How to resolve SCCS related problems in CMVC and how to reclaim space from unreferenced SCCS files

Document Number TR 29.3336

Angel Rivera

CMVC Customer Support
IBM Software Solutions
Research Triangle Park, North Carolina, USA

DISCLAIMER:

This technical report is not an official publication from the CMVC group. The author is solely responsible for its contents.

ABSTRACT

This technical report provides workarounds and procedures to fix common problems when CMVC and SCCS are out of synch. Also, some procedures and tools are provided to reclaim the space used by unreferenced SCCS files.

ITIRC KEYWORDS

- CMVC
- Versioning system

ABOUT THE AUTHOR

ANGEL RIVERA

Mr. Rivera is an Advisory Software Engineer and team lead for the CMVC Direct Customer Support team. 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.

CONTENTS

FIGURES

INTRODUCTION

This technical report provides workarounds and procedures to fix common problems when CMVC and SCCS are out of synch. Also, some procedures and tools are provided to reclaim the space used by unreferenced SCCS files.

The chapters in this TR are organized as follows:

- Chapter "Relationship between sourceId and SCCS file in the vc tree" on page 5 describes the relationship between certain attributes of the database related to CMVC files and CMVC versions and the related SCCS file in the $HOME/vc tree.
- Chapter "How to reclaim disk space from unreferenced SCCS files" on page 11 describes a strategy for reclaiming the space taken by old SCCS files that are no longer referenced meaningfully within the CMVC family.
- Chapter "How to fix errors with the SCCS s.XX and p.XX files" on page 25 describes how to fix errors with the SCCS s.XX and p.XX files.
- Chapter "How to fix some common errors" on page 33 describes how to fix the situations underlying some common error messages.

This technical report describes additional material that was originally mentioned in the following technical reports:

```
ftp://ftp.software.ibm.com/ps/products/cmvc/doc/tr/trcmfvc2.pdf
How CMVC exploits SCCS
TR 29.3317

ftp://ftp.software.ibm.com/ps/products/cmvc/doc/tr/tr292297.txt
CMVC frequently asked questions:
version control and database synchronization issues
TR 29.2297
```

The DBMSs that were used in the examples in this document are DB2 Universal Database (UDB) 5.2 and Oracle 7.3.4. However, the SQL clauses are the same for other DBMSs.

WARNING! READ THIS SECTION BEFORE ATTEMPTING TO FIX AN SCCS PROBLEM

Since directly modifying ANY data within the CMVC database or the SCCS files can be risky, here is a list of warnings and caveats:

1. If you rename, delete or move files in the vc directory structure, you will be doing this at your own risk. Problems with CMVC resulting from such changes do not constitute

"defects in the product". IBM has a services organization that can provide fee-based services to help you in such cases.

2. Before attempting to follow any of the recovery procedures described in this technical report, it is recommended that you stop the CMVC daemons and do a complete backup of both the database and the file system. In case there are problems, it is important to have a means of recovery. CMVC provides a sample backup utility, `backupCMVC`, in the samples directory.
3. Save copies of all files before making changes.
4. Try to avoid editing the SCCS archive files (s.XX) directly. They are binary.

   However, if you need to modify them, make sure that you make a backup copy, just in case.
5. Do not modify any data in the database.
6. Verify that the syslog is running.

   If it is not running, see the technical report "CMVC frequently asked questions: hints and tips on using databases and OEM platforms", section "Where are the error messages from the CMVC daemon and databases?".

   The latest version of this document can be found at:

   `ftp://ftp.software.ibm.com/ps/products/cmvc/doc/tr/tr292298.txt`

   You can also see Chapter 14 "Ongoing Maintenance" in *CMVC Server Administration and Installation, Version 2 Release 3* to set up syslog.

   Then, restart the daemons and recreate the problem to see if there are associated error messages in your syslog file.
7. If you are uncertain, please contact IBM Support before using these procedures by means of a Problem Management Report (PMR).

DISCLAIMERS

To avoid misunderstandings with the purpose of this technical report and to better understand its scope, the following disclaimers are in order:

- This technical report is not an official publication from the CMVC group. The author is solely responsible for its contents.
- This technical report was prepared when working with CMVC 2.3.1.4 in AIX 4.2.1 using DB2 Universal Database (UDB) 5.2 and Oracle 7.3.4. Therefore, if you have a different version of the mentioned software, then you may expect some differences in the information or in the procedures described in this technical report.

