request, or append lines to the existing section using a new mbuild `add` request. Or those sections may be edited outside of mbuild. mbuild will watch for changes made externally to those sections, and automatically read them back into mbuild's internal data. The mbuild `print -description` and `print -status` requests will display contents of these sections.

The Build Script Language statements of the new **Changes --** section are usually constructed initially from results of mbuild `scan` and `analyze` requests. These statements describe each original-content segment found by the `scan` request. mbuild tries to match each segment found by the scan request with an equivalent segment in library directories identified by the current library descriptor. It uses this match data to guess whether each segment is being ADDed or REPLACEd, to place the segment as a member of a bound segment or locate it in a specific library directory, etc.

The developer uses an external editor to correct mbuild's guesses at operation, containing bound object, target library directory, etc.

When exiting the editor and returning to mbuild, it detects editing of Changes -- section statements, and reminds the developer to issue an mbuild `read -changes` request to bring those revisions into mbuild's internal data. That can only be done by restarting all build operations by: issuing a `scan` request to look for new original-content segments added to the build directory; or issuing a `read -changes` request to discard scan's guesses and read Build Script Language statements corrected by the developer's editing.

Statements supported by the Build Script Language are unchanged by this MCR, except that the **Install_directory**: statement has been removed from the Build Script Language. Its information is no longer needed by mbuild nor wanted by the installer.

Finally, the **Information --** section records information about the following build properties.

Phase: records the development cycle phase which is currently underway. One of the following values must be given.

developing
changes are still being made to the build. (default)

auditing
changes are frozen (mostly) and code is now being audited.

installing
fully-audited changes are now being installed.

Descriptor: records pathname of the library descriptor describing library directories targeted by this build. (default: pathname of the process's current library descriptor when mbuild is first invoked for a given build directory)

LogDir: records pathname of the log directory to be used in the update_seg initiate command which created by the mbuild `install_ec` request. (default: result of the active function: [library_descriptor pathname *.log])

Approve_IDs: zero, one or more MCRnnnnn identifiers (or other Approve_ID values accepted by the history_comment operations). If only one identifier is set, it will be used as the default operand in any history_comment operation that accepts the -approve control argument. This includes the operations:

```
hcom add   …   -approve  { APPROVE_ID }
hcom add_field   … -apv  { APPROVE_ID }
hcom install … -approve  { APPROVE_ID }
hcom replace_field COMMENT_SPECS … -approve { APPROVE_ID }
```

Install_ID: records an MRnn.mm-cccc Multics Release installation identifier (e.g., MR12.8-1215). It will be used by the installer as default operand in any history_comment operation that accepts the -install control argument. This includes the operations:

```
hcom add   …   -install { INSTALL_ID }
hcom add_field   …  -in { INSTALL_ID }
hcom install … -install { INSTALL_ID }
hcom replace_field COMMENT_SPECS … -in { INSTALL_ID }
```

Developers:

Auditors:

Installers: records the user_ID (person_name.project_name) of the person using mbuild when the build script segment was created or is updated by an mbuild request (or automatic save of the build script). That user_ID is added to the property selected by the current Phase value (if not already present in that list).

These property values can be set by editing the build script **Information --** section. Some can also be set by using a Multics command, or by one or more mbuild requests as shown in the table below.

TO SET	USE THE COMMAND	USE mbuild REQUEST

Phase	`mbuild -developing`
	`mbuild -auditing`
	`mbuild -installing`
		`set -phase developing`
		`set -phase auditing`
		`set =phase installing`

Descriptor	`lds set DESCRIPTOR_NAME`	`lds set DESCRIPTOR_NAME`
LogDir	`mbuild -set_log_dir DIR_PATH`	`set -log_dir DIR_PATH`
Approve_IDs	`mbuild -approve APPROVE_ID`	`set -approve APPROVE_IDs` `add -approve APPROVE_IDs`
Install_ID	`mbuild -install INSTALL_ID`	`set -install INSTALL_ID`

Current value of all properties will be displayed using the mbuild request: `print -info`

