How to build and package the CMVC Server/Client for UNIX

Document Number TR 29.2269

Angel Rivera, Lee Perlov, Gurpreet Koura

TeamConnection/CMVC Development
IBM Software Solutions
Research Triangle Park, North Carolina
Copyright (C) 1997 IBM Corp.
All rights reserved.

ABSTRACT

The purpose of this technical report is to provide a real example of the use of CMVC for developing, maintaining, building and packaging the CMVC Server and Client for UNIX, which encompasses several operating systems and database management systems. This example illustrates the complexity associated with the development of applications in heterogeneous environments.

ITIRC KEYWORDS

CMVC

UNIX

Building

Packaging

ABOUT THE AUTHORS

ANGEL RIVERA

Mr. Rivera is a Staff Programmer and team lead for the CMVC development. He joined IBM in 1989 and since then has worked in the development and support of library systems.

Mr. Rivera has an M.S. in Electrical Engineering from The University of Texas at Austin, and B.S. in Electronic Systems Engineering from the Instituto Tecnológico y de Estudios Superiores de Monterrey, México.

LEE R. PERLOV

Mr. Perlov is a Staff Programmer in the TeamConnection/CMVC development group. He started working for IBM in 1985 in Gaithersburg, Md, working in the Federal Systems Division on various projects for the United States intelligence community. He then moved to RTP to work on library development and support.

Mr. Perlov received a B.S.Acc degree in Accounting from the University of Florida in 1983. He also completed two years of graduate work in the Department of Computer Science at the University of Florida.

GURPREET KOURA

Mrs. Koura is a Senior Associate Programmer and customer service contact for CMVC and TeamConnection. She joined IBM in 1996.

Mrs. Koura received a B.Sc. in Computer Sciences from North Carolina Central University in 1993.

CONTENTS

ABSTRACT . . . iii
ITIRC KEYWORDS . . . iii

ABOUT THE AUTHORS . . . v
Angel Rivera . . . v
Lee R. Perlov . . . v
Gurpreet Koura . . . v

Figures . . . x

Building and packaging CMVC for UNIX . . . 1
Disclaimer . . . 2

Challenges related with heterogeneous environments . . . 3
Maintaining all needed combinations of hardware and software . . . 3
Role of 'build' userid and exported directory . . . 3
Standardize the location of ksh to /usr/bin/ksh . . . 5
Identifying the operating system . . . 5
Shared .profile . . . 7
Using a bulletin to communicate news (message-of-the-day) . . . 7

Overall process with CMVC . . . 9
Families, Components and Releases . . . 9
Process model . . . 10
Project lead creates components and releases . . . 10
Handling defects and features . . . 11
Developers work with files using tracks (Code and Unit Testing) . . . 11
Preparing the code for Function Verification Test (FVT) . . . 12
Backing out a faulty FVT driver . . . 13
Dealing with Tracks/Features during FVT . . . 14
Dealing with defects created during FVT . . . 15
Severities for defects . . . 15

Preparing the builds . . . 17
General assumptions . . . 17
One-time only activities . . . 17
Create CMVC family 'build' to be used with DB2 . . . 17
Create necessary links . . . 18

Creation of the Build/Ship/Package tree . . . 19
Estimated time for build and packaging . . . 20
Estimated time for all combinations . . . 20

Building the code . . . 21
Overview of the whole process . . . 21
Location of the source code . . . 23
Build/Ship/Package directory structure . . . 23
Top directory structure - created via shell script . . . 23
Source directory structure - created by extracting release 'cs230' . . . 24
Install directory structure - created by extracting release 'inst230' . . . 26
Bottom directory structure - created by doing make with 'ship' . . . 27
Activities to be done per each version, release or modification . . . 28
Files that need to be modified due to changes in Copyright or VRM . . . 28
Files that need to be modified when adding new files . . . 32
Preparation for the Build . . . 32
Process Tracks, Levels and Level Members . . . 33
Extraction of the tools for the build and packaging process . . . 33

Building the CMVC Server and Line Commands . . . 35
Complete cleanup of the Build tree . . . 35
Cleanup of the previous build . . . 35
Extraction of the source files . . . 35
Extraction using Release or Level Extract . . . 36
Building the code . . . 37
Miscellaneous notes . . . 37
Notes on the MAKE.cmvc shell script . . . 37
Files affected by different MAKE.cmvc -c options . . . 38
Making a DEBUG driver . . . 39
Making an FVT driver . . . 40
Making a BETA driver . . . 40
Making a Gold driver . . . 40
Gathering the shippable files (ship step) . . . 40
Beta installation . . . 41
Known building problems . . . 41
Known Problems when Testing the code . . . 43

Building the CMVC GUI . . . 45
Build/Ship/Package directory structure for the GUI . . . 45
Preparation for the build . . . 46
Complete cleanup of the build tree . . . 46
Extraction of the source files . . . 46
Extraction using Release or Level Extract . . . 46

Building the code . . . 47
Cleanup of object modules and message catalogs . . . 47
Considerations before building . . . 47
Making a DEBUG driver . . . 47
Making an FVT driver . . . 48
Making a BETA driver . . . 48
Making a Gold driver . . . 48
Building the GUI . . . 48
Known build problems . . . 49
Copying the output files into the ship directory . . . 50
Performing a sanity test (based on a pseudo-installation) . . . 50
Check the dates of the CMVC code recently built . . . 50
Copy the CMVC code recently built into the install directories . . . 51
Test the CMVC code recently built from the target host . . . 51
Erase the CMVC code from the install directories (optional) . . . 52

Packaging the CMVC server, line commands and GUI . . . 53
Overview of packaging for the Server . . . 53
Overview of packaging for the Client (line commands and GUI) . . . 54
Standalone files . . . 54
Workbench Integrated files . . . 56
Preparing AIX smit installp images . . . 57

Appendix A. Introduction to the checklists . . . 59
Miscellaneous notes . . . 59

Appendix B. CMVC for UNIX, build and package status checklist . . . 61

Appendix C. CMVC Client/Server UNIX, Build and Package Checklist . . . 63

Appendix D. CMVC GUI for UNIX, Build and Package Checklist . . . 67

Appendix E. How to put the CMVC code into tapes . . . 69
How to get tapes via VM . . . 69
Sequence to make a tape for CMVC . . . 69

Appendix F. Bibliography . . . 71
How to get electronic copies of manuals and TRs . . . 71
IBM Intranet . . . 71
Internet . . . 72

Appendix G. Copyrights, Trademarks and Service marks . . . 73

FIGURES

1. get_opsys(): Shell script to find out name of operating system 6
2. motd: Shell script to edit and view the message-of-the-day 8
3. Top directory structure - created manually via script 23
4. Source directory structure - created by release 'cs230' (1 of 2) 24
5. Source directory structure - created by release 'cs230' (2 of 2) 25
6. Bottom directory structure - created by release 'inst230' 26
7. Bottom directory structure - created by doing make with 'ship' 27
8. Header file with service keywords for the what command 29
9. Source file with service keywords for the what command 30
10. Files affected by different MAKE.cmvc -c options, part 1 38
11. Files affected by different MAKE.cmvc -c options, part 2 39
12. CMVC Client/Server UNIX, Build and Package Checklist, Level: ____ 63
13. CMVC GUI for UNIX, Build and Package Checklist, Level: _____ 67

BUILDING AND PACKAGING CMVC FOR UNIX

The purpose of this technical report is to provide a real example of the use of CMVC for developing, maintaining, building and packaging a product in a heterogeneous client/server environment. CMVC itself is an excellent example, because we use CMVC itself to build the CMVC Server and Client using several operating systems and database management systems.

We describe the challenges that we encounter when developing in heterogeneous environments and how we cope with them. Then we describe the development process that we used with CMVC for the development of the application.

Then, in order to give you the extent of the complexity associated with the development of an application in heterogeneous environments, we dedicate the rest of this document to the prerequisites, configuration, building and packaging the CMVC Client and Server for UNIX.

CMVC for UNIX is made available for the following operating systems:

AIX Version 3.2.5
AIX Version 4
HP-UX Version 9
HP-UX Version 10
Solaris 2.4
SunOS 4.1.3

The CMVC Server for UNIX is built using the following relational databases (although we do not provide all possible combinations of operating system and database).

DB2 Version 1
DB2 Version 2
Oracle Version 7

Starting with Oracle 7.3, there were many APIs that were dropped from Oracle, and thus, CMVC is not compatible with Oracle 7.3.

Informix Version 5
Informix Version 7
Sybase 4.9

Although we do not do a native build on Sybase 10, the build on Sybase 4.9, works fine on Sybase 10.

A set of quick status checklists with a summary of the build and package activities for the Client/Server code and the GUI code are provided in the appendices.

DISCLAIMER

The intention in writing this detailed process is certainly NOT to dictate to you the ONLY way to build the code; the intention is to document the process that we have used and that has proven to be successful.

This process is NOT perfect and there is room for improvement, and you are welcome to improve it and to provide feedback to update this document to reflect that improvement.

CHALLENGES RELATED WITH HETEROGENEOUS ENVIRONMENTS

There are several challenges that we faced during the development of CMVC for UNIX in a heterogeneous environment:

The main challenge is to maintain all needed combinations of hardware and software.

The next challenge was to seamlessly work across the different platforms. To overcome this challenge we used the same shared userid in all systems called "build".

Then, we used the message-of-the-day (motd) to communicate between the developers and reserve working areas.

MAINTAINING ALL NEEDED COMBINATIONS OF HARDWARE AND SOFTWARE

The main challenge is to maintain all needed combinations of hardware and software.

The concurrent development of the CMVC Server for UNIX is done across multiple platforms/versions (6 variations), multiple hosts (9 machines), and multiple relational database management systems (6 variations).

Furthermore, the development of the CMVC Client is done for standalone mode or integrated with SoftBench, plus in AIX and HP-UX we have to provide two versions, one for Motif 1.1/X11R4 and another for Motif 1.2/X11R5.

Only in AIX 3.2.5, CMVC is provided in all combinations of databases and clients. In the other platforms, only certain combinations were needed.

The tasks of obtaining the hardware and software, renewing the contracts, requesting and installing patches, providing maintenance (preventive and corrective) constitute a major portion of the development activities.

ROLE OF 'BUILD' USERID AND EXPORTED DIRECTORY

All building activities in all operating systems is performed from the userid 'build' which has to have the same main characteristics in all the platforms: same user id number, same group id number and same home directory.

```
uid=21 (build)
gid=222(syscmvca)
groups= (system),1(staff),2(bin),3(sys),222(syscmvca)
HOME=/home/build

NOTES:
  The groups should be appropriate to the platform
  The group system is required because it builds a setuid-root
    application (cmvcd, the CMVC daemon).
```

The actual home directory for this userid resides in an exported file system from a file server (in our case in host 'rtplib02') which must have enough capacity to hold the builds, install images and miscellaneous for all the needed combinations (in our case 2.5 GB).

This exported file system is mounted in all other platforms, which have the same userid, groupid and home directory:

```
mount rtplib 2:/home/build      /home/build
```

The advantages of this scheme are:

- This approach allows for maximum reuse and sharing of common files such as .profile and directories such as $HOME/bin.
- It is possible to work simultaneously in the same 'build' account from different operating systems. For example, one developer could be doing a build for AIX 3 and DB2 V1 meanwhile another developer could be using HP-UX 9 and Oracle 7.
- It is possible to perform CMVC release and level extracts and use the same host and specify only the target directory.