- This technical report covers information that the author has gathered thru the years while working with the CMVC technical support team.
- It is the intention of this technical report to provide recommendations and guidelines that can be helpful to CMVC administrator. In some cases, the procedures will not be exhaustive, and will just show the overall sequence that has worked before, which might be different in your case.
- Real values that were used in our setup will be used in this technical report. Thus, you will need to customize the commands that you issue to reflect the values that are meaningful to your setup.
- It is assumed that the reader has knowledge of CMVC, the appropriate database management system (DBMS) and the appropriate operating system.

  This technical report is not a substitute to the information provided by CMVC and the appropriate DBMS and operating system. Please refer to the appropriate documentation provided with the corresponding software.

HOW TO GET THE MOST UP TO DATE VERSION OF THIS TECHNICAL REPORT.

The most up to date version of this technical report can be obtained from the IBM CMVC ftp site at URL:

`ftp://ftp.software.ibm.com/ps/products/cmvc/doc/tr/trcmfvc3.txt`

For the list of available technical reports, see the file:
ftp://ftp.software.ibm.com/ps/products/cmvc/doc/tr/README.index.txt

ACKNOWLEDGEMENTS

Many of the questions and answers that are compiled in this technical report were obtained from the CMVC forum in the IBMPC conferencing disk and from the CMVC6000 forum in the IBMUNIX conferencing disk. We want to thank the main participants in these electronic forums for their support!

We want to thank in particular the following co-workers:

- Lee Perlov, Websphere/VisualAge TeamConnection Services, IBM RTP, North Carolina, USA.
- Edna Wong Kyu, OEM Lab in IBM RTP, North Carolina, USA.
- Keith Purcell, OEM Lab in IBM RTP, North Carolina, USA.

MISCELLANEOUS TOPICS

What are the necessary patches for SCCS for the Year 2000

The following document contains more information about the necessary patches for the SCCS utility for the Year 2000 and beyond:

`ftp://ftp.software.ibm.com/ps/products/cmvc/README.year2000.txt`

SCCS keeps using only 2 digits to represent the year

The following document contains more information about the way SCCS uses the year inside the archive files:

`ftp://ftp.software.ibm.com/ps/products/cmvc/README.year2000.details.txt`

At the time this technical report was prepared, the relevant paragraphs from the mentioned document are shown below.

- SCCS issues (outside the scope of CMVC)

  CMVC has not control over the internal handling of dates by SCCS.

  The SCCS keywords that deal with dates (such as D, surrounded by %), represent the year with only 2 digits. When an archive file is prepared by SCCS and the date keywords are expanded by SCCS, the expansion shows ONLY 2 digits for the year. CMVC just passes this expanded file as-is to the end user; that is, CMVC does NOT attempt to correct the SCCS expanded year with 2 digits.

  The important point is that SCCS must have the necessary fixes/patches to allow it to handle the year 2000 correctly. The 2 digit years were corrected to use the same "window" that the operating system date uses:

  – Where 69 thru 99 is interpreted to be 1969 thru 1999.

  – Where 00 thru 68 is interpreted to be 2000 thru 2068.

  For more information on SCCS and the Year 2000, you can visit the following URL:

  `http://www.free-lunch.demon.co.uk/CSSC/manual/cssc_48.html#SEC55`

RELATIONSHIP BETWEEN SOURCEID AND SCCS FILE IN THE VC TREE

This chapter describes the relationship between certain attributes of the database related to CMVC files and CMVC versions and the related SCCS file in the $HOME/vc tree.

For more details on the tables and views used in the CMVC database, see the "CMVC User's Reference", chapters 6 (Views) and 7 (Tables).

SEQUENCE TABLE: LASTSERIAL ATTRIBUTE FOR 'SOURCE'

The lastSerial attribute for the record named "source" in the Sequence table indicates what is the latest id number given to the sourceId attribute for the Files table.

Only when a new file is created in CMVC, this lastSerial value is read, then it is increased by 1 and the resulting value will be the actual sourceId used for the new file.

The rest of the CMVC File actions will not cause the modification of this attribute.

To find out the value for the lastSerial attribute for the record "source" in the Sequence table, you can issue the following command:

```
db2 "select * from Sequence where name='source'"
```

FILES TABLE (FILEVIEW VIEW): SOURCEID ATTRIBUTE

Every file stored in CMVC has associated in the File table or in the FileView view of the database, an attribute called "sourceId". The value for this attribute is an integer (up to 6 digits) that indicate the precise location of the SCCS file that contains the change deltas; this SCCS file is located inside a hierarchical structure under the $HOME/vc directory, which is called the "vc tree".