The changes proposed by this MCR are described by the new-style build script shown below. This illustrates a typical build script, and documents the changes proposed by the MCR.

```
mbuild:  print -script

Description --
 A) Update for mbuild_data_version_4 changes to mbuild_data_.incl.pl1 to
    convert mbuild -debug (hidden control_arg) from being storing in
    bld.pad character string to bld.switches.debugS bit string.
 B) Update mbuild_data_.incl.pl1 to include new script_status structure
    and script_info sub-structure.
 C) Update mbuild.pl1 for revised user interface.
 D) Modify mbuild_script_ to handle 4 different sections of build script:
     Description --
     Changes --
     Information --
     Status --
 E) Add mbuild_script_info_.rd to parse the new  Information --  section
    of the build script.
 F) Modify mbuild.info to document many changes in mbuild command and in
    its subsystem requests.
 G) Add mbuild_monitor_.pl1 to watch for changes to mbuild runtime
    environment and build script contents.
 H) Fix translator_temp_$empty_all_segments to zero code argument before
    work starts to ensure it gets set.  (Unreported bug found during
    mbuild testing.)



Changes --


 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_compare_.pl1               REPLACE  compiler: pl1 -ot;
   source:     mbuild_compile_.pl1               REPLACE  compiler: pl1 -ot;
   source:     mbuild_data_.pl1                  REPLACE  compiler: pl1 -ot;
   source:     mbuild_display_.pl1               REPLACE  compiler: pl1 -ot;
   source:     mbuild_et_.alm                    REPLACE  compiler: alm;
   source:     mbuild_history_.pl1               REPLACE  compiler: pl1 -ot;
   source:     mbuild_info_checks_.pl1           REPLACE  compiler: pl1 -ot;
   source:     mbuild_install_.pl1               REPLACE  compiler: pl1 -ot;
   source:     mbuild_lib_names_.pl1             REPLACE  compiler: pl1 -ot;
   source:     mbuild_library_.pl1               REPLACE  compiler: pl1 -ot;
   source:     mbuild_monitor_.pl1               ADD      compiler: pl1 -ot;
   source:     mbuild_print_.pl1                 REPLACE  compiler: pl1 -ot;
   source:     mbuild_progress_.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_info_.rd            ADD      compiler: rdc
                                                           -ot -trace off;
```

```
   source:     mbuild_script_parse_.rd           REPLACE  compiler: rdc
                                                           -ot -trace off;
   source:     mbuild_set_.pl1                   REPLACE  compiler: pl1 -ot;
   source:     mbuild_xref_.pl1                  REPLACE  compiler: pl1 -ot;

 Bound_obj:    bound_proj_admin_                 IN: sss  UPDATE;
   source:     translator_temp_.pl1              REPLACE  compiler: pl1 -ot;

 Include:      mbuild_data_.incl.pl1             IN: sss.include  REPLACE;
 Include:      mbuild_request_parms_.incl.pl1    IN: sss.include  REPLACE;

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

Information --

 Phase:             developing;
 Descriptor:        >t>multics_libraries_;
 LogDir:            -library *.log;

 Approve_IDs:       MCR10126;
 Install_ID:        ;

 Developers:        GDixon.Multics;
 Auditors:          ;
 Installers:        ;


Status --
 -- Goals, Bugs and Suggestions

    Ticket #290:  mbuild 2.00 Improvements

    mbuild needs to be changed to:
     - No longer store name of build script, or pathname of build directory
       in the build script segment.

     - Track library descriptor associated with each build directory.

     - Track update_seg log directory associated with each build directory.
       This is often tied to the library descriptor, but not always.

     - Track progress through major phases of the build:
                 developing, auditing, installing
        - Use start-up requests tailored to needs of the current phase.

     - Track names of developer, auditor, installer in the build script file.

     - Track MCR number and installation ID associated with the build
       directory.
```

```
    - Track broader development status (detailed notes) in a new section of
      the build script.

    - Monitor changes in the build environment occurring while mbuild is
      running:
       - working directory
       - translator search paths
       - library descriptor
       - updates to the build script segment done via an external editor


   AREAS                                PROGRESS
-----------------------------------   --------------

-- Design

 mbuild.pl1                           complete
  - use cases for new user interface    done
  - setup mbuild_monitor_               done
 mbuild_monitor_.pl1                  complete
  - new subsystem to monitor changes
    in mbuild's runtime environmental
    dependencies.
 mbuild_script_.pl1                   complete
  - read split into a step for each     done
    section of build script
  - save will completely replace        done
    .mb content whenever any section
    is modified.
 mbuild_script_info_.rd               complete
  - routine to parse  Information --
    section of build script
 mbuild_script_parse_.rd              complete
  - modify to handle different
    divider for  Changes --  section
 mbuild_print_.pl1                    complete
  - print -save_format
    will generate content
    of .mb stored by save
    request
  - will have new -ctls to display     complete
    each .mb section, or all sections
 mbuild_set_.pl1                      complete
  - set request will:                   done
    - have -ctls to:
      - replace content of
        Description & Status
        sections.
      - replace attributes
        of Information section.
  - add request (new) will add lines    done
    to sections: Description and
    Status; add array elements to
    Approve_ID, Developers, Auditors,
    Installers
```

```
-- Coding

 ... major changes summarized in the
     Design subsection (see above)
 mbuild.pl1                              complete
 mbuild_monitor_.pl1                     complete
 mbuild_script_.pl1                      complete
  - read request                          done
  - save request                          done
 mbuild_script_info_.rd                  complete
 mbuild_set_.pl1                         complete
 mbuild_print_.pl1                       complete

 ... smaller changes summarized here
 bound_mbuild_.bind                      complete
  - update for new objects
       mbuild_script_info_
 mbuild_analyze_.pl1                     complete
  - do "save" request if analysis
    completes successfully to keep
    build script synchronized with
    mbuild internal data.
 mbuild_clean_.pl1                       complete
  - support use of bld.debugS
 mbuild_data_.pl1                        complete
  - change $reset_build_lists to use
    translator_temp_$empty_all_segments
  - support use of bld.debugS
 mbuild_display_.pl1                     complete
  - narrow output line length
 mbuild_et_.alm                          complete
  - new error codes
 mbuild_history_.pl1                     complete
  - substitute script_info.install_ID
    and .approve_IDs(1) into hcom
    requests missing operand of
    -install and -approve
 mbuild_install_.pl1                     complete
  - call mbuild_set_$log_directory
    to get update_seg -log_dir value
 mbuild_scan_.pl1                        complete
  - add $segs_matching_starname entry
    point to search build dir for
    potential build script segments.
  - use first name on all build script
    segments as name given in Seg
    structures of mbuild internal data.
 mbuild_script_parse_.rd                 complete
 mbuild_request_parms_.incl.pl1          complete
  - add check_parms2 entry point.
 translator_temp_.pl1                    complete
  - always set code output arg in
    entry points added by earlier change.
```

```
 ... segs changed only to update
     mbuild_data_.incl.pl1
 mbuild_Tlist_.pl1                       complete
 mbuild_archive_.pl1                     complete
 mbuild_compare_.pl1                     complete
 mbuild_info_checks_.pl1                 complete
 mbuild_lib_names_.pl1                   complete
 mbuild_progress_.pl1                    complete
 mbuild_xref_.pl1                        complete


-- Testing

 test case design                        complete

 test case description & status          complete
  0: empty directory                      done
     - test creation of .mb in
       empty dir
     - then add 1 small include file
     - test: set -desc
     - test: emacs *.mb to change
       Status
  1: .mb w/o Changes                      done
     - test hcom af -approve
  2: use of different lib descriptor      done
     - test different phases
     - test hcom af -install
     - test hcom rpf *.pl1 last
         -in DIFFERENT_ID
  3: test monitored envir changes         done


-- Documentation
 mbuild.info                             complete
  mbuild:                                 done
  summary:                                done
  build_script.gi:                        done
  read:                                   done
  save:                                   done
  set:                                    done
  add:                                    done
  print:                                  done

-- History Comments
 being added to describe                 complete
  changes to each source module

-- MCR:
 MCR10126                                underway

-- Auditing
 auditor: Eric Swenson                   planned

-- Installing
```