  Furthermore, it is possible to specify the "nomount" option when extracting a release/level in order to not use NFS. This local extract is faster than an extract that involves NFS and avoids the risks mentioned in “Risks of building code across NFS mounts” on page 5.
- The number of host list entries in CMVC is reduced because the system login id is always "build". We need to create only those host list entries for our CMVC user ids and hostnames.

  One important consideration when using the CMVC GUI in such environment is that there is ONLY one "Cmvc" resource file and one CMVC tasks list. Furthermore, the developers need to be aware of the CMVC user id when using the GUI, because it will reflect the name of the previous developer.

Risks of building code across NFS mounts

If your NFS setup is not too reliable, then you may have problems during the extraction of large releases or levels. One alternative is to consider doing a local extract by using the "nomount" option; also, you can consider using AFS or DFS.

New versions of NFS now support TCP instead of UDP to verify that each packet that is sent is actually received. When using NFS with UDP you should test the executables and search for compiled/linked objects of size 0 (zero):

```
find . -size   -print
```

Standardize the location of ksh to /usr/bin/ksh

The latest generation of AIX and HP-UX have adopted the directory /usr/bin as the place to have the main user utilities, such as the "ksh" (Korn shell).

This location affects the shell name used in the first line of our shell scripts, which we standardized as follows:

```
#!/usr/bin/ksh
```

In order to avoid a big pitfall with older versions of the operating systems (the ksh cannot be found but the error message is extremely cryptic and confusing), it is necessary to do the following symbolic link in those systems:

```
ln -s /bin /usr/bin
```

Identifying the operating system

We used heavily in our tools the following routine (and shell script) to identify the version of the operating system: getopsys

This was specially important to identify the Solaris system because even though Solaris and the old SunOS are NOT binary compatible, the data given by the UNIX "uname" utility is misleading, because for Solaris systems, the name "SunOS 5.x" is used to identify the "Solaris 2.x" systems.

```
#!/usr/bin/ksh
#-------------------------------------------------------------------+
# NAME: get_opsys
#
# Return a string identifying the operating system.
# Output:    OPSYS = (AIX, HP-UX, SunOS, Solaris)
#
# Notes about uname:
#      The command "uname -a" returns the following information:
#
#           operating_system_name     node_name     release_name     version_name
#
#   Examples of uname:
#
# AIX 3:              AIX carcps 6 2 3     11985C
# AIX 4:              AIX oem-ppc3 1 4     22559
# HP-UX 9:          HP-UX oem-hp 2 A. 9. 3 E 9   /735 2   248696 8-user license
# HP-UX 1 :        HP-UX oem-hp 1 B.1 . 1 A 9   /735 2 1 377972 two-user license
# Solaris 2.4: SunOS oem-sn 1 5.4 generic sun4m sparc
# SunOS 4.1.3: SunOS oem-sn 3 4.1.3_U1 1 sun4m
#
#-------------------------------------------------------------------+
get_opsys()
{
raw= uname -a
OPSYS= echo $raw | cut -f1 -d' '

# Test for Solaris (5... or higher) or SunOS (4...)
# The problem is that the Solaris op-sys is identified as SunOS, and
# there is no string that indicates that is "Solaris". The distinction
# is done only by the version number.

if ffl "$OPSYS" = "SunOS" “
then
     var= echo $raw | cut -f3 -d' ' | cut -c1-1
     if ffl $var -ne 4 “
        then
           OPSYS="Solaris"
    fi
fi
export OPSYS
echo "get_opsys: " $OPSYS
}
get_opsys

# end of file
```

Figure 1. get_opsys(): Shell script to find out name of operating system

Shared .profile

The shared .profile has to be comprehensive enough to accommodate for the differences between platforms and databases. For example, the location of one utility may be /bin in one system or /usr/bin in another.

We use the name of the host inside a case statement in order to provide environment variables that deal with these differences.

USING A BULLETIN TO COMMUNICATE NEWS (MESSAGE-OF-THE-DAY)

Because the same build directory is used across the systems, it is important to have a "bulletin" for messages to avoid collisions with the usage of subdirectories.

It is recommended to use the file $HOME/motd (for "Message Of The Day") to place a note that would be of importance to the other users of the build directory. This file is displayed immediately after the logon and before the first prompt.

An example is shown below:

```
System/DB:      Orac7.  Orac7.1 Orac7.2 Inf5 Inf7 DB2v1 DB2v2 Syb4.9 Syb1
 oem-ibm   AIX3                                  X          X                  X        X
 carcps 8 AIX3    X        X-1
 oem-ppc1 AIX4             X                                        X
 oem-ppc3 AIX4                                          X          X
 oem-sn 1 solaris        X                                                  X        X
 oem-sn 3 sunOS   X                              X                                X
 oem-hp 2 V9       X                             X
 oem-hp 4 V9               X-2                   X
 oem-hp 3 V1              X                            X
Notes 1: Database is available, but not currently running
        2: Database is installed, but not configured.

Release: cs23
          cmvc23 /aix3/cs/informix5     - L  22, Mig DB2 & docs - Angel
          cmvc23 /aix3/cs/oracle7       - L  24, Austin stuff - Lee
          cmvc23 /sunos/cs/sybase49     - L  24+, bug X - JRY (oem-sn 1)
          cmvc23 /sunos/cs/oracle7      - L  24+, bug X - JRY (oem-sn 1)
          cmvc23 /hp1 /cs/informix7     - L  24+, bug X - JRY (oem-hp 3)
Release: gui23
          cmvc23 /solaris/gui/x11r5/stal   - PMRs - Angel
```

The shell script used for displaying the message of the day is shown below:

```
#!/usr/bin/ksh
#
# Edit and update the message-of-the-day (motd)

# Take last line of file and save in temporary file
outfile="/tmp/motd.build"
let x= cat $HOME/motd | wc -l | tr -d ' '
let y=$x-1
head -$y $HOME/motd > $outfile
echo Entering vi to edit current motd
echo Note: do not enter date, it will be appended automatically
sleep 2

# Edit file and append time/date stamp
chmod u+w $outfile
vi $outfile
echo Exiting motd, date is being appended
echo "   End $HOME/motd as of  date . 'motd' to update   " >> $outfile
mv $HOME/motd $HOME/motd.old

# Save old file and install new one
cp $outfile $HOME/motd
if ffl -f $outfile “
then
    rm  $outfile
else
     echo "Error, new motd not copied.    Saving $outfile"
fi

# Show results
clear
echo "Displaying new motd:"
sleep 2
cat $HOME/motd
exit

# end of file
```

Figure 2. motd: Shell script to edit and view the message-of-the-day

OVERALL PROCESS WITH CMVC

CMVC is used to manage the software configuration and the file changes for the CMVC Server and Client for UNIX. This section describes the overall process for handling the files and the different CMVC objects associated with the CMVC Server and Client for UNIX.

FAMILIES, COMPONENTS AND RELEASES

The CMVC 2.3.0 product is developed and maintained in the "toro" family. The components and releases are:

The main component for the server and the line commands in the toro family is cmvc and the release is cs230 .

The component for the tools to install the code in the toro family is called cmvcInstallp and the release is inst230 .

The component for the GUI in the toro family is called cmvcgui and the release is gui230 .

The parent component for the obsolete components and releases is called "OBSOLETE". In this way, these obsolete objects do not clutter the tree view of the active components.

The components have the default process which include the Verify subprocess for Defects and Features and the Design, Size and Review subprocess for Features (and not for Defects).

The releases have the Fix, Track and Level subprocesses during all the development phases. The Approval subprocess is activated during the times when the code is about to be prepared for verification and for packaging; in that way, the changes to the release are carefully monitored and controlled.

Instead of three separate releases, it might be possible to use only one release for all the code (line commands, GUI, installation tools), which could minimize the overhead associated with tracks and levels. However, we found that having three releases gives us also much needed flexibility for handling situations where the development activities concentrated on the server code and there were frequent code changes, and thus, we could continue at slower pace of change with the GUI.

For business reasons, the CMVC version/release 2.3.0 was the only one that we did, and we just provide full refreshes of the release; that is, the latest refresh supersedes the previous one, and there are no deltas to be delivered and applied. We identify these refreshes by using the CMVC Level as the 4th identifier from left to right; for example, our latest refresh for the code was Level 18 or refresh 2.3.0.18 . Thus, we did not encounter the situation where

we had to create a new release (such as one for version 2.4 or 3.1) and we did not have to maintain the links of files between multiple releases.

PROCESS MODEL

This section describes the development process followed by the CMVC development team to handle defects and features.

There are three important characteristics in this process:

There is only one active level at a time.

This avoids the problem of having prerequisite tracks that are in another active level (spaghetti tracks).

The criteria to commit a level is that the code can be built successfully, regardless if the function is working as desired. This guarantees that any level can be extracted and built correctly.

If the level builds fine, but if there are functional problems, then new defects are created and handled in the next level.

We try to avoid a large number of file changes and tracks in a level.

We would rather have many small levels than a single extremely large level. There are a couple of reason for this decision:

- The Level -check and Level -commit are by far the most expensive operations in CMVC, in terms of CPU usage and in exploitation of the database.
- If you wait a long time before committing a level, then the users who created defects or features will not see any verification records and the testers of your software may not see a driver built in a long time.

While reading this document, it might be helpful to have a copy of the Appendix A "CMVC State Diagram", page 72 from the CMVC Concepts manual. In this way you will be able to follow the state transitions and the relationships between the elements.

PROJECT LEAD CREATES COMPONENTS AND RELEASES

1. The project lead creates the necessary components under the family.
2. These components use the Design-Size-Review (DSR) subprocess for features, but the DSR subprocess will not be used for defects. The verify subprocess will be used for both features and defects.
3. The project lead creates under the appropriate component the relevant releases.

Each release should use the subprocesses fix, track, test and level. Depending on the stage of the release, it might be necessary to add the approval subprocess.

Originally we used the test subprocess when the team was larger and we had one tester fully dedicated to CMVC, but now that the team is smaller, we found that the verification records provide us with the opportunity to ensure that the fix indeed addressed the problem. Thus, we do not use the test subprocess anymore.

HANDLING DEFECTS AND FEATURES

1. The project lead or a team lead will assign to the developers the appropriate defects and features to be implemented.
2. The developers accept the defects and the defects move to the 'working' state.
3. The features are designed, sized, reviewed, and accepted, until eventually they will be in the 'working' state.
4. A track is created to associate each defect or feature with the appropriate release or releases.

DEVELOPERS WORK WITH FILES USING TRACKS (CODE AND UNIT TESTING)

This is the phase for Coding and Unit Testing.

1. The developers will create/check-out, compile, test, debug and check-in the files affected by the release.

   Because the release uses the track subprocess, during check-in, the developers need to specify the appropriate track.

   The existing files that are not changed as a result of the feature (but need to be included during the compilation), are extracted.

   We recommend that the user performs a release/level extract of the committed version.
2. All the components that have file changes will have a a fix record for a track.

   You can see the fix record by selecting the "Fix Records..." option from the "Windows" pulldown in the "CMVC - Tracks" window.
3. You can see the file changes in a track by selecting the "Change History" option from the "Show" pulldown in the "CMVC - Tracks" window.