To find out all the values for the sourceId attributes in the Files table, you can issue the following command:

```
db2 "select sourceId,nuVersionId,id,basename from Files"
```

The structure for the SCCS files is different between the 2 main different file types in CMVC, because each type is handled differently (text files use forward deltas and binary files use backward deltas):

- Text files are described in “Relationship between sourceId and vc tree for a text file” on page 7.
- Binary files are described in “Relationship between sourceId and vc tree for a binary file” on page 8.

VERSIONS TABLE: SOURCEID ATTRIBUTE

Every CMVC File action that modifies the version number (SID) of a file, such as the creation of the file (version 1.1) and the checkin of a file (such as 1.2), will create a new record in the Versions table that is associated with the sourceId for the file from the Files table.

To find out all the values for the sourceId attributes in the Version table, you can issue the following command:

```
db2 "select sourceId,id,previousId,SID from Versions"
```

ALL THE INFORMATION THAT SCCS USES IS IN THE VC TREE

CMVC maintains the information about the files and versions in several tables and views, and the detailed discussion of the relationship is outside the scope of this technical report, because the user should use CMVC commands to handle the files, and should not manually change the records in the database.

SCCS maintains the information about the files and their changes entirely in the vc tree directory by means of the s.XX and p.XX files. That is, there is no special database that SCCS uses under the covers. To solve some CMVC synchronization problems, it might be necessary to manually manipulate the SCCS files; it is strongly recommended to make a backup of these files (by using a name that SCCS will not use, to avoid confusion) before attempting to manipulate them.

USING THE VCPATH TOOL TO FIND OUT THE SCCS FILE

The CMVC server tool "vcPath" can be used to find out the name of the SCCS file/directory and where is located. For example:

```
vcPath text.txt rel1
```

The output should look like this:

```
The file, text.txt, associated with release,
rel1, maps to the file, s.45, in the
directory, $HOME/vc/0/1/2/3, of the CMVC family's account
```

This means that the complete path for the file is:

```
$HOME/vc/0/1/2/3/s.45
```

RELATIONSHIP BETWEEN SOURCEID AND VC TREE FOR A TEXT FILE

Finding the SCCS file for a text file given a sourceId value

Let's assume that a text file created in CMVC is named "file1" and belongs to the release "rel1". The following SQL statement can be used to find out the sourceId for this file:

```
db2 "select sourceId from FileView where baseName='file1' and \
     releaseName='rel1' "
```

In this example, let's assume that the value is "12345". The first step is to prefix the number with enough zeros to make the number to be a 6 digit number; in this case "012345".

Given the sourceId of "012345" the 4 most significant digits ("0123") will indicate the hierarchy of subdirectories inside the $HOME/vc tree:

```
$HOME/vc/0/1/2/3/
```

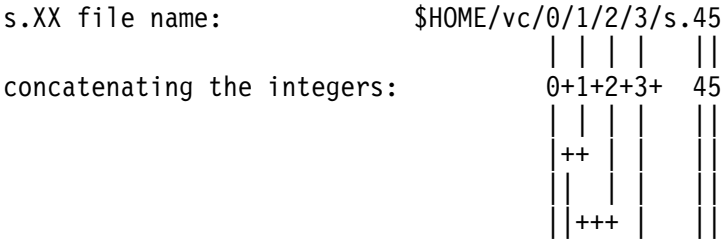
Then, the 2 least significant digits ("45") indicate the SCCS file that has the actual deltas for the CMVC file. The name for this SCCS file for a text CMVC file has the format "s.XX" where the XX is a placeholder for the last 2 significant digits from the sourceId attribute, in this case, the SCCS file name is "s.45".

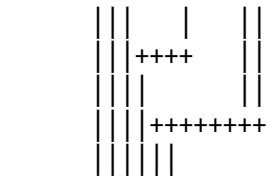
Therefore, the full path name for the SCCS file that has the delta changes for the CMVC file that has the value "012345" for the attribute "sourceId" in FileView, is:

```
$HOME/vc/0/1/2/3/s.45
```

Finding the sourceId value given the full path name of a SCCS file

If you have the full pathName of the s.XX file in the CMVC vc tree, such as $HOME/vc/0/1/2/3/s.45, then you obtain the sourceID by concatenating all the integers that appear in the vc subdirectories and the file name, from left to right:





```
CMVC sourceID in FileView:          012345
```

Once you have the sourceId, you can issue the following command to find out the file name and its release:

```
db2 "select releaseName,baseName from FileView where sourceId=012345"
```

Also, you could not specify the leading zeros:

```
db2 "select releaseName,baseName from FileView where sourceId=12345"
```

In both cases, the output should look like this:

```
RELEASENAME           BASENAME
--------------------- -----------------------------------
rel1                  file1
```

RELATIONSHIP BETWEEN SOURCEID AND VC TREE FOR A BINARY FILE

Finding the SCCS directory for a binary file given a sourceId value

Let's assume that a binary file created in CMVC is named "file2" and belongs to the release "rel1". The following SQL statement can be used to find out the sourceId for this file:

```
db2 "select sourceId from FileView where baseName='file2' and \
     releaseName='rel1' "
```

In this example, let's assume that the value is "6789". The first step is to prefix the number with enough zeros to make the number to be a 6 digit number; in this case "006789".

