IGORE Software Configuration / Development with CMS/MMS

IGORE Version 7.1

Release Date: December 1, 1995

Contents

1 Introduction **1**

2 Igore CMS Libraries **1**

3 Common CMS Commands **2**

3.1 Library Commands 2

3.2 FETCH 2

3.3 RESERVE 3

3.4 REPLACE 3

3.5 Handling Multiple Reservations 3

4 Igore MMS Description Files **4**

5 IGORE Run-time Directory Structure **5**

6 Igore CMS/MMS Usage **6**

7 Remote Site Configuration Plan (OSSENU) **7**

8 Distribution Kit Preparation **7**

1 Introduction

The nominal baseline for the CMS/MMS configured version is Igore 7.1. Further revisions to Igore will be made in the CMS/MMS configured version (in parallel with existing IGRDEV version until cut-over). This document is a guide to the procedures used to make those revisions in the CMS/MMS configured environment.

2 Igore CMS Libraries

Source code is contained in CMS libraries. CMS provides a set of tools to access the source modules in the libraries. On-line help for the CMS is available. For detailed information, see Digital's document, *Guide to the VAX DEC/Code Management System.*

The Igore source code is split across several libraries. CMS performance benefits from the use of several smaller libraries rather than a single large library. The libraries are assigned logical names for convenience. Following is a list of logicals and corresponding libraries:

```
"IGR_CMS_AFE" = "ODT2:[IGRCMS.AFE.CMSLIB]"
"IGR_CMS_BLD" = "ODT2:[IGRCMS.BLD.CMSLIB]"
"IGR_CMS_DOC" = "ODT2:[IGRCMS.DOC.CMSLIB]"
"IGR_CMS_ICL" = "ODT2:[IGRCMS.ICL.CMSLIB]"
"IGR_CMS_IDL" = "ODT2:[IGRCMS.IDL.CMSLIB]"
"IGR_CMS_SDB" = "ODT2:[IGRCMS.SDB.CMSLIB]"
"IGR_CMS_ST" = "ODT2:[IGRCMS.ST.CMSLIB]"
"IGR_CMS_STRUC" = "ODT2:[IGRCMS.STRUC.CMSLIB]"
"IGR_CMS_SYS" = "ODT2:[IGRCMS.SYS.CMSLIB]"
```

CMS also provides an option to keep a "reference copy" of the latest generation of any library element in a particular directory. The following logicals and directories contain reference copies of the Igore source code:

```
"IGR_REF_AFE" = "ODT2:[IGRCMS.AFE.REFLIB]"
"IGR_REF_BLD" = "ODT2:[IGRCMS.BLD.REFLIB]"
"IGR_REF_DOC" = "ODT2:[IGRCMS.DOC.REFLIB]"
"IGR_REF_ICL" = "ODT2:[IGRCMS.ICL.REFLIB]"
"IGR_REF_IDL" = "ODT2:[IGRCMS.IDL.REFLIB]"
"IGR_REF_SDB" = "ODT2:[IGRCMS.SDB.REFLIB]"
"IGR_REF_ST" = "ODT2:[IGRCMS.ST.REFLIB]"
"IGR_REF_STRUC" = "ODT2:[IGRCMS.STRUC.REFLIB]"
"IGR_REF_SYS" = "ODT2:[IGRCMS.SYS.REFLIB]"
```

3 Common CMS Commands

3.1 Library Commands

From a programmer's perspective, there are subset of CMS commands that are used frequently. The following should be considered only as an introduction to that subset of commands. Please see the documentation and on-line help for details.

CMS maintains a dynamic per-process list of libraries in its search path. To display this library list, use the command:

```
$ CMS SHOW LIBRARY
```

To set the list to a specific library or set of libraries, use:

```
$ CMS SET LIBRARY library-name1 [,library-name2...]
```

To add a library to the end of the list:

```
$ CMS SET LIBRARY/AFTER library-name
```

A library can also be added to the beginning of the list, or before or after a specified element of the current list.

Files in a CMS library are known as "elements". To display a list of elements in the current library list:

```
$ CMS SHOW ELEMENT
```

3.2 FETCH

To access an element in a CMS library for compiling or browsing, use the command:

```
$ CMS FETCH element-name "remark-string"
```

The element name must include the file extension or a period if no file extension is present. Note the remark field, which is arguably unnecessary for most fetch operations. A null string can be specified.

3.3 RESERVE

To access an element in a CMS library for modifications, the element must be reserved. This prevents multiple users from unwittingly modifying the same element. To reserve an element:

```
$ CMS RESERVE element-name "remark-string"
```

If another user has already reserved the element, that information is displayed although the reservation will complete successfully.