PREPARING THE CODE FOR FUNCTION VERIFICATION TEST (FVT)

Once the developers decide that it is time to integrate the code and go to the Function Verification Test (FVT) phase, then the following actions will take place:

The project leader will confer with the developers to make sure that all the files that will be in FVT must have completed the unit testing.

1. The project leader will create a level for the release.

   The initial state of the level is 'working'.

2. The fix records for each relevant track need to be completed.

   Ideally, the component owner for the fix record should review the code that was changed, and then mark the fix record as completed. This action will certify that the code compiles at the individual level and that unit testing has been performed.

   However, in practice, this step is done only on difficult file changes where a second opinion is advisable.

3. When all the fix records are marked as completed, then the track is moved from the 'fix' state to the 'integrate' state.

4. For each track, a level member has to be created for the level. The level member window can be shown by selecting the "Level Members" option from the "Windows" pulldown in the "CMVC - Levels" window.

   This action will move the level to the 'integrate' state.

   This action is done by the builder, who is the person in charge of adding the tracks to the level and then extracting the level in order to compile the code.

5. Then a build is done of the level in order to produce the testing driver for the release.

   For more information, see the process document for the build activity.

6. If a problem is found during the compilation/linking of the code from the level and if a change has to be done in a file, then it is necessary to move the track to the "fix" state; in order to do that it is necessary to do the steps described in “Backing out a faulty FVT driver” on page 13.

7. Before extracting the level, it is necessary to check it in order to ensure that all file changes are present, and that there are no missing prerequisites.

   If there are missing prerequisites, then the missing tracks should be added or the appropriate tracks present in the level should be removed (in case the prerequisite track is not ready to be integrated).

8. If starting a build from scratch, then the builder will perform a full extract of the latest committed level. Then, the builder will extract the Delta of the current integrated Level.

If not starting from scratch, then a Delta of the current Level might be sufficient. In case of doubt, delete the current build tree and do a full level extract plus a delta extract.

9. If no problems were found during the build activity, then the level is committed and completed.
10. We may change the value for the level field "type" to indicate how the code will be distributed:

    Gold, if the code is going to be part of a standalone CD-ROM or participate in the IBM Software Solutions for AIX Showcase CD-ROM.
    Refresh, if the code is just going to be placed in the internal and external FTP sites for users to download.
    Beta, if the code is going to be delivered to few customers as beta code.

Backing out a faulty FVT driver

If during the preparation of an FVT driver, a problem is found, then it could be necessary to back-out the offending track, fix the files, and repeat the process. This 'backing out' action can be accomplished by following the steps below:

1. Remove the level member associated with the track that contains the file or files to be modified.

   This can be done by selecting the "Remove" option from the "Actions" pulldown in the "CMVC - Level Members" window.

2. Move the track from the "integrate" state to the "fix" state, so that the files in question can be modified.

   This can be done by selecting the "Fix" option from the "Actions" pulldown in the "CMVC - Tracks" window.

3. Activate the fix records for the track.

   This can be done by selecting the "Activate" option from the "Actions" pulldown in the "CMVC - Fix Records" window.

4. Check-out, modify the contents of the files, compile, test, and check-in the files again.
5. Complete the fix records; this action will move the track to the integrate state.
6. Go back to step 5 on page 12.

DEALING WITH TRACKS/FEATURES DURING FVT

1. Once the level moves to the "complete" state, the tracks move to the "test" state.

   In case the release does not have an environment list, then the track moves to the "complete" state.

2. Once the track is in the "test" state, then the FVT tester will test the code and will have the opportunity to accept, abstain, or reject the track.

   Originally we had a full-time tester and thus we needed the test subprocess, but now our team is smaller and we do not use the test subprocess, instead we use the verification records.

3. If no problems are encountered that are related to the track/feature at hand, then the tester will "accept" the test record for the track.

   To access the test record, select the "Test Records..." option from the "Windows" pulldown in the "CMVC - Tracks" window.

4. If the tester finds a problem, then the tester will "reject" the test record.

   Then the tester should open a defect reporting the problem. See “Dealing with defects created during FVT” on page 15.

5. Once the test records for a track have been accepted/rejected/abstained, the track moves to the "complete" state.

   This in turn moves the defect or feature to the "verify" state.

6. The originator of the defect or feature is notified that the defect or feature is now in the verify state, and then the verification record for the defect or feature needs to be accepted, rejected, or abstained.

   When the verification records are modified, the defect or feature moves into the "complete" state.

7. If the verification record is rejected, the originator MUST create a new defect or feature in order to correct the problems.

   Note: After a verification record is rejected or abstained, CMVC will NOT automatically create a new defect or feature.

DEALING WITH DEFECTS CREATED DURING FVT

1. The testers will create defects for those problems encountered during FVT.

   The component owners will be notified by CMVC of the new defects and they will handle the defects, such as accept, reject, assign.

2. For each accepted defect, the owner of the defect needs to create a track.
3. Repeat the process of adding the track to a level.

Severities for defects

The definition of the severities that are used for the defects is shown below:

1. Function can not be used. Immediate fix is necessary.
2. Function can be used but with serve restrictions.
3. Function can be used with minor restrictions, not critical.
4. Function can be bypassed, but should be corrected.

PREPARING THE BUILDS

GENERAL ASSUMPTIONS

For each operating system it is necessary to have a build host with the appropriate hardware and software.

The build for each platform/database combination has to be done in each platform (no cross-compiler available).

A single userid, build , will be used for all building activities.

The relational database has to already be installed and a database (for example, in DB2 this is called a database instance) with the name of "build" has to be created.

The task of building and packaging the complete product for all the environments is complex and prone to error due to the magnitude of the combinations. It is suggested to use the checklist described in Appendix C, “CMVC Client/Server UNIX, Build and Package Checklist” on page 63 to keep track of the progress for each build.

ONE-TIME ONLY ACTIVITIES

Create CMVC family 'build' to be used with DB2

It is necessary to create a CMVC family named 'build' on each platform in order to perform the pre-compilation steps to resolve the SQL queries with the actual database.

This CMVC family must have the following characteristics:

login: build
gid: <same as the gid for the db2instance>

The database name 'build' is used in the Make.<database>.<platform> make scripts.

After the database has been created:

1. Execute "db2 connect to build"
2. The results should be:

```
   Database Connection Information

 Database product           = DB2/6    1.1.
 SQL authorization ID       = BUILD
 Local database alias       = BUILD
```

3. Execute "db2 connect reset"

Create necessary links

In order to have a common interface for the build process, in some cases it is necessary to provide symbolic soft links between the actual path for the databases/tools and the paths used inside the MAKE.cmvc and related files.

To do these links, it is necessary to execute the following command as root:

```
ln  -s  <actual-path>  <desired-path>
```

AIX

Host oem-ibm and database sybase49:

```
ln  -s  /home/sybase/sybase49  /usr/db/sybase49
```

Host carcps08 and database oracle7.0:

```
ln  -s  /db/oracle/product/7. .13  /usr/oracle7
```

Host oem-ppc1 and database oracle7.1:

```
ln  -s  /home/oracle71  /usr/oracle71
```

Host carcps08 and database informix:

```
ln  -s  /db/informix  /usr/informix
```

HP-UX

Host oem-hp03 and database oracle7:

```
ln  -s  /cmvc_db/u2/oracle/product/7. .12  /usr/oracle7
```

Host oem-hp03 and database informix:

```
ln  -s  /users/informix  /usr/informix
```

Host oem-hp02 and database informix:

```
ln  -s  /users/informix  /usr/informix
ln  -s  /users/informix  /usr/db/informix
```

SunOS

Host oem-sn03 and database sybase49:

```
not needed
```

Host oem-sn03 and database oracle7:

```
ln   -s   /home/oracle/product/7. .12   /usr/oracle7
```

Host oem-sn03 and database informix:

```
ln   -s   /home/informix   /usr/informix
```

Solaris

Host oem-sn01 and Softbench:

```
ln   -s   /opt/softbench   /usr/softbench
```

Host oem-sn01 and database sybase49:

```
ln   -s   /export/home2/sybase/sybase49   /usr/db/sybase49
```

The Oracle 7.1.6 code for Solaris is installed in a system that is a local drive in oem-ibm, thus, it is necessary to mount it:

```
mount oem-ibm:/home/lv 3      /export/home3
```

Host oem-sn01 and database oracle7:

```
ln   -s   /export/home3/ora716      /usr/oracle7
```

CREATION OF THE BUILD/SHIP/PACKAGE TREE

To create the top portion of the Build/Ship/Package tree (to be called just "Build tree") as shown in “Top directory structure - created via shell script” on page 23, do the following:

1. Login in as 'build'.
2. Create the directory that will complete the specification of TOP, such as 'cmvc230'. In this way the $TOP will be $HOME/cmvc230.
3. Change to the directory 'bin'.
4. Use CMVC to extract the file "mkbldtree" from release "tools230".
5. Create the directory for the Version-Release of CMVC, such as 'cmvc230'.
6. Change to the directory 'cmvc230'.

   The shell script will create the build tree from the current directory.
7. Execute the shell script as follows:

mkbldtree

ESTIMATED TIME FOR BUILD AND PACKAGING

The average approximate time involved in a single complete build is shown below. However, the compilation times may range from 2 hours in fast machines such as oem-hp04 to 5 hours in slow machines such as oem-ibm.

Furthermore, this average time should be multiplied for each combination of platform and database to be built.

This time assumes that this is not the first build, and assumes that all the hardware and software is operational, and that the one-time activities have been performed already. The breakdown is shown below (in hours)

Extract the installation/source files: .5
Perform the build: 4.
Package the files: 1.
Perform the installation test: 1.

Estimated time for all combinations

The approximate time involved in a complete build for all the combinations of platform/database is:

1 full-time week (4 hours)

BUILDING THE CODE

OVERVIEW OF THE WHOLE PROCESS

It takes approximately one full working week to do a complete build from scratch for a single CMVC level, such as 2.3.0.21.

As we add more platforms (such as AIX 4 and HP-UX 10) and more databases (Oracle 7.1 and Informix 7 and DB2 V2), the whole process will keep getting larger and larger, and taking more disk space everywhere. This time frame gives the luxury to do barely any testing at all: no installation test, no regression tests, no sanity tests.

The status checklist for the builds, Appendix B, “CMVC for UNIX, build and package status checklist” on page 61 is the "score card" or "cheat sheet" that was used in level 2.3.0.21 and you can appreciate the enormous amount of work that is involved here, and why it is so time consuming to prepare "special beta" versions for demanding customers, because these versions take a long time to prepare and usually is a throw away effort that cannot be reused, and many times we do not have file space to keep around the old version (non-beta) of an install file and the beta version of the install file demanded by the customer.

Furthermore, it is difficult to fully automate the process because of many limitations: network problems, disk space, manual specification of the options to build, etc.

The CMVC Level is NOT committed until all the builds are done. Once all the compilations in all the platforms are clean, then the level can be committed. This means that if a bug is encountered after the level is committed, a defect will be opened for the next level. The reason for this is to guarantee that a full extract of a level is buildable in all platforms, even though the function itself may not be correct.