Command Interface: mbuild, mb

An important goal of this change proposal is to de-couple name(s) on the build script segment from names on the build directory (the directory that contains the build script and the original-content sources to be built). A BUILD_SCRIPT argument must therefore be added to the mbuild interface to allow the user to tell mbuild: where the build directory resides in the file system, and name of the build script segment within that directory.

Also, control arguments are needed to set properties of the build operation, including: phase, Approve_ID, and Install_ID. The mbuild command already has a -set_log_dir control argument to set the LogDir property. And since mbuild uses the library descriptor selected by the user process, the mbuild command can call mbuild_library_$get_descriptor_path to get the process' selected descriptor pathname.

The following documentation for the mbuild command more fully describes the improved user interface. New arguments are highlighted in bold font.

```
help mbuild.info -all

>udd>m>gd>w>mb>mbuild.info   (100 lines in info)

2022-09-07  mbuild, mb

Syntax as a command:  mb {BUILD_SCRIPT} {-control_args}

Function: A subsystem of requests which build and prepare to install
all original-content segments in a build directory.  Original-content
segments include: program source files, include files, info segments,
bind files, source archive(s) for a new bound segment, etc.

The build directory contains a build script segment giving details of
the build.  For information about contents of the build script, type:
   help build_script.gi

Arguments:
BUILD_SCRIPT
   pathname of the build script segment describing the changes to be
   built.  A suffix of .mb is assumed if not given in the final
   entryname.  If the directory part of this pathname does not
   reference the working directory, then the working directory is
   changed to the specified directory as the mbuild subsystem opens.
   mbuild will prompt to create this build script if it is not found.
   The star convention is allowed.  (default: search for **.mb in the
   working directory: if one match is found, that is the build script;
   otherwise the user is asked to re-invoke mbuild giving a specific
   build script pathname.)
```

```
Control arguments (phase):
   mbuild tracks a "phase of building" which specifies whether the
   user is "developing" a set of segments being changed/added, or
   "auditing" those changes, or "installing" them.  This phase is
   stored in the  Information --  section of the build script (.mb)
   segment.  One of the following control_args may be given to change
   this stored phase setting.
-developing, -dev
   changes are still being added to the build.  (default if no build
   script segment exists in the build directory.)
-auditing, -aud
   changes are frozen (mostly) and are now being audited.

-installing, -inst
   fully-audited changes are now being installed in the Multics
   Libraries.  This phase should be used only for installations into
   the Multics Libraries directories.  One or more Approve_ID values
   must be set, along with an Install_ID value.  These properties are
   appended to the  Description --  section as in the example below:
    (MCR10126; MR12.8-1516)

Control arguments (initial requests):
-scan_read, -scrd
   scans the installation directory looking for original-content
   segments before entering the mbuild request loop.  If a build
   script segment is found, asks the user if it should be read.
   (default for developing phase)
-read_analyze, -rdaz
   reads an existing build script segment and analyzes segments
   listed in its  Changes --  section to determine exact contents
   build.  Original content segments not referenced in the build
   script are ignored.  No scan of the directory is performed.
   (default for auditing/installing phases)

-scan, -sc
   perform a scan of the installation directory before entering the
   request loop.  Do not ask about, or actually read, any build script
   Changes --  section.
-no_scan, -nsc
   omit the scan request before entering the mbuild request loop.

-request STR, -rq STR
   executes STR as an mbuild request line before entering the
   request loop.
-request_loop, -rql
   enters the mbuild request loop. (default)
-no_request_loop, -nrql
   does not enter the request loop.

Control arguments (setup):
-approve MCRnnnnn,
-apv MCRnnnnn
   adds an MCR number for this build directory.  If only one value is
   associated with this build, that value will be used the APPROVE_ID
   is omitted from the request:  hcom add_field -approve APPROVE_ID
```

```
-install INSTALL_ID,
-in INSTALL_ID
   associates a Multics Release INSTALL_ID (e.g., MR12.8-1012) with
   the build script.  That value will be used if the INSTALL_ID is
   omitted from the request:  hcom add_field -install INSTALL_ID

-set_log_dir DIRNAME, -sld DIRNAME
   sets the log directory used by the update_seg initiate command.
   (default is to use the first directory returned by active function:
   [lds pn *.log])

-abbrev, -ab
   enables abbreviation expansion of request lines.
-profile path, -pf path
   specifies the pathname of the profile to use for abbreviation
   expansion.  The suffix "profile" is added if necessary.  This
   control argument implies -abbrev.
-no_abbrev, -nab
   does not enable abbreviation expansion of request lines.  (default)

-prompt STR, -pmt STR
   sets the request loop prompt to STR.  The default is:
         ^/mbuild^[ (^d)^]:^2x
-no_prompt, -npmt
   suppresses the prompt for request lines
```

mbuild creates a template build script if the given BUILD_SCRIPT does not exist; or reads all sections of any existing build script into internal memory to obtain: current contents of the free-form sections (Description -- and Status --); and setup properties from the Information -- section.

mbuild delays reading the Changes -- section until requested by the user, since the user may want to re-scan the build directory for just-added original-content sources, and then have mbuild create Build Language Statements for all sources to replace current contents of the Changes -- section.