Given the sourceId of "006789" the 4 most significant digits ("0067") will indicate the hierarchy of subdirectories inside the $HOME/vc tree:

```
$HOME/vc/0/0/6/7/
```

Then, the 2 least significant digits ("89") indicate the SCCS directory that has the actual deltas for the CMVC file. The name for this SCCS directory for a binary CMVC file has the format "b.XX" where the XX is a placeholder for the last 2 significant digits from the sourceId attribute, in this case, the SCCS directory name is "b.89".

Therefore, the full path name for the SCCS directory that has the delta changes for the CMVC file that has the value "006789" for the attribute "sourceId" in FileView, is:

```
$HOME/vc/0/0/6/7/b.89/
```

Finding the sourceId value given the full path name of a SCCS directory

If you have the full pathName of the b.XX directory in the CMVC vc tree, such as $HOME/vc/0/0/6/7/b.89, then you obtain the sourceID by concatenating all the integers that appear in the vc subdirectories and the file name, from left to right:

```
b.XX directory name:        $HOME/vc/0/0/6/7/b.89
                                    | | | |   ||
concatenating the integers:         0+0+6+7+  89
                                    | | | |   ||
                                    |++ | |   ||
                                    ||  | |   ||
                                    ||+++ |   ||
                                    |||   |   ||
                                    |||++++   ||
                                    ||||      ||
                                    ||||++++++++
                                    ||||||
CMVC sourceID in FileView:          006789
```

Once you have the sourceId, you can issue the following command to find out the file name and its release:

```
db2 "select releaseName,baseName from FileView where sourceId=006789"
```

Also, you could not specify the leading zeros:

```
db2 "select releaseName,baseName from FileView where sourceId=6789"
```

In both cases, the output should look like this:

```
RELEASENAME           BASENAME
--------------------- ------------------------------------
rel1                  file2
```

HOW TO RECLAIM DISK SPACE FROM UNREFERENCED SCCS FILES

The original architecture of CMVC provides only facilities for growing the CMVC family (both the database and file system). That is, it was not envisioned that a CMVC family will be reduced in size. Most of the objects in CMVC are not deleted from the database. For example, when a CMVC user is deleted, actually the dropDate field is filled in and this represents a "deleted" user; the main reason is that by removing the value in the dropDate field, then the user could be easily recreated. The exceptions are those "list" objects such as host lists, access lists, etc, which do not have a dropDate field, and when they are deleted, their records in the database are actually purged.

Another exception is with the CMVC Files, which when "deleted" their dropDate field is filled in, but when "destroyed", then the physical record in the database is literally eliminated; this action recovers few bytes that the database used. The "destroy" action by itself does not allow to regain meaningful amount of file system space. However, when a file is deleted/destroyed, the corresponding SCCS file may (or may not) become unreferenced, and these unreferenced SCCS files could be deleted, and potentially, some meaningful amount of disk space might be recovered.

This chapter describes a strategy for reclaiming the space taken by old SCCS files that are no longer referenced within the CMVC family. The structure of this chapter is the following:

1. Section “Expectations for the procedure to reclaim disk space” on page 12 sets the customer expectations.
2. Section “Why some SCCS files become unreferenced by CMVC” on page 12 describes scenarios in which an SCCS archive file becomes unreferenced by CMVC.
3. Section “Using release.complete and release.delete scripts” on page 13 describes two scripts that can help with the deletion of old releases, in order to increase the amount of valid unreferenced SCCS archive files.
4. Section “Identification of the unreferenced SCCS archive files” on page 16 identifies in a file those unreferenced SCCS archive files that can be safely deleted.
5. Section “Deleting the unreferenced SCCS archive files” on page 21 describes how to actually eliminate those unreferenced SCCS archive files that were identified in section “Identification of the unreferenced SCCS archive files” on page 16.

EXPECTATIONS FOR THE PROCEDURE TO RECLAIM DISK SPACE

If your primary motivation for recovering file system space is that you do not have enough space for the family, then it is important to mention that the process in this document is a short-term solution, because in order to obtain a viable mid-term and long-term solution, the CMVC file system needs to have additional physical space.