The whole sequence for the complete preparation for a single level is summarized below and is further explained in detail in other sections in this document:

1. Tell other CMVC developers to stop using the affected directories and to do a checkin or a copy to another directory of any files that are important to them.
2. Delete "rm -fr" the appropriate top directory, such as cmvc230/aix4/cs/db2
3. Do a full extract of the latest committed level, such as 020.
4. Do a delta extract of the current non-committed level, such as 021.
5. Login as build into the specific system to use, such as "oem-ppc1" for AIX 4 for DB2 V2.

6. Modify manually "qmake" to use the appropriate options, such as normal, db2v2, aix4, and all (both server and client).
7. Start "qmake" and monitor its progress.

   There are some compilations that usually fail and require manual intervention, such as Solaris and SunOS. The workarounds are known and easy to apply manually, but there is no way to automate them.
8. After the compilation is done, then do "qmake ship" to copy the shippable code into the "ship" directory.
9. Repeat the extract and build process for the GUI, for both the standalone and the integrated version of the client.
10. Also in AIX 3 it is necessary to build the GUI for X11R4 and X11R5, and in HP-UX 9 it is necessary to build the GUI for X11R4 and X11R5.
11. As root, change directory to /home/build/cmvc230/installp and kickoff the "make -f Makefile.<platform>.<option>" to create the appropriate tar file.
12. After ALL the servers and the normal client for a platform are done, then prepare the Non-Netls version of the client. This requires a manual change in qmake (only the client is needed) and repeat the steps for 'qmake clean', 'qmake' (build) and 'qmake ship'.
13. Login as root again to create the tar file for the Non-Netls client.
14. After all the tar files for servers, normal client and Non-Netls client are done, create backup tapes (using the non compressed files).
15. For AIX (both AIX 3 and AIX 4) we have to prepare also installp smit images, which are done by using "bld.all.images".
16. After all the install smit images are done, then create backup tapes (using the non compressed files).
17. Now it is time to copy the files into:

    our internal FTP site (keithp.raleigh.ibm.com)

    replace the CM23AIX and CMVCIUO packages in CMVCBETA

    the IBM external FTP site.

    In order to do this step, it is necessary to compress all the tar and installp smit files.
18. FTP to host keithp as userid cmvcsupt, change the directory to /home/ftp/pub/cmvc and then change to the appropriate directory.
19. FTP to the staging directory in Boulder and follow the appropriate instructions to place the code there. It takes around 24 hours to be copied in to the real external FTP site.

20. If participating in a CD-ROM for AIX Showcase, then it is necessary to login as userid build in rtplib02, then do cdromv4.klog to obtain the token for the AFS Toronto cell and then cdromv4.<platform>.copy for each platform to be used. This step may take 1 full working day, because the files in Toronto MUST be non-compressed.

LOCATION OF THE SOURCE CODE

The source code for the CMVC Server/Client for UNIX, Version 2.3.0, is in the "toro" family:

```
family:    toro
release:   cs23        -> server and line commands
release:   gui23       ->  GUI
release:   inst23    ->  packaging  tools  in  /home/build/cmvc23 /installp
release:   tools23   -> miscellaneous tools in /home/build
```

BUILD/SHIP/PACKAGE DIRECTORY STRUCTURE

Top directory structure - created via shell script

The top portion of the Build/Ship/Package directory structure is shown below and is manually created via the shell script 'mkbldtree' from the release tools230 (see section “Creation of the Build/Ship/Package tree” on page 19).

```
$HOME/
        bin/                    general tools from release tools23
        cmvc23 /
                <operating-system>/
                    cs/
                                <database>
                    gui/
                                stal/
                                intg/
```

Figure 3. Top directory structure - created manually via script

Source directory structure - created by extracting release 'cs230'

The following leaves of the directory structure tree are created when performing a Release/Level extract, and contain the actual source code that will be built.

```
$TOP/
        bin/
        samples/
        src/
            adm/
            archive/
            cat/
            daemon/
            inc/
            lib/
                  etc/
                  net/
                        packet/
                        request/
                                    file/
                                    ni/
                                        client/
                                        server/
                        session/
            restore/
            tools/
                     common/
                     common1/
                     informix/
                     oracle7/
                     sybase/
                     xwind/
             ui/                                        Command Lines
                simple/
                           cmd/
                           etc/
                           flags/
```

Figure 4. Source directory structure - created by release 'cs230' (1 of 2)

```
server/
        db/
            common/
            informix/
                        def/
                        gen/
                        init/
                        src/
            oracle7/
                        def/
                        gen/
                        init/
                        src/
            sybase/
                        def/
                        gen/
                        init/
                        src/
             xwind/                          for DB2
                        def/
                        gen/
                        init/
                        src/
        etc/
        tx/
        vc/
```

Figure 5. Source directory structure - created by release 'cs230' (2 of 2)

Notes:

1. The $TOP used in this document is:

   $HOME_FOR_BUILD/<cmvcVRM>/<platform>/cs/<database>

   where:
   <cmvcVRM> is 'cmvc23 ' for 2.3.
   <platform> is 'aix3', ├aix4 , ├hpux9 , ├hpux1 , ├sunos , 'solaris'
   <database> is db2, informix, oracle7, sybase49

   example for AIX 3 with DB2 on 'oem-ibm':
   /home/build/cmvc23 /aix3/cs/db2

Install directory structure - created by extracting release 'inst230'

The following leaves of the directory structure are created when performing a Release extract of 'inst230'. (see section “Extraction of inst230 using Release or Level Extract” on page 34).

```
$HOME/
        cmvc23 /
                installp/
                                com/
                                        cmvccltintg/
                                        cmvccltstal/
                                        cmvccltstalmEn_US/
                                        cmvcsrvdb2/
                                        cmvcsrvinfx/
                                        cmvcsrvmEn_US/
                                        cmvcsrvora7/
                                        cmvcsrvorac/
                                        cmvcsrvsyb/
                                hpux/
                                makecdrom/
                                nls/
                                sun/
                                solaris/                                placeholder
```

Figure 6. Bottom directory structure - created by release 'inst230'

Bottom directory structure - created by doing make with 'ship'

The following leaves of the directory structure are created when performing a make while specifying the 'ship' action (see section “Gathering the shippable files (ship step)” on page 40).

```
$TOP/ship/
        client/
                cmvc/
                        usr/
                                lib/
                                        nls/
                                                msg/
                                                        En_US
                                lpp/
                                        cmvc/
                                                bin/
                                                samples/
                cmvcsrv/
                        usr/
                                lib/
                                        nls/
                                                msg/
                                                        En_US
                                lpp/
                                        cmvc/
                                                bin/
                                                bind/
                                                install/
                                                samples/
                install/
                        r2/
        server/
```

Figure 7. Bottom directory structure - created by doing make with 'ship'

ACTIVITIES TO BE DONE PER EACH VERSION, RELEASE OR MODIFICATION

There are some activities that need to be done only once per each version, each release or each modification, and there is no need to repeat them for every build.

Files that need to be modified due to changes in Copyright or VRM

This section lists the files that need to be modified when there are changes to the Copyright notice (such as a new year) or to the Version-Release-Modification (VRM) number. It is also applicable for the information on the service level, which is included in the source files as an static char that has the strings "@(#)"; in that way a what command can be executed on the executable files to find out this service information, such as:

```
cd /usr/lpp/cmvc/bin
what cmvcd | grep cmvc
```

The result may be:

```
   cmvc        CMVC Version 2.3. .23, (C) Copyright IBM Corp.,1993,1996
   cmvc        5765-2 7, 5765-2 2, 5765-397, 5622- 63
  cmvc        96/ 8/28  SID=1.26.1.52
   cmvc        Platform: AIX 4
  cmvc        Database:  Informix
```

In order to incorporate these strings we did the following:

The include file "whatinfo.h" was created that contains all the static strings with @(#) and the keyword "cmvc". Based on Define symbols passed to the compiler, the proper operating system or database string was activated.

A condensed version of this file is shown in Figure 8 on page 29.

```
char copyrightInfoffl“
    = "@(#) cmvc        CMVC Version 2.3. .24, (C) Copyright IBM Corp.,1993,1996";
char productInfoffl“
    = "@(#) cmvc        5765-2 7, 5765-2 2, 5765-397, 5622- 63";

/  Determine the platform  /

#ifdef AIX4
char platformInfoffl“
    = "@(#) cmvc        Platform: AIX 4";
#endif

/  Determine the database  /

#ifdef informix
char databaseInfoffl“
    = "@(#) cmvc        Database: Informix";
#endif
```

Figure 8. Header file with service keywords for the what command

The include file "whatinfo.c" was created that contains the code that uses these static strings.

Some compilers will strip these static variables if they are not used in the program, thus, we had to create a dummy section of code that does not do anything practical, but because the variables were actually used, then the compiler will not strip them.

A condensed version of this file is shown in Figure 9 on page 30.

```
int       dummy;
 char   dummyC;
 char   dummyP;
 char   dummyS;
 char   dummyL;
 char   dummyD;

 dummyC = copyrightInfo;
 dummyP = productInfo;
 dummyS = serviceInfo;
 dummyL = platformInfo;
 dummyD = databaseInfo;

 if (dummy == -7 77)
{
       printf("%s%s%s%s%s\n", dummyC, dummyP, dummyS, dummyL,
                      dummyD);
}
```

Figure 9. Source file with service keywords for the what command

In the main file for each executable, we did the following:

- Included the following declaration of a static string that contains CMVC expandable keywords in order to place the date when the main file for the executable was extracted, and the version id associated with that file:

  ```
  / Support Information /
  char serviceInfoffl‘ = "@(#) cmvc         %D% SID=%I%";
  ```

- Included the header file whatinfo.h

  ```
  / Include the header file with the information for the 'what' command /
  #include "whatinfo.h"
  ```

- Included in the main routine, immediately after the declaration of the variables and before any executable statement, the code file whatinfo.c

  ```
  / Include the source file with the information for the 'what' command /
  #include "whatinfo.c"
  ```

Release "cs230"

The following files must be changed in release "cs230" (the actual code) in order to update the Copyright information and the Title:

*.c and *.h

src/inc/whatcmd.h
src/inc/whatinfo.h
src/cat/cmvc.msg

Readme files:

src/adm/README.what.is.new

Release "gui230"

The following files must be changed in release "gui230" (the actual code) in order to update the Copyright information and the Title:

source code

cmvc.c

message catalog

cmvchelp.msg

product id to be shown in Softbench

hp/intg/softcmvcprod
ibm/intg/softcmvcprod
sun/intg/softcmvcprod
solaris/intg/softcmvcprod

Read me files

hp/README.gui
ibm/README.gui
sun/README.gui
solaris/README.gui

Release "inst230"

The following files must be changed in release "inst230" (the files for packaging) in order to update the Copyright information and the Title:

Shell scripts

bld.all.images.3
bld.all.images.4
aix3/cmvcinstall
aix3/tarinstall.aix3
aix4/cmvcinstall
aix4/tarinstall.aix4
hp1 /cmvcinstall
hp1 /tarinstall.hpux1
hp9/cmvcinstall
hp9/tarinstall.hpux9
solaris/cmvcinstall
solaris/tarinstall.solaris
sun/cmvcinstall
sun/tarinstall.sunos
makecdrom/cmvcinst
makecdrom/cmvcinst.hpux1
makecdrom/cmvcinst.hpux9

Files that need to be modified when adding new files

At this stage, if there are new files to be packaged they will most likely be:

Documentation files in src/adm/README.*. Thus, only the src/adm/Makefile needs to be updated.