3.4 REPLACE

To replace an element in the library after modifications:

```
$ CMS REPLACE element-name "remark-string"
```

The remark string should be considered a requirement for a replace operation. Only an element reserved from the library can be replaced. An element obtained with the CMS FETCH command cannot be replaced in the library. Replacing the element creates a new "generation" of the element. It remains possible to access previous generations (see the manual).

3.5 Handling Multiple Reservations

In case of multiple reservations, it is not possible to replace the element without special action. Several options are possible, but first, it is useful to know which users have reserved that element. To display the reservations for a single element:

```
$ CMS SHOW RESERVATIONS element-name
```

A list is displayed with an entry for each user with a reservation on that element. To the left of the username, a reservation ID number is displayed in parentheses. Reservation date, time, and remark are also listed.

One possible action is to cancel all reservations but one, then replace the element. To cancel a reservation:

```
$ CMS UNRESERVE element-name -
_$ /IDENTIFICATION=reservation-id-number "remark"
```

The reservation ID number is only required when multiple reservations exist.

Another possibility is to replace all but one of the reservations as variant generations. The command is:

```
$ CMS REPLACE element-name /VARIANT=X "remark"
```

...where X is a single alphabetic character. This generation of the element becomes the head of a branch from the main line of descent. Variant generations can be merged almost automatically (see the manual), but in most cases this is a situation to be avoided.

4 Igore MMS Description Files

MMS is similar to the Unix "make" utility. Both built-in rules and user-defined rules are used to process user-specified dependencies between source and object, object and executable, or between any files whose relationship fits the source-process-product model. Revision dates are used to determine which products must be rebuilt to reflect newer source material. For details, see Digital's document, *Guide to the VAX DEC/Module Management System.*

The Igore MMS "description" files are contained in the CMS library: IGR_CMS_BLD. The description files in place (as of 2/8/93) are:

```
Elements in DEC/CMS Library ODT2:[IGRCMS.BLD.CMSLIB]

FDB_APPS_COMP.MMS "FDB application compile."
FDB_APPS_LINK.MMS "To build FDB shareable..."
FDB_BUILD.MMS "FDB Applications build file"
IGR_BLD_DOC.MMS "IGORE Document build"
IGR_BLD_GV.MMS "Build AFE_PROCESS (gen_vctr)."
IGR_BLD_ICH.MMS "IGORE description files."
IGR_BLD_IGORE_UTIL_VCTR.MMS "Make_igore_utilities_vctr build file"
IGR_BLD_NSHR.MMS "Build NOSHR_PSECTS.EXE (check_common)"
IGR_BLD_PRE.MMS "To build AFE generator (PREPRO.EXE)."
IGR_BLD_SYS.MMS "IGORE description files."
IGR_BUILDSHR.MMS "Make BUILDSHR utility."
IGR_SYSTRAN.MMS "To build Igore system structure utilities."
IGR_USERTRAN.MMS "To build Igore user structure translator  utilities."
MAKEFILE.MMS  "To build the whole enchilada."
MISC_APPS_COMP.MMS "Miscellaneous Applications Compilation Build file"
MISC_APPS_LINK.MMS "MISC link file"
MISC_BUILD.MMS "MISC build file"
```

```
PSR_APPS_COMP.MMS "PSR application compile."
PSR_APPS_LINK.MMS "PSR link file"
PSR_BUILD.MMS "PSR Applications build file"
SDB_APPS_COMP.MMS "SDB application compile."
SDB_APPS_LINK.MMS "IGORE description files."
SDB_BUILD.MMS "SDB Applications build file"
```

Many of the Igore description files define special rules to perform tasks such as generation of AFE's and option files, concatencation of linkimage commands into application initialization files, etc.

The Igore description files also define several special macros to control the build process. The Igore standards for these macros are:

- CLEAN - If true, build process will delete intermediate products. (.OBJ files, etc.)
- CLEAN_SOURCE - If true, local copies of source files will be deleted. Dangerous if source code is not yet configured.

Other macros specific to a given description file may also be present. Macros can be specified in an MMS command line as follows:

```
$ MMS/CMS/DESCRIPTION=LINK_IGORE.MMS/MACRO=(CLEAN=1)
```

In the example above the description file LINK_IGORE.MMS will delete any intermediate code, including object code (CLEAN=1). Note the use of the /CMS qualifier, which directs MMS to use CMS to search the CMS libraries in the current CMS library list for source code not found in the current directory.