Therefore, while the mbuild subsystem is running, there is always a known build script segment in the build directory. mbuild automatically saves its internal data in that script when needed, and monitors that script so changes made by via an editor are immediately seen in mbuild's internal data (e.g., by the mbuild `print` request).

mbuild monitor_

This MCR proposes to have the mbuild subsystem monitor three important features of its runtime environment, as well as watching for changes to the build script segment.

First, mbuild expects that all build operations are performed with the process working directory set to the build directory. So the monitor ensures this expectation is met by automatically changing the working directory back to the build directory when necessary. The user is notified when such changes are made.

Second, mbuild expects that the library descriptor selected by the process is the one specified for the build operation in the build script. When the build script is created, the process' selected library descriptor is recorded in the script. Later, if the user issues a `library_descriptor set DESCRIPTOR` request (or equivalent Multics command) while mbuild is running, the mbuild check notices the different descriptor selected for the process. It asks the user if that new descriptor should be used for mbuild operations.

- If the user answers "yes", then mbuild:
  - records the path of this new descriptor in the build script;
  - uses the clean request to remove all derived objects from the build directory;
  - empties mbuild's internal data store (to remove associations of original-content sources with segments in the previously-selected libraries);
  - tells the user to do either a scan or read request (or both) to rebuild the list of segments being changed by making associations between those segments and the just-selected libraries in the new descriptor.
- If the user answers "no", then mbuild resets the process' library descriptor back to the pathname specified in the build script.

Third, mbuild expects that changed include files in the build directory will be used when compiling original-content sources in the build. Since all compiles are done with the working_directory set to the build directory, this expectation is met only if the "translator" search paths for the process have a first path specifying the -working_dir rule. If the mbuild check finds a different value for first path, it requests the user to correct this deficiency in the "translator" search paths.

Fourth, mbuild checks for modifications to the build script segment, as described in the prior section of this MCR.

The checks are performed by a new mbuild_monitor_ subroutine enabled by the mbuild command. mbuild defines its expectations when invoking the mbuild_monitor_$setup subroutine. The final setup step tells ssu_ to use the monitor_ subroutine as its post-request-line replaceable procedure. This causes the ssu_ request loop to call the monitor_$watch procedure after executing each request line. Therefore, the monitor_ checks on runtime impact of each mbuild request or escaped ..MULTICS_COMMAND entered by the user.

If the monitor_ check finds any expectation is no longer met, it calls an mbuild-supplied co-routine to take action. That co-routine can tell the monitor_ to reset the environment back to the expected state, or to record the new state as the expectation for future checks. In this way, actions taken for runtime or build script changes are tailored to mbuild's needs.

- For the working directory check, the co-routine just tells monitor_ to change_wdir_ back to the build directory.

- For the library descriptor check, the co-routine asks the questions described above, and takes corrective actions based on user's answer: whether to return back to the expected descriptor, or to change the expected descriptor to be used for the build.
- For the "translator" search path check, the co-routine warns the user to correct those search paths so that the -working_dir rule appears first in the path list.
- For changes to contents of the build script segment, the co-routine reads in all sections of the revised build script except for the Changes -- section. If that section has a different length from the previous check, the co-routine asks the user to issue a `read -changes` request to bring any changed Build Script Language statements into mbuild's internal data when the user is ready for the build to include those changed statements.

mbuild Request: read, rd

Several improvements will be added to the mbuild `read` request. With two new sections being added to the build script segment, a new locate_sections internal procedure now splits build script lines into the four separate sections. Control arguments were added to permit each section to be read into mbuild's internal data as a separate operation; or they may all be loaded by a single read request.

A new mbuild_script_$read_all_sections_but_changes subroutine entry point was added to enable initial reading of the build script by the mbuild command, and to support on-going modification checks by the mbuild_monitor_ subsystem. This reads the Information -- and Description -- and Status -- sections into mbuild whenever the build segment's date-time-contents-modified changes. It also returns current length of the Changes -- section back to the monitor co-routine, so it can determine if that section has changed.

Finally, code was added to read data from an old-style build script sections into mbuild so that data can be re-saved in the new format.

Two sections of the new build script layout are free-form text lines that require no parsing beyond location of section contents: Description -- and Status -- sections.

The new Changes -- section will be parsed by the existing mbuild_script_parse_ subroutine since syntax of the actual Build Script Language statements was not modified.

See the Documentation section of this MCR for details of the extended user interface for the `read` request.

The new Information -- section needs a new subroutine to parse its contents; this subroutine is described in the next section of this MCR.

mbuild_script_info_

An mbuild_script_info_ subroutine was written to parse lines of the Information -- section into property values stored in a data structure. The subroutine uses the reduction_compiler language to parse tokens

found in the section into property name and value. The new subroutine is structured like the existing mbuild_script_parse_ subroutine which parses the Changes -- section.

mbuild Request: save, sv

Functions performed by the mbuild `save` request doubled to handle saving of two new sections of the build script. Complexity of the user interface was reduced by having the save code decide which sections were ready to save from mbuild internal data, and which sections of the existing build script should instead be preserved by the save operation.

It turns out that Description -- and Information -- and Status -- sections can be saved from internal data at any time because mbuild always has latest data from those sections in its internal store (thanks to the mbuild_monitor_ subsystem). Lines from the Changes -- section can be saved only after scan and/or read requests have been followed by an analyze request, because only at that point does mbuild have sufficient details about each segment to fill-in its Build Script Language statement. mbuild tracks the user's progress through steps of the build so it can make such decisions.

Therefore, each save request can generate a new copy of all four sections using eithier internal data, or by pre-saving the Changes -- section contents from the existing build script segment; then all four sections can be overwritten onto the existing build script segment.

Generating new section lines from internal data is done using the `print -save_format` request with output being redirected by the `file_output` request onto the build script segment. If build progress says the Changes -- section must be preserved from existing data rather than generated, lines of that section get copied to a temporary segment and then copied back into the build script segment by the save operation.