Samples in samples/*. Thus, the samples/Makefile and either samples/client.samples or samples/server.samples need to be updated.

PREPARATION FOR THE BUILD

This section is only applicable for the person who is responsible for the production build on the build machine.

For the target builds where the "build" directory is exported from rtplib02, you can perform all the extracts and the file checkout/checkin activities from build@rtplib02 and all the builds from build@target_server.

1. Login as "build" to the target server.

   The home directory is "/home/build". This directory resides actually in rtplib02 and is exported and mounted in the server.

Note: We cannot do incremental builds for CS (Client/Server), because there is not a good way to keep track of the dependencies with the current structure of the make files. This is the reason for a complete cleanup of the build-tree for each build.

Process Tracks, Levels and Level Members

Ensure that there is a level with all the appropriate tracks. For more details, see “Process model” on page 10.

Extraction of the tools for the build and packaging process

Source code extraction requires some knowledge of CMVC operations. Users are suggested to refer to CMVC manual "Understanding IBM AIX CMVC/6000 Concepts" for details.

Extraction of tools230 using Release or Level Extract

The release 'tools230' contains utilities to be included in the $HOME/bin directory that will help to perform tasks such as copying the code from the ship tree into the final ('real') directories for the server, client and gui code. This procedure will allow us to have a setup for testing the code, using the directories that the end-users will use.

You can use a working version of any CMVC client to extract the source files as follows:

1. Start the CMVC client with the family **toro**.
2. Bring up the Release window and select the release **tools230**.

   An alternative way is to bring up the Level window and select the desired level.
3. Select the action "extract" and specify:

   For releases, choose to extract the committed versions of the files

   Choose the option "Current versions" if this is the first build of the release (in case there are no committed versions yet).

   For levels, choose to extract either Delta or Full.

   Specify the Destination (we use a local extract, without involving NFS):

   host: nomount
   directory: /home/build/bin

   Specify the File and Directory Access:

   User number (UID): 21
   Group number (GID): 222
   File permissions:
   Directory permissions: 777

   Choose to expand the keywords (this is the default).

   Choose NOT to perform the CRLF conversion (this option is only available from the DOS/Windows and OS/2 clients).

4. Select OK to start the extraction of the files for the release.

Extraction of inst230 using Release or Level Extract

The release 'inst230' contains utilities to be included in the $HOME/cmvc<VRM>/installp directory that will help to perform tasks such as preparing the installable images for CD-ROM and tape for all the combinations.

You can use a working version of any CMVC client to extract the source files as follows:

1. Start the CMVC client with the family **toro**.
2. Bring up the Release window and select the release **inst230**.

   An alternative way is to bring up the Level window and select the desired level.
3. Select the action "extract" and specify:

   For releases, choose to extract the committed versions of the files

   Choose the option "Current versions" if this is the first build of the release (in case there are no committed versions yet).

   For levels, choose to extract either Delta or Full.

   Specify the Destination (we use a local extract, without involving NFS):

   host: rtplib 2
   directory: /home/build/cmvc/installp

   Specify the File and Directory Access:

   User number (UID): 21
   Group number (GID): 222
   File permissions:
   Directory permissions: 777

   Choose to expand the keywords (this is the default).

   Choose NOT to perform the CRLF conversion (this option is only available from the DOS/Windows and OS/2 clients).
4. Select OK to start the extraction of the files for the release.

BUILDING THE CMVC SERVER AND LINE COMMANDS

This chapter describes how to build the CMVC Server and Line Commands, where to obtain the source code, what is the directory structure used for the operations, and what is the sequence of these operations.

COMPLETE CLEANUP OF THE BUILD TREE

This type of cleanup is intended for those occasions when it is necessary to start from an empty build directory:

1. Change to the $TOP directory
2. Execute the following command to cleanup all the subdirectories:

   rm -fr

Cleanup of the previous build

In case there is no need to extract the source code again, but is necessary to perform another build, then you can follow the steps below to trigger a new build, ensuring that all the files will be compiled.

1. Change the directory to the TOP directory, such as $HOME/cmvc230
2. Change the directory to the desired platform-type-database, such as "aix3/cs/db2".
3. Run qmake clean
4. If you are going to do the ship step later on, you should issue the following command:

   rm -rf ship

EXTRACTION OF THE SOURCE FILES

Source code extraction requires some knowledge of CMVC operations. Users are suggested to refer to CMVC manual "Understanding IBM AIX CMVC/6000 Concepts" for details.

Extraction using Release or Level Extract

Note:

It is necessary to perform an individual source extraction for each entry in the build tree that has "cs" in the Path, that is, a separate source extraction will have to occur in order to build the client/server for each platform. Refer to Figure 3 on page 23 to see each of the platforms.

You can use a working version of any CMVC client to extract the source files as follows:

1. Start the CMVC client with the family **toro**.
2. Bring up the Release window and select the release **cs230.**.

   An alternative way is to bring up the Level window and select the desired level.
3. Select the action "extract" and specify:

   For releases, choose to extract the committed versions of the files

   Choose the option "Current versions" if this is the first build of the release (in case there are no committed versions yet).

   If you are doing Unit Testing or Function Testing you may specify to extract the current versions.

   For levels, choose to extract either Delta or Full.

   Specify the Destination (we use a local extract, without involving NFS):

   host: rtplib 2
   directory: /home/build/cmvcVRM/x/y/z

   where
   VRM is the Version and Release number, such as: cmvc23

   x is aix3, aix4, hpux9, hpux1 , sunos, solaris
   y is cs or gui
   z is for the database (only if building client/server):
   oracle7
   informix
   sybase49
   db2

   Specify the File and Directory Access:

   User number (UID): 21
   Group number (GID): 222
   File permissions:
   Directory permissions: 777

   Choose to expand the keywords (this is the default).

4. Select OK to start the extraction of the files for the release.

BUILDING THE CODE

The utility qmake in the $TOP directory is a front end to the MAKE.cmvc shell script which is really the build shell script.

To build the code follow these steps:

1. Change the file permissions for qmake to be 755:

   chmod 755 qmake

2. Edit qmake and select the proper options for platform, database, buildOption and compileOption.
3. Execute qmake
4. The output of the build will be stored in a file named nohup. .
5. You will see an appropriate message to indicate that the build has finished.
6. You can kill the build process by doing Control-C.

MISCELLANEOUS NOTES

Notes on the MAKE.cmvc shell script

The MAKE.cmvc in the $TOP directory is used to initiate the build and ship activities. The format and the options for this command are shown below:

```
Usage:    MAKE.cmvc -p platform <-d dataBase> <-b buildOption>
                    <-c compileOption> <action>

Where:
      -p platform          Enter: aix3, aix4, hpux9, hpux1 , sun, solaris
      -d dataBase          Enter: oracle7, informix, sybase4, db2

      -b buildOption       Enter: all, server, client

      -c compileOption Enter: normal, debug, FVT, noNetLS

      action                Enter additional action such as 'clean' or 'ship'
```

Use MAKE.cmvc to set the correct environment variables for the Client/Server build.

If you would like to leave the job running overnight, or have it running in background, it is recommended that you use the 'nohup' command as follows:

```
nohup MAKE.cmvc    -p aix3    -d db2    -b all    -c normal      clean > nohup.out
&
```

This will cause a file called 'nohup.out' to be generated. If you would like to see the progress of the build, you can look at the file that is being generated by issuing the command:

```
tail -f nohup.out
```

Once the build ends, you can Ctrl-c the 'tail' process in order to get back to the prompt.

FILES AFFECTED BY DIFFERENT MAKE.CMVC -C OPTIONS

In order to clarify what is the impact of compiling with different options the CMVC product, Figure 10 and Figure 11 on page 39 lists the files (from release cs230 and level 004) which are affected (their names are relative with respect to the TOP directory):

BETA option:
src/daemon/cmvcd.c
MAKE.cmvc

DEBUG option:
src/daemon/cmvcd.c
MAKE.cmvc

FVT option:
src/lib/server/etc/notify.c
src/daemon/cmvcd.c
MAKE.cmvc

NOFORK option:
MAKE.cmvc

IBMIUO option:
src/ui/simple/etc/netls.c
MAKE.cmvc

Figure 10. Files affected by different MAKE.cmvc -c options, part 1

NO_NETLS option:
src/ui/simple/cmd/Access.c
src/ui/simple/cmd/Approval.c
src/ui/simple/cmd/Approver.c
src/ui/simple/cmd/File.c
src/ui/simple/cmd/Component.c
src/ui/simple/cmd/Defect.c
src/ui/simple/cmd/Environment.c
src/ui/simple/cmd/Host.c
src/ui/simple/cmd/Size.c
src/ui/simple/cmd/Report.c
src/ui/simple/cmd/Notify.c
src/ui/simple/cmd/Test.c
src/ui/simple/cmd/Track.c
src/ui/simple/cmd/User.c
src/ui/simple/cmd/Coreq.c
src/ui/simple/cmd/Verify.c
src/ui/simple/cmd/Fix.c
src/ui/simple/cmd/Level.c
src/ui/simple/cmd/LevelMember.c
src/ui/simple/cmd/Feature.c
src/ui/simple/cmd/Release.c
src/ui/simple/cmd/Migrate.c
src/ui/simple/etc/netls.c
MAKE.cmvc

Figure 11. Files affected by different MAKE.cmvc -c options, part 2

Note: The DEBUG option uses also NO_NETLS and NOFORK

Making a DEBUG driver

To make the Client/Server with DEBUG compile options, it is necessary to use the -c flag for MAKE.cmvc as follows:

```
MAKE.cmvc  -p  r2  -d  db2  -b  all  -c  debug
```

In MAKE.cmvc, the NO_NETLS define is added in the debug driver. Thus, if you want to debug the NetLS portion of the code, then you need to modify temporarily the MAKE.cmvc file by eliminating the -DNO_NETLS from the appropriate section. Furthermore, it is important not to exclude the NetLS libraries, by eliminating the following lines that disable the NetLS libraries:

```
NETLS_LIBS=
NCS_LIBS=
```

Making an FVT driver

To make the Client/Server for FVT, it is necessary to use the -c flag for MAKE.cmvc as follows:

```
MAKE.cmvc  -p  r2  -d  db2  -b  all  -c  FVT
```

The following files are affected by the FVT compile option:

```
MAKE.cmvc
src/lib/server/etc/notify.c
```

Note: Because only the notify.c source file is affected, the client code that is built with this option is NOT affected.

Making a BETA driver

By specifying a BETA driver, the ONLY difference from a normal driver is that the 'what' service information will say 'This is BETA code'.

The objective of a beta driver is to be provided when a level has not yet been committed, such as when providing a temporary fix to a customer.

Making a Gold driver

To provide executable code that is the Gold driver, it is necessary to use the -c flag for MAKE.cmvc as follows:

```
MAKE.cmvc  -p  aix3  -d  db2  -b  all  -c  normal
```

Note: Be sure to change the type of the level to "gold".

GATHERING THE SHIPPABLE FILES (SHIP STEP)

Upon successful completion of the build step, run qmake ship to copy the files that need to be delivered to the customers from the appropriate build directory to the ship directories.

Then we could package the code in the ship directories into the installable images (in tar format or installp-smit) for the installation and verification tests.