5 IGORE Run-time Directory Structure

The run-time directory structure and associated logical names are as follows:

```
"IGR$ROOT"  = "IGR:[000000]" - initialization files.
"IGR$DAT" = "IGR:[DAT]"    - Igore data files.
"IGR$DOC" = "IGR:[DOC]"    - documentation and help.
"IGR$EXE" =  "IGR:[EXE]"    - Igore executables and shareables.
"IGR$IDL" =  "IGR:[IDL]"    - Idl procedures.
"IGR$LIB" = "IGR:[LIB]"    - Current object libraries.
"IGR$STRUC" = "IGR:[STRUC]"  - Structure definitions, source and binary.
```

6 Igore CMS/MMS Usage

The following steps are required to update or create a source module.

1. Set the CMS library list to contain the appropriate libraries. For example, to update an Igore SDB application module:

   ```
   $ CMS SET LIBRARY IGR_CMS_SDB
   ```

2. Only if updating a module, use the CMS reserve command to obtain a copy of the source module to be updated. Continuing with the example:

   ```
   $ CMS RESERVE SDB$OPEN.FOR "For update of file-info table."
   ```

   Note that the third parameter for this command is a remark string. A null string can be provide, but a brief comment is highly recommended.

3. Edit the source code to make the required modifications.

4. If you wish to test the module before you configure the source, invoke MMS to process the appropriate description file, although first you will need to add the libraries for include files and description files to your library list. Continuing with the example:

   ```
   $ CMS SET LIB/AFTER IGR_CMS_ICL, IGR_CMS_BLD
   $ MMS/CMS/DESCRIPTION=SDB_BUILD.MMS/SKIP -
   _$ /MACRO=(FFLAGS=/DEBUG/OPT,LFLAGS=/DEBUG,CLEAN=TRUE)
   ```

   This will produce a copy of the IGORE_SDB.EXE image in the current directory that can be used for testing. Note the use of the /AFTER qualifier to CMS, which places the specified libraries after any libraries currently in the list.

5. Use the procedure TEST_IMAGE.COM to to invoke Igore to test this sort of private image. In this example is invoked by:

   ```
   $ IGR_TESTIMAGE SDB
   ```

   The parameter to this command may be any subset of the following list: IGORE, SDB, FDB, PSR, MISC. The appropriate logicals will be temporarily defined to use local copies of the specified images. (Not yet available as of 2/8/93.)

6. Once updated code has been determined to perform correctly, the new source code should be replaced in the appropriate CMS library. To continue the example with a updated module:

   ```
   $ CMS REPLACE SDB$OPEN.FOR "Update file-info utilities."
   ```

Or, if the a new module is to be added:

```
$ CMS CREATE ELEMENT SDB$TWINKLE.FOR "TWINKLE ANALYSIS UTILS."
```

Note that by default, CMS deletes the input file once it creates the library element. If necessary, this can be overridden with the /KEEP qualifer. As a rule this qualifier should not be used, avoiding the confusion of unconfigured source code on the loose.

7. The new image should next be built into the Igore run-time directory. Continuing with the example:

```
$ SET DEFAULT IGR$LIB
$ MMS/CMS/DESCRIPTION=SDB_BUILD.MMS /SKIP_INTERMEDIATE -
_$ /MACRO=(CLEAN=TRUE,CLEAN_SOURCE=TRUE) -
_$ PRODUCTION
```

The parameter in this command example specifies a target name. The convention used by Igore description files shall be to define the target name "PRODUCTION" to equate to the correct run-time directory specification for resulting executable or other product file(s).

Also, note the use of the /SKIP qualifier on the MMS command. This signals MMS not to bother rebuilding intermediate files, in this instance individual object files, when the appropriate module in the object library is already up to date with the source. See the DEC documentation and help for details. As a rule, the Igore description files are designed to allow this behaviour if required.

When working on a development copy of Igore, steps 4-5 may be skipped.

7 Remote Site Configuration Plan (OSSENU)

Once the baseline Igore 7.1 has been established, OSSENU will be supplied with CMS libraries identical to the libraries at NRL. The libraries will be maintained as closely in parallel as possible.

CMS history records will allow the configuration manager to identify new or modified source code and take the appropriate actions to update other sites. This will remain a manual procedure, but will benefit from the tools provided by CMS.

Included with the CMS libraries will be the MMS description files to build and/or update Igore. (The initial Igore build of the baseline version at OSSENU will be performed by NRL, as will further updates until desired by OSSENU.)

8 Distribution Kit Preparation

(TBS)