Note that mbuild `set` and `add` requests will perform an automatic `save` request if no errors occur while changing the data. Successful completion of an `analyze` request will also do an automatic `save` request to capture the revised Changes -- section data in the build script. Therefore, there is seldom any need for the user to manually issue a `save` request.

mbuild Request: print, pr, p

Functions of the mbuild `print` request were added to handle display of the two new sections of the build script segment, and to display several new information items: the Phase property and the build directory pathname. Its existing code structure was easily expanded to handle those new functions.

See the Documentation section of this MCR for details of the extended user interface of the `print` request.

mbuild_display_

The `print` request calls the mbuild_display_ subroutine to display information about segments of the build operation as either: Build Script Language statements (analyzed format); or as incomplete data (just-scanned format). This subroutine was changed to produce Build Script Language statements that fit in an

80-column line length format to make it easier to include the Changes -- section in an MTB or MCR document.

See results of that shorter-line format in the sample build script included in the Build Script Segment section of this MCR (see above).

mbuild Request: set

Functions added to the `set` request permit replacing individual property values maintained in the Information -- section of the build script, as well as replacing the Status -- section with a new set of lines. Existing functions for replacing the Description -- section and setting -library LIB.DIR values for segments of the build continue to work as usual.

Each successful set request performs an automatic `save` request to update the build script segment with the new or changed information.

See the Documentation section of this MCR for details of the extended user interface.

mbuild Request: add

It is often desirable to append new lines to an existing Description -- or Status -- sections. A new `add` request provides this functionality. In addition, several of the properties in the Information -- section support an array of values. The add request enables adding new elements to those array properties.

Each successful add request performs an automatic `save` request to update the build script segment with the new information.

See the Documentation section of this MCR for details of the extended user interface.

mbuild Request: history, hcom

The mbuild `hcom` request will now use a single APPROVE_ID or INSTALL_ID property stored in the Information -- section of the build script. If the user gives any one of the four hcom operations (add, add_field, install, and replace_field) that include an -approve (-apv) or install (-in) control argument and no operand follows that control, then the mbuild property value will be inserted after that control argument as the missing operand.

See the Documentation section of this MCR for details of the extended user interface.

translator_temp_ Subroutine

When the translator_temp_$empty_all_segments entry point was added, its code output parameter was never initialized. In the case where only a single temp segment was in-use, the code parameter never gets used by the entry point. So stack garbage (from the caller's stack frame) gets returned to the caller.

This unreported problem was found in testing mbuild's use of this subroutine. It will be corrected as part of this MCR.

Testing

The main changes proposed by this MCR occur in mbuild code unrelated to build requests like compile, archive_prep, and install_ec. So testing does not include those requests or their associated build operations. Such requests work in the same fashion for both currently-installed mbuild and the enhanced mbuild 2.00.

New features implemented or affected by changes proposed in this MCR include:

- mbuild subsystem start-up
  - locating the build directory and build script segment
  - reading the build script, and converting an old-style script to new format
  - setting up the build runtime environment to match properties given in the build script Information -- section
  - enabling the mbuild_monitor_ subsystem
- mbuild requests affected by start-up, or supporting new properties
  - read request
  - save request
  - set request
  - add request
  - print request
  - hcom request

The new mbuild user interface implements six mbuild start-up use cases outlined during the design phase. Tests were then performed to ensure each use case resulted in the expected setup of the mbuild subsystem, and proper return to the original runtime environment upon exit from the mbuild subsystem. All of these setup/exit tests completed successfully.

Tests were then performed to ensure the existing and new features of each changed mbuild request worked correctly, including any auto-save operations when mbuild internal data was modified. Each of these request-oriented tests completed successfully.

Finally, functional testing of the overall correct operation of mbuild 2.00 subsystem was performed by using the enhanced mbuild to manage build/install operations for the mbuild 2.00 changes.

The enhanced mbuild has also been used for several other changes planned for the MR12.8 release, and for installations to other libraries on the GHM system. A few minor issues were found/fixed during these more extensive tests. Overall, the mbuild 2.00 seems to resolve most issues on the wish list driving this enhancement effort. My thanks go to Eric Swenson for his help in this pre-install testing.

Documentation

The mbuild.info segment contains several info blocks that document requests or general info topics being changed by this MCR. The major changes are shown in the sub-sections below either as compare_ascii output, or by full text describing the request.

Note that full text describing the mbuild command is include above in the **Command Interface: mbuild, mb** section of this MCR.