BETA INSTALLATION

You do this step in lieu of the last step in the build and ship section.

You can copy the stuff from the ship directory into the normal directories (that is, /usr/lpp) by doing the following:

1. Logon as build.
2. Change to the bin directory
3. Change the user to root.
4. Issue the following command:

   ./copy.server <top_directory_where_ship_resides> <db>
   ./check.server <top_directory_where_ship_resides>
   ./copy.client <top_directory_where_ship_resides> <db>
   ./check.client <top_directory_where_ship_resides>

   example for SUN with Sybase 4,9:
   ./copy.server $HOME/cmvc23 /sunos/cs/sybase49 sybase49

   The output of the 'check' scripts should be piped to more.
5. Exit from root.

Known building problems

This section contains a compilation of the errors found during the building process.

If the build fails with generic problems

Here are things to check if a build fails. This list is operating system dependent.

Prior to running the MAKE.cmvc with a 'ship' action, from the /src/tools/xwind directory, issue: rm *.o *.c

After touching the message catalog (src/cat/cmvc.msg), do a MAKE.cmvc with a 'clean' action and rebuild.

If the build fails for AIX

Not applicable.

If the build fails for HP-UX

In HP-UX for client/server during "qmake ship":

```
cpp: "execute.c", line 1: error 4 36: Can't open include file
'sqlhdr.h'.
   error code 1
```

To solve the problem go to the corresponding directories and touch all the object files:

```
cd src/lib/server/db/informix/gen
touch  .o
cd src/lib/server/db/informix/src
touch  .o
```

Then restart the build.

If the build fails for SunOS

If the build of the src/daemon fails:

```
ld: Undefined symbol
    _dbgAnyApproverExists
    _dbgApproverAdd
   Error code 1
```

Do the following:

- Go to the $TOP/src/lib/server/db/sybase/gen directory and get a list of all .o files that have a length of 0 bytes.
- NOTE: Do not erase all the *.o files.
- Pre-compile, using CPC, that list of files
- For Sybase, touch the .cp file that corresponds to the .o file that has a length of 0 bytes.

  For sybase, extension: cp

  For oracle, extension: x

  For informix, extension: ec
- Try the build again.

If the build seems to take a long time for SunOS (oem-sn03), see if you can telnet to carsun01 which has the server for the acc licenses, to ensure that the controller for the LAN bridge is still operational.

If you cannot telnet, then go to the OEM lab and turn the controller 8209 off and then turn it on again.

If the build fails for Solaris

If the build seems to take a long time for Solaris (oem-sn01), see if you can telnet to carsun01 which has the server for the acc licenses, to ensure that the controller for the LAN bridge is still operational.

If you cannot telnet, then go to the OEM lab and turn the controller 8209 off and then turn it on again.

Known Problems when Testing the code

In the Solaris/SunOS system, if the cmvcd fails to run due to a semaphore problem, then do the following:

1. su as root
2. issue the following command to see the semaphores:

   ipcs -s

3. issue the following command to kill a semaphore:

   ipcrm - <id of the semaphore shown with ipcs -s>

BUILDING THE CMVC GUI

BUILD/SHIP/PACKAGE DIRECTORY STRUCTURE FOR THE GUI

The Build/Ship/Package directory structure for the GUI is shown below. This tree is automatically created when performing a Release/Level extract. However, the $TOP portion of the tree is manually created.

```
$TOP/
        config/
        help/
        hp/
                intg/
                stal/
        ibm/
                intg/
                stal/
        menus/
                 Softdm/
                          CM/
                              Softcmvc/
        solaris/
                intg/                                          placeholder
                stal/
        sun/
                intg/
                stal/
```

The following notes clarify important details:

The $TOP used in this document is:

$HOME_FOR_BUILD/<cmvcVRM>/<platform>/gui/<x11>/<mode>

where:
<cmvcVRM> is 'cmvc23 '
<platform> is 'aix3', ├aix4 , ├hpux9 , hpux1 , ├sun , ├solaris
<x11> is 'x11r4' (Motif 1.1), 'x11r5' (Motif 1.2)
<mode> is 'stal' (for stand-alone) and 'intg' (for integrated)

example for Solaris, standalone:
/home/build/cmvc23 /solaris/gui/x11r5/stal

PREPARATION FOR THE BUILD

This section is only applicable for the person who is responsible for the production build on the build machine.

For the target builds where the "build" directory is exported from rtplib02, you can perform all the extracts and the file checkout/checkin activities from build@rtplib02 and all the builds from build@target_server.

1. Login as "build" into the target server.

   The home directory is "/home/build". This directory resides actually in rtplib02 and is exported and mounted in the server.

Source code extraction requires some knowledge of CMVC operations. Users are suggested to refer to CMVC manual "Understanding IBM AIX CMVC/6000 Concepts" for details.

COMPLETE CLEANUP OF THE BUILD TREE

This type of cleanup is intended for those occasions when it is necessary to start from an empty build directory, such as after many changes have been done to the source files. If minor/few changes were done to the source files, this cleanup action might not be necessary.

1. Change to the $TOP directory
2. Execute the following command to cleanup all the subdirectories:

   ```
   rm -fr
   ```

EXTRACTION OF THE SOURCE FILES

Extraction using Release or Level Extract

Note: It is necessary to perform an individual source extraction for each combination of <platform><gui><x11><stal/intg>.

You can use a working version of any CMVC client to extract the source files as follows:

1. Start the CMVC client with the family **toro**.
2. Bring up the Release window and select the release **gui230**.

   An alternative way is to bring up the Level window and select the desired level.
3. Select the action "extract" and specify:

For releases, choose to extract the committed versions of the files.

Choose the option "Current versions" if this is the first build of the release (in case there are no committed versions yet).

If you are doing Unit Testing or Function Testing you may specify to extract the current versions.

For levels, choose to extract either Delta or Full.

Specify the Destination (we use a local extract, without involving NFS):

Specify the File and Directory Access:

```
User number (UID):          21
Group number (GID):         222
File permissions:
Directory permissions:    777
```

Choose to expand the keywords (this is the default).

4. Select OK to start the extraction of the files for the release.

BUILDING THE CODE

Cleanup of object modules and message catalogs

This type of cleanup is intended for a complete rebuild using the code that already exists in the build tree (that is, for incremental development). This is useful also when changing the message catalog, in order to ensure that all the code is compiled with the latest help.c file.

1. Change to the $TOP directory, such as:

   ```
   cd $HOME/cmvc23 /aix3/gui/x11r5/stal
   ```

2. Execute the shell script **clean.gui** to cleanup the object modules and the message catalog.

CONSIDERATIONS BEFORE BUILDING

It is very important to take into account the following considerations before building the code.

Making a DEBUG driver

So far, there is no provision in the makefiles to specify to compile with the debug option.

Making an FVT driver

In contrast with the Client/Server code, there is no provision and no need for an FVT version.

Making a BETA driver

To provide executable code that has a BETA identifier when performing the UNIX command "what cmvc | grep cmvc", invoke the make tool with the following string:

```
make -f solaris/stal/Makefile BETA=-DBETA
```

The BETA identifier looks like:

```
cmvc            This is BETA code       Date: 94/ 6/28
```

Only "cmvc.c" and "migResources.c" need to be compiled with the BETA flag. This means that it might be possible to first perform a normal compile, then touch these 2 files and then invoke the make command again but this time by adding the BETA flag as indicated.

Note: Be sure to change the type of the level to "beta".

Making a Gold driver

To provide executable code that is the Gold driver, do not specify any additional strings.

Note: Be sure to change the type of the level to "gold".

BUILDING THE GUI

1. Change to the $TOP directory, such as:

   ```
   cd $HOME/cmvc23 /aix3/gui/x11r5/stal
   ```

2. Change the permissions to qmake to 755:

   ```
   chmod 755 qmake
   ```

3. Edit qmake and specify the platform, the mode and the X11 version. This script invokes the proper make file.

   ```
   make  -f  <platform>/<mode>/Makefile<.x11r5>
   ```

   Where

   ```
   <platform> is:   ibm, solaris, sun, hp
   <mode>  is:          stal,  intg
   <.x11r5> is needed if building for x11r5
   ```

For X11R4:
```
make   -f  ibm/stal/Makefile
```

For X11R5:
```
make   -f  solaris/stal/Makefile.x11r5
```

Examples for making a beta:
```
make    -f ibm/stal/Makefile BETA=-DBETA
make    -f solaris/stal/Makefile.x11r5 BETA=-DBETA
```

4. It is recommended that if you want to leave the job running overnight or to have it running in the background, use the command 'nohup' as follows:

   For X11R4:
   ```
   nohup      make -f solaris/stal/Makefile      > nohup.out &
   ```

   For X11R5:
   ```
   nohup      make -f solaris/stal/Makefile.x11r5      > nohup.out &
   ```

   This will result in a file named 'nohup.out' to be generated. If you want to see the progress of the build, you can peek at this nohup.out file that is being generated by issuing the command:

   ```
   tail -f nohup.out
   ```

   Once the build ends (the link step for the 'cmvc' executable), you can ctrl-c the process for 'tail', in order to get back to the prompt.

KNOWN BUILD PROBLEMS

If the cmvchelp.msg or the cmvcui.msg files (message catalogs) have syntax errors, which are most likely to be a lack of '\n\' at the end of the sentences, you will see the following error message when compiling the code (such as):

```
$ make -f solaris/stal/Makefile.x11r5
or
$ make -f ibm/stal/Makefile

LANG=C; export LANG; \
gen_msg cmvchelp cmvchelp.msg > cmvchelp.msg1; \
gencat cmvchelp.cat cmvchelp.msg1; \
rm cmvchelp.msg1;
   Error code 139
make: Fatal error: Command failed for target  cmvchelp.cat'
```

If the *.msg message catalogs have a double quote inside the help strings, then the compilation of help.c will fail.

COPYING THE OUTPUT FILES INTO THE SHIP DIRECTORY

It is not necessary to perform a "ship" step (as with the Client/Server code) to copy the output files into a ship directory. In a sense, the current build directory tree is also the ship tree.

PERFORMING A SANITY TEST (BASED ON A PSEUDO-INSTALLATION)

Perform the following steps to copy the CMVC GUI executables and related files in the actual install directories of your target platform. This testing is desired because it will be exactly as a normal end-user; that is, no special setup is needed.

1. Check the dates of the CMVC code recently built
2. Copy the CMVC code recently built into the install directories
3. Test the CMVC code recently built from the target host
4. Erase the CMVC code from the install directories (optional)

Check the dates of the CMVC code recently built

As user "build", issue the following command to check the dates of the existing code in the install directories (if any) and the dates of the currently built code in the build tree.

./check.gui.<mode> <from_directory>

Where:
from_directory: is the full path of the top portion
of the directory where the GUI output
resides, such as:

/home/build/cmvc23 /solaris/gui/x11r5/stal
mode: intg, stal

For example:
./check.gui.stal
/home/build/cmvc23 /solaris/gui/x11r5/stal
./check.gui.intg /home/build/cmvc23 /aix3/gui/x11r4/intg

Copy the CMVC code recently built into the install directories

Note: This step can only be done by 'root'.

1. Login as build and change the directory to 'bin'.
2. As user 'root', issue the following command to copy the code that was recently built from the build tree into the install directories.