It is important to set the customer expectations right: the amount of file system space that might be reclaimed will be very likely NOT as much as the customer is expecting.

The main reason is this: if file1 was created in release-1 and assuming that it takes 1 MB of space in the vc file system, and then the file is linked to release-2 and then the file is deleted and destroyed from release-1, the SCCS file (1 MB) is still used in release-2, and thus, the 1 MB file from SCCS cannot be deleted. In this case, the sourceId (which is used to determine the name of the SCCS file) is no longer used in release-1 but it is used in release-2.

WHY SOME SCCS FILES BECOME UNREFERENCED BY CMVC

This section describes two situations in which some CMVC activities cause that a SCCS file becomes unreferenced.

Deleting and destroying a file

The CMVC `File -destroy` command remove records from the CMVC database. If a committed file is deleted and then it is destroyed, then its sourceId is not referenced by any CMVC Release.

The delete action keeps the actual record in the database but the attribute dropDate is filled in.

The destroy action actually deletes the complete row for the object from the database, but the SCCS file is not touched. So, over time unreferenced SCCS archives may accumulate in a family.

The obsolete CMVC function `cmvcarchive` provides an option to delete SCCS archives and the database records for levels of a release, or releases. **Warning:** We discourage the use of the cmvcarchive utility.

Deleting a file that was not committed.

If the CMVC "File -undo" action is performed on a file that was never committed and eventually the creation of the file is 'undone', then, this causes the deletion of the record for that file from the database. In this case, the SCCS file is actually deleted.

When a new file is created, then the source filed in the Sequence table is increased by 1, and a new row is added to the Files table with the value for the sourceId attribute being the new value for the source field; for example, 13. This source id helps to define the s.XX name of the SCCS file. Also, a new record in the Versions table is added.

However, if this file is not committed and then File -undo is done, then eventually the file will be totally deleted and destroyed from CMVC.

There are 3 important actions for the undo action when a file is not committed:

- The File table will not have a record for this file. From the CMVC point of view, the file was never present in CMVC. That is, the File that was created, but never committed and then it was deleted permanently from CMVC.
- The source field in the Sequence table is NOT updated, that is, it is not decreased by 1. For example, if the value was 12 before the File create, and then 13 was used as the sourceId for the new file, then the next new file will have the sourceId of 14 and the old 13 is no longer in use and it is not reused.
- The corresponding file in SCCS, such as s.12 is deleted and it is not reused.

USING RELEASE.COMPLETE AND RELEASE.DELETE SCRIPTS

If you have "old" releases that are not active anymore, you can use the following scripts (in this particular order):

1. Login to the CMVC family user id.
2. Stop the CMVC family. It is important that NO new write-related activity is happening to the CMVC family while the cleanup procedure is in place.
3. Backup the CMVC family, which consists of 2 components:
   - The database.
   - The complete $HOME of the CMVC family user id. If there are symbolic links or mounted NFS file systems that are mapped this directory structure, then they also need to be backed up.
4. Script "release.complete" can be used to complete the release For more details see “Script: release.complete” on page 14.

5. Script "release.delete" can be used to delete the files associated with the release that was completed. For more details see “Script: release.delete” on page 15.

   This script does not destructive actions, such as File -destroy. The File -delete action will keep the record of a file in the database, but it will mark the file as deleted (dropDate attribute is filled in). In contrast, the File -destroy action (which you need to execute in a separate step) will physically remove the database record for a file that has been marked as deleted.

6. Once the files are "deleted" (that is marked as deleted, but their records are still in the database, you can create a new track and then use "File -destroy" to actually remove the old record from the database. Because the destroy action eliminates only the records in the database, you still need to perform the step to remove the unreferenced SCCS files.

Script: release.complete

The script release.complete can be downloaded from:

`ftp://ftp.software.ibm.com/ps/products/cmvc/fixes/release.complete`

The header of this Korn shell script contains the following information:

```
# NAME: release.complete -f family -r releaseName
#       Where: family      is the CMVC family name.
#              releaseName is the active CMVC release.
# NOTES:
#   * This script must be run by a SUPERUSER.
#
# PURPOSE:
#   To ensure that all the objects related to a release
#   such as tracks, approval records, fix records, test
#   records and levels are completed.
#   This script finds out the processes that are used
#   by the release, and then it will do:
#
#   * If the track subprocess is used, then it will
#     try to integrate the Tracks in fixed state.
#
#   * If the approval subprocess is used, then it will
#     try to delete any Approval records in ready state.
#
#   * If the fix subprocess is used, then it will
#     try to complete any Fix records in active or ready state.
#
#   * If the level subprocess is used, then it will
#     try to remove any level members of uncommitted levels,
#     remove uncommitted levels, and create a "cleanup" level
#     to process all the remaining tracks in integrate state,
#     and commit and complete this level; finally, it tries
#     to complete any levels in committed state.
#
#   * If the test subprocess is used, then it will
#     try to abstain any Test records in ready state.
#
# AUXILIARY FILES:
#  This script generates 2 files, one that has the details
#  of the processing, and another of the commands that failed.
```