Info: build_script.gi.info

Changes to the build_script.gi.info info block are shown below. Only a few of the early statements in Build Script Language are changing: in this case, being removed. Therefore, the BNF description of the language is excluded from the cpa output below, but remains in the actual info block immediately following the last changed line shown below.

```
A1260     :Info: build_script.gi: build_script:  2019-10-25  Build Script Language
A1261
A1262     A build script:
A1263       - briefly (1-4 lines or so) describes a software change to the
A1264         Multics Libraries; and
A1265       - lists the original-content files residing in that installation
A1266         directory.
A1267
A1268     The script gives instructions for creating a change or new
A1269     installation.  Included steps cover: compiling source files; changing
A1270     bound object source and object archives; bindfile changes to create
A1271     the bound object; and installing the source, bound and unbound objects,
A1272     include files and info segments, etc. into directories of the Multics
A1273     Libraries.
A1274
A1275
A1276     Origin:
A1277     The mbuild save request creates an initial version of the build script
A1278     based upon mbuild's analysis of segments in the install directory.  The
A1279     user may then edit the script to correct analysis errors, supply
A1280     unknown data, specify names to be added to or removed from segments,
A1281     etc.
A1282
A1283     A subsequent mbuild read request parses the edited build script,
A1284     loading the corrected information into mbuild.
A1285
A1286
A1287     Build script overview:
Changed by B to:
B1339     :Info: build_script.gi: build_script:  2022-09-06  Build Script Language
B1340
B1341     Notes on build script segments:
B1342     A build script contains four sections:
B1343       Descriptions --
B1344         free-form text (perhaps as bullet items) briefly describing
B1345         purpose of a software change to the target library.
B1346       Changes --
B1347         lists the original-content segments residing in that installation
B1348         directory, with information about an OPERATION to be performed,
B1349         a containing library for each object, etc.  Each segment is
B1350         described by a statement of the Build Script Language (see below).
```

```
B1351       Information --
B1352         gives attributes for the build process.  mbuild maintains content
B1353         of this section.
B1354
B1355

B1356       Status --
B1357         free-form text giving the developer's perspective on progress in
B1358         designing, coding, testing, and documenting the change.  The
B1359         developer, auditor or installer may edit this section.
B1360
B1361
B1362     Contents of the  Changes --  section is described by the information
B1363     below.
B1364
B1365
B1366     Notes on changes section:
B1367     The  Changes --  section of the script gives instructions for creating
B1368     a change or adding a new segment, etc.  Included steps cover:
B1369     compiling source files; changing bound object source and object
B1370     archives; bindfile changes to create the bound object; and installing
B1371     the source, bound and unbound objects, include files and info
B1372     segments, etc. into directories of the Multics Libraries.
B1373
B1374
B1375     The mbuild command creates an initial version of the build script
B1376     segment starting with an empty  Changes --  section.
B1377
B1378     The scan request looks for original-content segments in the build
B1379     directory, and guesses about where each segment fits in library
B1380     directories specified in the current library descriptor.  It points
B1381     out new segments which don't match any existing library item.  scan
B1382     guesses are stored in mbuild's internal data.
B1383
B1384
B1385     The analyze request uses scan information to associate each source
B1386     with an existing bound segment in the library.  For new source
B1387     segments, it guesses they will be compiled into a standalone object
B1388     not associated with any bound segment.  A new info segment is
B1389     associated with the existing info directory of the current library,
B1390     or with an UNKNOWN.info directory if the library descriptor describes
B1391     several libraries.
B1392
B1393
B1394     When the analyze request ends, it saves a Build Language Statement for
B1395     each original-content segment into the  Changes --  section of the
B1396     build script.
B1397
B1398     The user may then edit the  Changes --  section of the script to
B1399     correct analysis errors, supply unknown data, specify names to be
B1400     added to or removed from segments, etc.
B1401
B1402
B1403     A subsequent mbuild read request parses the edited  Changes --
B1404     section and loads the corrected information into mbuild's internal
B1405     data.
B1406
B1407
```

```
B1408     Overview of build script language:
B1409     Each original-content segment of a build is described by a statement
B1410     in the Build Script Language.  Language statements are summarized
B1411     below.
B1412
```

Doc for Request: read, rd

Full documentation for the mbuild `read` request is shown below. New control arguments are highlighted in bold text.

```
2022-08-06  read, rd

Syntax:  rd   {-control_args}

Function: reads information about the build operation from a build
script segment.  If reading the  Changes --  section of the script,
data from those Build Script Language statements replaces segment
build details from any earlier scan or read request.  Run another
analyze request to determine operations needed for the build.

Control arguments (sections):
   Select zero, one or more sections to read from the build script.
   By default, all sections are read.
-description, -desc
   Read the  Description --  section of the build script.  Segment
   information obtained by earlier scan or read requests is preserved.
-changes, -change, -chg
   Read the  Changes --  section of the build script.  Segment build
   details from any earlier scan or read request are replaced by
   statements in the current Changes section.  Run another analyze
   request to determine operations needed for the build.

-information, -info
   Read the  Information --  section of the build script.  Segment
   information obtained by earlier scan or read requests is preserved.
-status, -st
   Read the  Status --  section of the build script.  Segment
   information obtained by earlier scan or read requests is preserved.

-all, -a
   Read the entire build script.  Segment information from an earlier
   scan or read request is replaced by segment information read from
   the build script.  This replacement data must then be analyzed.
   (default)
-no_description, -ndesc
   Do not read the  Description --  section of the script.
-no_changes, -no_change, -nchg
   Do not read the  Changes --  section of the script.
-no_information, -ninfo
   Do not read the  Information --  section of the script.
```

```
-no_status, -nst
   Do not read the  Status --  section of the script.

Control arguments (display):
-print, -pr
   Print segment information obtained by this read request. (default)
-no_print, -npr
   Do not print data read by this request.
-brief, -bf
   Do not report an error if the build script segment does not exist.

-debug LEVEL, -db LEVEL
   LEVEL=0: disable debugging when reading the script. (default)
   LEVEL=1: display the matching reduction as input tokens in the
            script are matched.
   LEVEL=2: display all input tokens before parsing begins; then
            perform level 1 debugging.
```

Doc for Request: save, sv

Full documentation for the mbuild `save` request is shown below.

```
2022-08-24  save, sv

Syntax:  save
         sv

Function: save mbuild's information about the current build operation
to the build script segment.  All sections are updated from mbuild's
internal data.  If statements from the  Changes --  section have not
been analyzed yet, the  Changes --  section from the existing build
script is preserved.
```