   ./copy.gui.<mode> <from_directory>

   Where:
   from_directory: is the full path of the top portion
   of the directory where the GUI output
   resides, such as:

   /home/build/cmvc23 /solaris/gui/x11r5/stal
   mode: intg, stal

   For example:
   ./copy.gui.stal
   /home/build/cmvc23 /solaris/gui/x11r5/stal
   ./copy.gui.intg /home/build/cmvc23 /aix3/gui/x11r4/intg
3. Exit from root to become build again.

Test the CMVC code recently built from the target host

1. Make sure that all the CMVC environment variables are setup as a normal user:

   CMVC_HOME=/usr/lpp/cmvc
   PATH=$PATH:...:$CMVC_HOME/bin:...
   NLSPATH= # blank

   The NLSPATH could be empty, but if needed to be set, then here are the following values for the different environments:

   AIX: /usr/lib/nls/msg/En_US/%N
   SunOS: /usr/lib/nls/msg/En_US/%N
   Solaris: /usr/lib/locale/C/LC_MESSAGES/%N
   HP: /usr/lib/nls/C/%N
2. Copy the CMVC client application resource file, Cmvc, from the /usr/lib/X11/app-defaults directory to your home directory.
3. Execute 'cmvc' from an x-terminal and perform the testing.

Erase the CMVC code from the install directories (optional)

This step is very important when doing an official verification test, in order to avoid any possible contamination with previous installs.

Note: This step can only be done by 'root'.

As user 'root', issue the following command to erase the CMVC related files from the install directories.

./erase.gui.<mode>

Where:
mode: intg, stal

PACKAGING THE CMVC SERVER, LINE COMMANDS AND GUI

This chapter describes how to package the CMVC server, line commands and GUI for UNIX.

OVERVIEW OF PACKAGING FOR THE SERVER

The CMVC server for UNIX that the system administrator installs (from CD-ROM or from tape) is the collection of files shown below. These files are placed together in a single installable image, one per platform and per database (DB2, Oracle 7, Informix, and Sybase 4.9). For example, one installable image for the DB2 server for AIX, one installable image for Oracle 7 for HP-UX.

CMVC UNIX files, which are built and shipped (ship/server) as part of the server (using MAKE.cmvc with either '-b server' or '-b all') process described in this document:

directory: /usr/lpp/cmvc/bin

```
age
binToLong
chfield
cmvcarchive
cmvcd
cmvchost
cmvcrestore
dspmsg
monitor
notifyd
resetAge
sclean
vcPath
 PVCS.binary        # only for AIX
 PVCS.text          # only for AIX
```

directory: /usr/lpp/cmvc/bind # only for database DB2

all files

directory: /usr/lpp/cmvc/db2Convert # only for database DB2

all files

directory: /usr/lpp/cmvc/install

all files

directory: /usr/lpp/cmvc/samples

server.samples
<all the samples listed in server.samples>

directory: appropriate directory for Message Catalog (see bin/copy.server)

cmvc.cat

OVERVIEW OF PACKAGING FOR THE CLIENT (LINE COMMANDS AND GUI)

The CMVC client for UNIX that the system administrator installs (from CD-ROM or from tape) and that the end-user uses is the collection of files shown in the next subsections. These files are placed together in a single installable image, one per platform and per mode (standalone, integrated with SDE). For example, one installable image for the Standalone client for AIX, one installable image for the Integrated client for HP-UX.

Standalone files

Motif GUI files, which are built by the process described in this document:

– directory: /usr/lpp/cmvc/bin

README.gui
cmvc

– directory: /usr/lpp/cmvc/samples

Cmvc
system.cmvcrc
migResources

– directory: appropriate directory for Message Catalog (see bin/copy.gui.*)

cmvcui.cat
cmvchelp.cat

– directory: /usr/lib/X11

system.cmvcrc
XKeysymDB.cmvc # only for SunOS and Solaris
motifbind.cmvc # only for SunOS and Solaris

– directory: /usr/lib/X11/app-defaults

Cmvc

– directory: /usr/include/X11/bitmaps

cmvc.icon
cmvc.error
cmvc.info
cmvc.question
cmvc.tree

CMVC UNIX line commands, which are built and shipped (ship/client) as part of the client (using MAKE.cmvc with either '-b client' or '-b all').

- directory: /usr/lpp/cmvc/bin

  Access
  Approval
  Approver
  Component
  Coreq
  Defect
  Environment
  Feature
  File
  Fix
  Host
  Level
  LevelMember
  Migrate
  Notify
  Release
  Report
  Size
  Test
  Track
  User
  Verify

- directory: /usr/lpp/cmvc/samples

  client.samples
  <all the samples listed in client.samples>

- directory: appropriate directory for Message Catalog (see bin/copy.client)

  cmvc.cat

CMVC UNIX utilities, which are built (in src/adm) and shipped (in ship/client) as part of the server (using MAKE.cmvc with either '-b server' or '-b all').

- directory: /usr/lpp/cmvc/bin

  Fileimport
  Filemap
  Filemigrate
  dspmsg
  xecit

save.

Workbench Integrated files

Motif GUI files, which are built by the process described in this document:

- directory: /usr/lpp/cmvc/bin

  softcmvc

- directory: /usr/lpp/cmvc/samples

  Cmvc

- directory: appropriate directory for Message Catalog (see bin/copy.gui.*)

  softcmvc.cat

- directory: /usr/lib/X11 ????

  system.cmvcrc
  XKeysymDB.cmvc # only for SunOS and Solaris
  motifbind.cmvc # only for SunOS and Solaris

- directory: /usr/lib/X11/app-defaults

  Cmvc

- directory: /usr/$softbench/config/products

  softcmvcprod

- directory: /usr/$softbench/config/softinitsrc/class-defaults

  softcmvcinit

- directory: /usr/$softbench/help/app-defaults

  Softcmvc.hlp

- directory: /usr/$softbench/menus/help/app-defaults

  Softcmvc.hlp

- directory: /usr/$softbench/help/C

  Softcmvc.cat

- directory: /usr/$softbench/menus/help/C

  Softcmvc.cat

- directory: /usr/$softbench/menus/Softdm/CM/Softcmvc

  C
  MDF.m

Where $softbench is for AIX 'lpp/workbench' and 'softbench' for the other platforms.

CMVC UNIX line commands, which are built/shipped as part of the client process.

none

CMVC UNIX utilities, which are built/shipped as part of the server process.

none

PREPARING AIX SMIT INSTALLP IMAGES

To prepare the AIX smit installp images do:

1. Login as build.
2. Go where the installation tools are:

   cd cmvc23 /installp
3. For AIX 3.2 run:

   bld.installp
4. For AIX 4 run:

   bld4.installp

   An individual preparation for the smit installp images is performed as follows (for CMVC release 2.3, this is already in BldShip)

   For AIX 3.2, do:

   ```
   BldShip   db2        21
                              ===>  level
   ```

   For AIX 4, do:

   ```
   BldShip4   db2        21
                               ===>  level
   ```
5. You can disregard the warning message:

   End of inslist file: No such file or directory

APPENDIX A. INTRODUCTION TO THE CHECKLISTS

This document consists of the following:

A quick status checklist that in one page of text contains all the information about the latest builds for UNIX only. This checklist can be maintained on-line.

1. Use a forward slash to indicate when an activity has begun.

   This will allow you to see quickly which tasks need your attention.

2. Use a plus sign to indicate when an activity has ended.

A set of generic checklists that can be used to keep track of all the build activities that are necessary to produce CMVC 2.3.0 Client/Server and GUI for UNIX (6 copies needed, one for AIX 3.2, AIX 4.1, HP-UX 9, HP-UX 10, SunOS and Solaris).

1. Fill in the platform field in the title of the table. (Recommendation: use red ink)
2. Fill in the host name in the second row in the header of the table. (Recommendation: use red ink)
3. Use a forward slash to indicate when an activity has begun. (Recommendation: use pencil)

   This will allow you to see quickly which tasks need your attention.
4. Use a backward slash on top of the forward slash to form an X to indicate when an activity has ended. (Recommendation: use pencil)

The explanation of all the steps that are needed to build, package and make tapes can be found in their appropriate files in the directory /home/build/process.

MISCELLANEOUS NOTES

When extracting source code for AIX 3, AIX 4, HP-UX 10, SunOS or Solaris, use the following information:

hostname: rtplib 2
userid: 21
group id: 222

When extracting source code for HP-UX 9 use the following information:

hostname: oem-hp 4
userid: 212
group id: 2

The $HOMEBUILD directory depends on the host, such as in oem-ibm: /u/build

The target directory depends on the combination of platform and database, such as in Sybase 4.9 for AIX: $HOMEBUILD/cmvc230/aix3/cs/sybase49

For Solaris, do copy.client for oem-sn05.

For SunOS, do copy.client for oem-sn04.

For AIX 3, do copy.client for carcps08.

For HP-UX 9, do:

1. FTP /usr/lpp/cmvc/bin from oem-hp03
2. FTP /usr/lib/nls/C/cmmvc.cat from oem-hp03

The column labeled NoNetLS is for tracking the build for the Non-NetLS version of the line commands.

When doing remote login from AIX into other platforms, take the following into account:

1. Solaris

   Terminal type: xtermm
   This is code already in the .profile

2. SunOS

   Terminal type: xterms
   This is code already in the .profile

3. HP-UX

   The following information is not in the profile and needs to be entered when asked:

   Terminal type: ansi

   When possible, you can telnet to an HP-UX box, set the DISPLAY variable back, then issue:

   hpterm &

APPENDIX B. CMVC FOR UNIX, BUILD AND PACKAGE STATUS CHECKLIST

Use as reference the following real quick checklist status for the building of level 23 for UNIX.