Script: release.delete

The script release.complete can be downloaded from:

`ftp://ftp.software.ibm.com/ps/products/cmvc/fixes/release.delete`

This script does NOT perform the "File -destroy" action. Once the track that has the delete actions is committed, then you need to create a new track and perform the "File -destroy" on the files, then commit the track.

The header of this Korn shell script contains the following information:

```
# NAME: release.delete -f family -r releaseName -d defect [-w]
#       Where: family      is the CMVC family name.
#              releaseName is the active CMVC release.
#              defect      is the CMVC defect for the track in fix state.
#              w           is an option to identify what SCCS and
#                          mapfiles can be deleted.
# NOTES:
#   * This script must be run by a SUPERUSER.
#   * It is strongly recommended to backup the CMVC family before
#     attempting to run this command due to its destructive nature.
#
# PURPOSE:
#
#   This utility will delete the files in a given release.
#
#   It will also identify a list of the SCCS files that are not related
#   to any other release that can be removed. This script does not
#   actually remove them.
#   It is up to the user to manually remove them in order
#   to reclaim disk space.
#
#   The following conditions must be present in order to delete
#   a release:
#
#   * All pending work must be committed. To accomplish this goal,
#     it is recommended to issue the sample: release.complete
#
#   * All the files in the release must be unlocked.
#
#   The user will need to commit the track in which the files were
#   deleted. In that way, the files can be destroyed later on and the
#   actual release object can be deleted.
```

IDENTIFICATION OF THE UNREFERENCED SCCS ARCHIVE FILES

This the recommended process to identify those unreferenced SCCS files which can be safely deleted from the file system in order to recover some space.

The Korn shell script described in this section (named identifySCCSFilesToBeRemoved-o) will go thru each sourceId entry in the database and if the sourceId is no longer in use in any release, then the script will identify the corresponding SCCS files that can be safely removed (manually, by the administrator). This script will NOT actually do the removal, just the identification. To actually remove the files, see section “Deleting the unreferenced SCCS archive files” on page 21.

The script identifySCCSFilesToBeRemoved-o requires the use of the CMVC "Report -general" command which was added in CMVC 2.3.1.2 to certain combinations of databases and operating systems. Perform the following to see if your CMVC family supports this command:

```
Report -general Sequence
```

An example of the output is:

```
defect|5304
general|38293
source|113940
```

If your CMVC family does not support the Report -general, then stop. You may want to modify the Korn shell script to obtain the data without using the Report -general command.

Obtaining the tools and preparing the family

1. Login to the CMVC family user id.
2. Obtain a baseline of the amount of file system currently occupied by the vc tree of the CMVC family:

   ```
   du -k -s $HOME/vc
   ```

   It is important to have a baseline for comparison in order to judge if the procedure was worthy. Write down this amount.
3. Stop the CMVC family. It is important that NO new write-related activity is happening to the CMVC family while the cleanup procedure is in place.
4. Backup the CMVC family, which consists of 2 components:
   - The database.
   - The complete $HOME of the CMVC family user id. If there are symbolic links or mounted NFS file systems that are mapped this directory structure, then they also need to be backed up.
5. Download the following scripts, preferably into $HOME or $HOME/bin:
   - identifySCCSFilesToBeRemoved-o

     ```
     ftp://ftp.software.ibm.com/ps/products/cmvc/fixes/identifySCCSFilesToBeRemoved-o
     ```

     The -o suffix identifies this shell script as being for an Oracle database (using sqlplus).
   - handleListOfSourceIds

     ```
     ftp://ftp.software.ibm.com/ps/products/cmvc/fixes/handleListOfSourceIds
     ```