Doc for Request: print, pr, p

The first part of the full documentation for the mbuild `print` request is shown below. The unshown parts of the full documentation are not changed by this MCR. New -control_args are highlighted in bold font.

```
2022-08-20  print, pr, p

Syntax:  p {-control_args}

Function: prints information about a build operation, before or after
an analyze request.  This information may come from sections of the
build script segment, or from analysis of the Build Language
statements, etc.

Control arguments (build script sections):
   One or more of the following may be given to select data to be
```

```
   printed about the current build.
-script, -scr
   print information contained in every section of the build script
   segment.
-directory, -dir
   print the current build directory pathname.
-description, -desc
   print the current description added by the installer for this
   build directory.  The description is specified using the set
   request, lines may be appended using the add request, or by editing
   the Build Script segment to add/change the  Description --  section
   at the top of this segment.

-changes, -change, -chg
   print from mbuild's internal data all Build Script Language
   statements describing this build.  This information may include
   guesses made by the scan and analyze requests.  If an analyze
   request has not succeeded since the last scan or read request,
   the displayed statement may not appear as yet in the build script.
-information, -info
   print data items in the  Information --  section of the build
   script segment.
-status, -st
   print the current data describing status of build operations.  This
   data may be specified using the set request, lines may be appended
   using the add request, or by editing the Build Script segment to
   change the  Status --  section at the top of this segment.

Control arguments (other types of data):
   To display other kinds of data, only one of the following may be
   given.
-Seg STARNAME, -seg STARNAME
   print all information about a given original-content segment in the
   build directory.
-list {LIST_NAME}, -ls {LIST_NAME},
-thread {LIST_NAME}, -th {LIST_NAME}
   print all entries on one of the threaded lists maintained by mbuild.
   LIST_NAME may be "all" to display items on all threaded lists; or
   one of the IDs given in "List of thread selectors" below.  "all" is
   the default if no LIST_NAME is given.

-log_dir, -ld
   print the current log directory, used in the update_seg initiate
   request.  This directory contains segments describing the
   installation: Installations.log, Installations.info
-new
   print original-content segments found by a scan of the install
   directory that are not mentioned in the build script.  When a
   read request loads the build script into mbuild, it reports this
   same possible omission of segments; print -new allows the user
   to re-display this data.
-unknown, -unk
   print original-content segments found by a scan of the install
```

```
   directory that have an UNKNOWN library designation.
-save_format, -save
   print build data as it would appear in the build script segment.

Control arguments (build statement format):
-scan, -sc
   print data gathered by most recent scan of the install directory,
   or read from the build script segment. (default if an analyze
   request has not been issued.)
-analyze, -az
   print build data reshaped by an analyze request.  (default
   after an analyze request is issued.)

Control arguments (build statement details):
…
```

Doc for Request: set

Full documentation for the `set` request is shown below. New -control_args are highlighted in bold font.

```
2022-08-06  set

Syntax:  set -control_args

Function: sets the target library or library.directory for a given
segment in the  Changes --  section of the build script; or sets
values for other sections of the build script.

The build script includes four sections:
 Description --
   Free-form text giving an hcom-style summary of major changes.
 Changes --
   Build Script Language statements describing the segments being
   changed by the build.
 Information --
   Build phase and key identifiers associated with the build script.
 Status --
   Free-form text describing current status of various steps of the
   build operation.

This request sets (completely replaces) current values with new
information.  To append new lines to the free-form sections, or
additional APPROVE_IDs or USER_IDs, use the add request.  For details,
type:    help add

To change the library descriptor named in the  Information --
section, use the mbuild request:  lds set DESCRIPTOR_NAME
```

```
Control arguments (build script information section):
-phase PHASE_NAME
   sets the current phase of operation to one of the following values:
      developing
         changes are still being added to the build.
      auditing
         changes are frozen (mostly) and are now being audited.
      installing
         fully-audited changes are now being installed.

-approve APPROVE_IDs,
-apv APPROVE_IDs
   associates one or more APPROVE_IDs (e.g., MCR10019) with the build
   script.  If several APPROVE_IDs are given, all will be listed in
   the build script.  And if only one APPROVE_ID is given, that value
   will be used if no APPROVE_ID is given in the request:
     hcom add_field -apv APPROVE_ID
   or any other hcom operation accepting the -approve control
   argument.

-install INSTALL_ID,
-in INSTALL_ID
   associates a Multics Release INSTALL_ID (e.g., MR12.8-1012) with
   the build script.  That value will be used if no INSTALL_ID is
   given in the request:  hcom add_field -install INSTALL_ID
   or any other hcom operation accepting the -install control
   argument.
-log_dir DIRNAME, -ld DIRNAME
   sets the log directory used by the update_seg initiate command.  If
   no log directory has been set by "mbuild -set_log_dir DIR" or the
   request "set -log_dir DIR", the install_ec request will use the
   directory selected by library descriptor matching -library *.log
   if one is defined.

-developer USER_IDs,
-dev USER_IDs
   one or more PERSON_ID.PROJECT_ID values identifying users
   developing the changes.
-auditor USER_IDs,
-aud USER_IDs
   one or more PERSON_ID.PROJECT_ID values identifying users
   auditing the changes.
-installer USER_IDs,
-inst USER_IDs
   one or more PERSON_ID.PROJECT_ID values identifying users
   installing the changes.
```

```
Control arguments (build script other sections):
-description, -desc
   prompts for a new description of the build operation.  The
   description text ends with a line containing only a period (.)
   character.  The new text is saved in the  Description --  section of
   the build script to preserve its value across mbuild invocations.
   Another way to set or revise the description is by editing the
   build script, then issue the request:  read -desc
-status, -st
   prompts for a new status of the build process.  The text entered
   ends with a line containing only a period (.) character.
   The new text is saved in the  Status --  section of the build
   script to preserve its value across mbuild invocations.
   Another way to set or revise the status text is by editing the
   build script, then issue the request:  read -status

Control arguments (build script changes section):
   Name a segment using the -seg control_arg followed by a -library
   control_arg giving a new value.
-Seg SEG_NAME -control_arg,
-seg SEG_NAME -control_arg
   selects a segment to modify.  The star convention may be used
   to select several segments to modify.  The -control_arg specifies
   which attribute of the Seg structure to modify.  Only the following
   attribute may be changed.

-library LIBRARY, -lb LIBRARY
   gives a new value for the Seg().library attribute.  Use one of the
   formats: LIB_NAME.DIR_NAME  or  LIB_NAME  or  DIR_NAME

List of preferred library names:
   LIB_NAME.DIR_NAME combinations that are short and readily
   identifiable.  Note that hard.i and mcs.i reference a directory
   named info, but it contains hardcore/communication configuration
   files and maps, not info segments.  To see directories named by
   these libraries, enter the request: libs
```