Subject: (U) Status of CMVC CS/GUI builds as of Level 23, as of 8/3
Date last updated: October 4, 1996

Convention: "-" next activity to be performed
"/" activity started
"+" activity completed

Client / Server builds for CMVC

				extract	q m a k e	s h i p	obj s r v	tar c l t	t s s t	t c l t	ftp e p e	c a x t o m
Solaris	oem-sn	1	sybase49 all									
Solaris	oem-sn	1	oracle7.1 server									
Solaris	oem-sn	1	oracle7.1client (noNetLS)									
SunOS	oem-sn	3	sybase49 all	f2-								
SunOS	oem-sn	3	informix server									
SunOS	oem-sn	3	oracle7. server									
SunOS	oem-sn	3	oracle7. client (noNetls)									
AIX 3	oem-ibm		db2 all	f22	d23	+	+		+	+		
AIX 3	oem-ibm		sybase49 server									
AIX 3	oem-ibm		informix server	f22	d23	+	+		+			
AIX 3	oem-ibm		informix client (noNetLS)			+	+		+			
AIX 3	carcps	8	oracle7. server	f23		+	+		+			
AIX 4	cordoba		informix all	f22	d23	+	+		+	+		
AIX 4	oem-ppc1		oracle7.1 server	f23		+	+		+			
AIX 4	oem-ppc1		db2v2 server	f22	d23	+	+		+			
AIX 4	cordoba		db2v2 client (noNetls)			+			+			
HP 9	oem-hp	2	informix all									
HP 9	oem-hp	2	oracle7. server									
HP 9	oem-hp	2	oracle7. client(noNetLS)									
HP 1	oem-hp	3	informix all									
HP 1	oem-hp	3	oracle7.1 server									
HP 1	oem-hp	3	oracle7.1client (noNetLS)									

ps/products/cmvc/doc/readme +
ps/products/cmvc/info +

extract q s obj tar t t ftp c
m h s c s c e a i e d
a i r l r l s p n x r
k p v t v t t e t t o
e m

Motif GUI builds for CMVC
extract qmake

Solaris oem-sn 1 x11r5/stal f2 d21 +
Solaris oem-sn 1 x11r5/intg f2 d21 +

SunOS oem-sn 3 x11r4/stal f2 d21 +
SunOS oem-sn 3 x11r4/intg f2 d21 +

AIX 3 oem-ibm x11r4/stal f2 d21 +
AIX 3 oem-ibm x11r4/intg f2 d21 +
AIX 3 oem-ibm x11r5/stal f2 d21 +
AIX 3 oem-ibm x11r5/intg f2 d21 +

AIX 4 oem-ppc1 x11r5/stal f2 d21 +

HP 9 oem-hp 4 x11r4/stal X11r4 not installed
HP 9 oem-hp 4 x11r4/intg X11r4 not installed
HP 9 oem-hp 4 x11r5/stal f2 d21 +
HP 9 oem-hp 4 x11r5/intg f2 d21 +

HP 1 oem-hp 3 x11r5/stal f21 +
HP 1 oem-hp 3 x11r5/intg Encapsulator not installed

APPENDIX C. CMVC CLIENT/SERVER UNIX, BUILD AND PACKAGE CHECKLIST

Figure 12 (Page 1 of 4). CMVC Client/Server UNIX, Build and Package Checklist, Level: ____

Activity	**NoNetLS** (_____) **client**	**oracle7** (_____) **all/server**	**informix** (_____) **all/server**	**db2** (_____) **all/server**	**sybase49** (_____) **all/server**
logon as build					
change to target directory					
delete target build directory					
extract full latest committed level					
extract delta latest uncommitted level					
chmod 755 qmake					
update qmake and specify build options					
qmake clean					
qmake					
qmake ship					
-- prepare installp (obj) images for AIX --					
cd $HOMEBUILD/cmvc230/installp					
prepare installp images: bld.all.images					

Figure 12 (Page 2 of 4). CMVC Client/Server UNIX, Build and Package Checklist, Level: ____

Activity	**NoNetLS** (_____) **client**	**oracle7** (_____) **all/server**	**informix** (_____) **all/server**	**db2** (_____) **all/server**	**sybase49** (_____) **all/server**
-- prepare tar images for others --					
su - root (ksh, set -o vi)					
cd $HOMEBUILD/cmvc230/installp					
make -f Makefile.PLATFORM.DATABASE					
make -f Makefile.PLATFORM.client (use SA_SHIP if building only for Stand Alone)					
perform quick sanity test					
-- if OK, prepare tapes and copy images --					
perform build of the GUI					
prepare backup tapes with install images					
place install images in internal ftp site					
update home page for CMVC					
place install images in external ftp site					
place install images for AIX ShowCase CD-ROM					
-- end now if not doing formal FVT --					

Figure 12 (Page 3 of 4). CMVC Client/Server UNIX, Build and Package Checklist, Level: ____

Activity	**NoNetLS** (_____) **client**	**oracle7** (_____) **all/server**	**informix** (_____) **all/server**	**db2** (_____) **all/server**	**sybase49** (_____) **all/server**
--continue here if doing formal FVT --					
cd $HOMEBUILD/bin					
./copy.all directory db (if both server and client)					
./copy.server (if only server)					
./copy.client (if only client)					
exit (from root to build)					
touch.fvt					
update qmake for FVT					
qmake					
su - root					
cd $HOMEBUILD/bin					
./copy.beta.fvt directory fvt					
exit (from root to build)					
cd /usr/lpp/cmvc/bin					
what cmvcd | grep cmvc (also in FVT and BETA)					

Figure 12 (Page 4 of 4). CMVC Client/Server UNIX, Build and Package Checklist, Level: ____

Activity	**NoNetLS**	**oracle7**	**informix**	**db2**	**sybase49**
	(_____)	**(_____)**	**(_____)**	**(_____)**	**(_____)**
	client	**all/server**	**all/server**	**all/server**	**all/server**
copy client to _____					
send notes and post append in V2					

APPENDIX D. CMVC GUI FOR UNIX, BUILD AND PACKAGE CHECKLIST

Figure 13 (Page 1 of 2). CMVC GUI for UNIX, Build and Package Checklist, Level: _____

Activity	**x11r4 intg** (_____)	**x11r4 stal** (_____)	**x11r5 intg** (_____)	**x11r5 stal** (_____)
logon as build				
change to target directory				
delete target build directory				
extract full latest committed level				
extract delta latest uncommitted level				
chmod 755 qmake				
update qmake for BETA or normal, x11r4 or r5				
qmake clean				
qmake				
qmake ship				
su - root				
cd $HOMEBUILD/cmvc230/installp				
make -f Makefile.PLATFORM.gui				
cd $HOMEBUILD/bin				

Figure 13 (Page 2 of 2). CMVC GUI for UNIX, Build and Package Checklist, Level: _____

Activity	**x11r4 intg** (_____)	**x11r4 stal** (_____)	**x11r5 intg** (_____)	**x11r5 stal** (_____)
./copy.gui.intg directory				
cd /usr/lpp/cmvc/bin				
what cmvc | grep cmvc what softcmvc | grep cmvc				
exit (from root and back to build)				
send notes and post append in V2				
test code				
additional comments				

APPENDIX E. HOW TO PUT THE CMVC CODE INTO TAPES

Notes:

1. Do not use IBM's 4mm tapes in the HP tape drives.
2. To quickly get 4mm or 8mm tapes, contact Stay-On-Line, Durham, NC Phone: 919-361-1222

HOW TO GET TAPES VIA VM

8mm

PCWARES -> Shop -> Search
492587 8mm Data Tape D8-112 by 3M (4 858) 1 SPEC
(obsolete number 6315 7)

4mm

CPOPS -> 7: Catalog Shopping -> RALEIGH Staples Raleigh Staples National
523274 CARTRIDGE DL6 DATA 4MM 88 38 EA 6.82

use: Divison RC
Major 75
Minor 34
Project 9212
Ship to Code: XP9, IBM 3 39 Cornwallis

SEQUENCE TO MAKE A TAPE FOR CMVC

1. Login as user build from oem-sn03 (SunOS), oem-sn01 (Solaris), oem-hp04 (HP) and from carcps08 (AIX).
2. Change to the top directory for the install files:

   cd cmvc23 /installp
3. Change to the platform directory, such as aix, hpux, sun, solaris

   For making a tape with tar files, see 4.

   For making a tape with installp files, see 5 on page 70.
4. For making a tape with tar files, do:

   a. Enter:

      mktape

   b. Specify the non-rewinding device, and let it go.

c. You can open a second session in the meantime and compare the sizes with the contents on the tape.map file in the current directory.

d. After the tape is written, then verify that it can be read, by running the following command that will rewind the tape, read the first 2 files and then rewind the tape again:

```
get.first.two      <non-rewinding-device>
```

Note: The first file is the cmvcinstall shell script and the second file is the first CMVC file (standalone client).

Go to step 6

5. For making a tape with smit installp images, do:

a. Enter:

```
mkstack
```

b. By default the non-rewinding device will be used (/dev/rmt0).

c. You can open a second session in the meantime and compare the sizes with the contents on the stack.map file in the current directory.

d. After the tape is written, then verify that it can be read, by running the following command that will rewind the tape, read the first 4 files and then rewind the tape again:

```
get.file      <non-rewinding-device>     4
```

Note: In a stack tape like this, the first 2 files are dummy, the 3rd file is the table of contents and the 4th is the first real file from CMVC.

Continue with step 6.

6. Take out the tape, label it and move the tab to the write protection setting, which is different for each type of tape:

8mm: the raised red tab should be moved towards the center of the tape and the whole red tab should be visible.

4mm: the raised grey tab should be moved towards the center of the tape and the whole grey tab SHOULD NOT BE visible.

APPENDIX F. BIBLIOGRAPHY

For more information on how to use CMVC, you can consult the following manuals and publications:

SC 9-1596- 1 IBM CMVC Client Installation and Configuration
SC 9-1597- 1 IBM CMVC User's Reference
SC 9-1631- 2 IBM CMVC Server Administration and Installation
SC 9-1633- IBM CMVC Concepts
SC 9-1635- 1 IBM CMVC Commands Reference

The following Redbooks offer practical advice on CMVC:

GG24-4178- Did you say CMVC?
GG24-4345 CMVC: Customer's Perspective

The following technical reports describe in detail useful hints on using CMVC:

29.213 Diagnosing and solving Release -extract problems with CMVC
29.2169 How to use CMVC with National Language Support (NLS) and Double-Byte Character Sets (DBCS)
29.218 How to do routine tasks in the OEM Platforms for CMVC
29.2183 Using CMVC for Products with Multiple National Language Versions (NLVs)
29.2232 How to do migration tasks with CMVC
29.2244 How to build and package the CMVC client for Windows 3.1
29.2245 How to build and package the CMVC client for OS/2
29.2253 Comparison between CMVC 2.3 and TeamConnection 2
29.2254 Migrating from CMVC 2.3 to TeamConnection 2

HOW TO GET ELECTRONIC COPIES OF MANUALS AND TRS

Many of the manuals and technical reports mentioned in this document can be downloaded as follows:

From the IBM intranet (only for IBM employees).

From the Internet (open to everyone).

IBM Intranet

Web Home Page

You can access the CMVC Service/Development Home Page at:

http://keithp.raleigh.ibm.com/~cmvcsupt

From the index at the top of the page, select:

Technical Reports and related tools

Copies of ALL the archived versions for the forums CMVC and CMVC6000

Documentation in PostScript files

Documentation in ASCII text files

FTP

You can download the code from our internal FTP site, by doing:

1. ftp keithp.raleigh.ibm.com
2. login as 'anonymous' and for password give your email address.
3. cd pub/cmvc/doc
4. binary
5. get fileName
6. quit

Internet

Web Home Page

Not available.

FTP

You can download the code from our external FTP site, by doing:

1. ftp ftp.software.ibm.com
2. login as 'anonymous' and for password give your email address.
3. cd ps/products/cmvc/doc
4. binary
5. get fileName
6. quit

APPENDIX G. COPYRIGHTS, TRADEMARKS AND SERVICE MARKS

The following terms used in this technical report, are trademarks or service marks of the indicated companies:

TRADEMARK, REGISTERED TRADEMARK OR SERVICE MARK	COMPANY
AIX, OS/2, IBM, DB2, CMVC, TeamConnection	IBM Corporation
UNIX	X/Open Co., Ltd.
NetLS	Gradient Technologies, Inc.
Motif, DFS	Open Software Foundation, Inc.
INFORMIX	Informix Inc.
ORACLE	Oracle Corp.
SYBASE	Sybase Inc.
NFS, SunOS, Solaris Sparc, SPARCstation	Sun Microsystems Inc.
HP-UX, SoftBench	Hewlett-Packard Company
Microsoft, Windows,	Microsoft Corporation
PVCS	InterSolv, Inc.
AFS	Transarc Corp

END OF DOCUMENT