Details on the script: identifySCCSFilesToBeRemoved-o

The header of this Korn shell script contains the following information:

```
# Name:
# identifySCCSFilesToBeRemoved-o outputFile ffl startSourceId endSourceId “
#   See the code section below about Usage.
#
# Purpose: This is a sample shell script that identifies the sourceIds
#          of those SCCS files that can be removed from the $CMVC/vc tree
#          because these files are not longer referenced in the
#          proper places in the CMVC database.
#
#          This script will NOT perform the removal of the
#          files. The CMVC administrator will have to do it.
#
# Output:  The sourceIds of those SCCS files that can be removed are
#          listed in outputFile.
#
#          This output file can be processed by the companion script:
#              handleListOfSourceIds
#
# Warnings:
# * Stop the CMVC family before running this script.
# * Backup the CMVC family file system and the CMVC database before
#   executing this command.
#
# Notes:
# * This script needs to be run from the CMVC family user id;
#   the $LOGNAME variable identifies the CMVC family to be used.
# * It uses the environment variable $ORACLE_PASS
#    which has the password for the CMVC family database.
# * A temporary file "temp.log" is created by this script.
```

You can run the script without arguments to see the usage details:

```
identifySCCSFilesToBeRemoved-o

Usage:
identifySCCSFilesToBeRemoved-o outputFile ffl startSourceId endSourceId “
Where: outputFile will contain the list of files to remove
       startSourceId (optional, default is 1) indicates
         the first sourceId to handle, then it will be increased
         by 1, and so on, until endSourceId.
       endSourceId (optional, default is source value from
         Sequence table) indicates the last sourceId to handle.
Examples:
a) identifySCCSFilesToBeRemoved-o list1.lst
   The list of files to be removed are shown in list1.lst.
   Assuming that the source value in the Sequence table is 130,
   then, it the start sourceId is 1 and the end one is 130, thus
   it will be equivalent to (b) below:
b) identifySCCSFilesToBeRemoved-o list1.lst 1 130
c) identifySCCSFilesToBeRemoved-o list1.lst 10 20
   The first sourceId will be 10 and the last is 20.
Environment variables:
$LOGNAME identifies the CMVC family name.
$ORACLE_PASS identifies the password for the CMVC database.
```

Running the script: identifySCCSFilesToBeRemoved-o

1. A real example of executing the command is shown below.

   It is strongly recommended that at the beginning you should specify a small range of sourceIds, such as from 1 to 9 or 1 to 20, and become familiar with the script.

   If you do not specify the small range, then the default is to process ALL the files and this could take a while.

```
./identifySCCSFilesToBeRemoved-o list1.lst 10 20

identifySCCSFilesToBeRemoved-o: Starting.
This script identifies unreferenced SCCS files
which reside in the directory:  /disk2/cmpc3ora/vc
for the CMVC Family:            cmpc3ora

These files are listed in the output file:  list1.lst

The sourceIds of the files to be processed are
  Starting with sourceId:  10
  Ending with sourceId:    20
Processing sourceId=10
Processing sourceId=11
Processing sourceId=12
Processing sourceId=13
Processing sourceId=14
   This sourceId is unreferenced.
Processing sourceId=15
Processing sourceId=16
Processing sourceId=17
Processing sourceId=18
Processing sourceId=19
Processing sourceId=20

The number of unreferenced SCCS files found is:  1

End of script
```

2. If you want to store the displayed messages, then you can redirect the output of the script to a file (such as identify.out):

   ```
   ./identifySCCSFilesToBeRemoved-o list1.lst 10 20 2>&1 | tee identify.out
   ```

3. This script will produce a list of unreferenced SCCS files that can removed without compromising the integrity of the CMVC family. This script will not remove the files.

4. A temporary file "temp.log" is created by this script. The outputFile will contain the list of sourceIds (not of the actual files).

5. In this example, the contents of the file "list1.lst" is one line in my case that contains the number 14:

   ```
   14
   ```

   This means that the sourceId 14 identifies a SCCS file that can be removed.

6. You can use this list of sourceIds to manually identify and remove the corresponding SCCS files.

Because the identified sourceId does not have data anymore in the CMVC File table, it is not easy to identify if the file is text or binary. The type of file is critical in order to provide the exact path name for the SCCS file.

DELETING THE UNREFERENCED SCCS ARCHIVE FILES

Thus, from the information gathered in the section “Using release.complete and release.delete scripts” on page 13, it is possible now to have the fully qualified names of the unreferenced SCCS archive files that can be deleted, and then go the proper directory structure indicated by the sourceId in the outputFile, and remove the appropriate s.XX file or b.XX directory.

The script "handleListOfSourceIds" will handle the sourceIds listed in the output file of the first script "identifySCCSFilesToBeRemoved-o". This script will generate 2 new files:

- One output file has the fully qualified names of the unreferenced SCCS archive files that can be deleted together with their byte count and a total of the bytes to be reclaimed.
- The other output file is really a script with the remove commands for each file/directory.

  Note: We have done only basic testing with this removal script. If you do not have a backup copy of your complete CMVC family, then do not proceed with this procedure.