Doc for Request: add

Full documentation for the new `add` request is shown below.

```
2022-08-06  add

Syntax:  add -control_arg

Function: adds lines to the  Description --  and  Status --  sections
of the build script; or adds values to the Approve_IDs, Developers,
Auditors, or Installers properties of the  Information --  section of
the build script segment.
```

```
The build script includes four sections:
 Description --
   Free-form text giving an hcom-style summary of major changes.

 Changes --
   Build Script Language statements describing the segments being
   changed by the build.

 Information --
   Build phase and key identifiers associated with the build script.

 Status --
   Free-form text describing current status of various steps of the
   build operation.

To completely replace any of the free-form sections or arrays, or
to change the values of scalar elements in the  Information --
section, use the set request.  For details, type:  help set

Control arguments (build script information section):
   Select data to be set by giving one of the following control args.
-approve APPROVE_IDs,
-apv APPROVE_IDs
   adds one or more APPROVE_IDs (e.g., MCR10019) to the
   Information --  section of the build script.  If several
   APPROVE_IDs are given, all will be listed.  And if only one
   APPROVE_ID is listed in the build script, that value will be used
   if no APPROVE_ID is given in the request:
     hcom add_field -apv {APPROVE_ID}
-developer USER_IDs,
-dev USER_IDs
   one or more PERSON_ID.PROJECT_ID values identifying users
   developing the changes.  These will be added to any existing
   user_IDs if they are not already present in the array.

-auditor USER_IDs,
-aud USER_IDs
   one or more PERSON_ID.PROJECT_ID values identifying users
   auditing the changes.  These will be added to any existing
   user_IDs if they are not already present in the array.
-installer USER_IDs,
-inst USER_IDs
   one or more PERSON_ID.PROJECT_ID values identifying users
   installing the changes.  These will be added to any existing
   user_IDs if they are not already present in the array.

Control arguments (build script other sections):
-description, -desc
   prompts for new lines to be appended to the Description --  section
   of the build script.  The new lines end with a line containing only
```

```
   a period (.) character.  Another way to add or revise the
   description is by editing the build script segment.
-status, -st
   prompts for new lines to be appended to the  Status --  section of
   the build script.  The new lines end with a line containing only a
   period (.) character.  Another way to add or revise the status text
   is by editing the build script segment.
```

Doc for Request: history, hcom

mbuild documentation for the `hcom` request includes only a modified version of the Multics history_comment command documentation, with a list of operations supported by the hcom command. This info block discusses differences between the mbuild hcom request and the Multics history_comment command. Specific documentation for each history_comment operation is not included.

So the difference between the -approve and -install control arguments usage in the mbuild hcom request versus the Multics history_comment command will be discussed in this "differences" block for the mbuild hcom request.

The added -control_arg differences are highlighted in bold font.

```
2022-09-04  history, hcom

Syntax:
   hcom    HCOM_OPERATION {SEG_NAME} {HCOM_ARGS_AND_CONTROL_ARGS}

Syntax as an active request:
   [hcom    HCOM_OPERATION {SEG_NAME} {HCOM_ARGS_AND_CONTROL_ARGS}]

Function: applies the Multics history_comment command to
original-content source, include, info, and bindfile segments
in the build directory.

This information describes differences between the full Multics
history_comment (hcom) command and the mbuild history (hcom) request.
The mbuild hcom request makes the SEG_NAME argument optional by
supplying names of original-content segments from the build script
when SEG_NAME is omitted from the hcom request.  It then invokes the
Multics hcom command with those addition arguments for each matching
original-content segment.

To learn more about the full Multics history_comment command, type:
   .. help hcom
To learn more about individual history_comment operations, type:
   .. help hcom.HCOM_OPERATION
For example:  .. help hcom.add_field
```

```
Arguments:
HCOM_OPERATION
   names the history_comment operation to perform on each segment.  See
   "List of hcom operations" below.
SEG_NAME
   names the original-content segment(s) to process.  The name includes
   the language suffix (e.g., probe.pl1).  If not given, all
   original-content source, include, info, and bindfile segments in the
   installation directory are processed.  The star convention may be
   used to select several segments.

HCOM_ARGS_AND_CONTROL_ARGS
   give any arguments and control arguments accepted by the:
      history_comment HCOM_OPERATION
   command.  For argument details, type:
      .. help hcom.HCOM_OPERATION
   where HCOM_OPERATION is an operation from "List of hcom operations"
   below.  Use double-quote characters around any Argument containing
   special characters.

Control arguments (that differ from the hcom command):
-approve {APPROVE_ID},
-apv {APPROVE_ID}
   inserts into selected history comments having no approve element
   the given APPROVE_ID value.  The APPROVE_ID operand may be omitted
   if the build script  Information --  section contains a single
   Approve_IDs value; that Approve_ID attribute is used as the missing
   APPROVE_ID value.

-install {INSTALL_ID},
-in {INSTALL_ID}
   specifies an identifier associated with installing the changed
   module into execution libraries.  The INSTALL_ID operand may be
   omitted if the build script  Information --  section contains an
   Install_ID value; that Install_ID attribute is used as the missing
   INSTALL_ID value.
```

…

Version History

Date	Revision	Author	Comment
2022-09-11	1.0	Gary Dixon	Initial version of the MCR proposal.
2022-09-12	1.1	Gary Dixon	Change mbuild -installing per request from Eric Swenson.