Details on the script: handleListOfSourceIds

The header of this Korn shell script contains the following information:

```
# Name:
#   handleListOfSourceIds inputFile
#   See the code section below about Usage.
#
# Purpose: This is a sample shell script that handles a file
#          that has the list of the sourceIds which have
#          SCCS files that can be removed from the $CMVC/vc tree
#          because these files are not longer referenced in the
#          proper places in the CMVC database.
#
#          This script will NOT perform the removal of the
#          files. The CMVC administrator will have to do it.
#
# Output:
#
#  a) A file with individual entries with the size of the fully
#     qualified SCCS file or directory that can be removed.
#     This file will have a total of the amount of space that
#     these entries occupy in the file system.
#     The file name is the inputFile name with a suffix of .out
#
#  b) A shell script file with individual entries that have the
#     specific command to remove the SCCS file or directory.
#     The file name is the inputFile name with a prefix of
#     'remove' and with a suffix of '.ksh'
#
# Warnings:
# * Stop the CMVC family before running this script.
# * Ensure that you have a recent backup the CMVC family file
#   system and the CMVC database before executing this command.
#
# Notes:
# * This script needs to be run from the CMVC family user id.
```

You can run the script without arguments to see the usage details:

```
./handleListOfSourceIds

Usage:
handleListOfSourceIds inputFile
Where: inputFile should contain the list of sourceIds of
       SCCS files or directories that are not referenced
       anymore in the CMVC family and which can be removed.
Example:
handleListOfSourceIds list1
  * The list of fully qualified file names to be removed
    with their sizes are shown in the file: list1.out
  * The shell script that contains the actual remove commands
    is called: remove.list1.ksh
```

Running the script: handleListOfSourceIds

1. A real example of executing the command is shown below.

```
./handleListOfSourceIds list1

handleListOfSourceIds: Starting.
This script identifies the fully qualified names of the
unreferenced SCCS files and directories which reside in
the directory:        /disk2/cmpc3ora/vc
for the CMVC Family:  cmpc3ora

These files are listed in the output file:  list1.out
The script that has the remove commands is:  remove.list1.ksh

Handling sourceId: 11
Handling sourceId: 13
Handling sourceId: 14
Handling sourceId: 30
ERROR: Cannot find the file associated with sourceId: 000030

These files are listed in the output file:  list1.out
The script that has the remove commands is:  remove.list1.ksh
End of script
```

2. The main explanations for the particular error for sourceId 30 (physical SCCS file or directory not found) are:
   - The physical file was previously deleted by the CMVC administrator.
   - A CMVC file was created and thus a new sourceId was reserved for it and the source field in the Sequence table was increased by 1. But then, the CMVC "File -undo" action was done and the CMVC file was not actually created. In this case, CMVC does not reuse the sourceId in the File table and the source field in the Sequence table is not touched. Thus, this is a sourceId that is not being used at all in the CMVC database and which does not have a corresponding SCCS file or directory.
3. An example of the list1.out file is:

```
List of fully qualified file names and
their file size in bytes.

/disk2/cmpc3ora/vc/0/0/0/0/s.11 = 405
/disk2/cmpc3ora/vc/0/0/0/0/b.13/ = 409600
/disk2/cmpc3ora/vc/0/0/0/0/s.14 = 214

The total byte count is: 410219
```

4. An example of the remove.list1.ksh is:

```
#!/usr/bin/ksh

rm /disk2/cmpc3ora/vc/0/0/0/0/s.11
rm /disk2/cmpc3ora/vc/0/0/0/0/b.13/*
rmdir /disk2/cmpc3ora/vc/0/0/0/0/b.13/
rm /disk2/cmpc3ora/vc/0/0/0/0/s.14

# end of script
```

5. WARNING: You must review the file "remove.list1.ksh" (or its equivalent if you used other names). You need to understand and agree what are the removal actions that will take place. You may want to edit the contents of this file. The only way to recover from a problem during this stage is by restoring the complete CMVC family directory.

   After reviewing the file and if you understand and agree with its contents, you may go ahead and execute it.

6. Obtain a baseline of the amount of file system of the vc tree for the CMVC family that is now available after the deletion of the unreferenced SCCS archive files:

   ```
   du -k -s $HOME/vc
   ```

   You can compare this amount of available space with the baseline taken before the removal. In that way you could assess if you obtained a meaningful reclamation of file system space.

7. Restart the CMVC family and perform some CMVC activities such as a release or level extract.

HOW TO FIX ERRORS WITH THE SCCS S.XX AND P.XX FILES

This chapter describes how to fix errors with the SCCS s.XX and p.XX files.

CMVC 0010-046, SCCS 1232-005, FILE S.XX DOES NOT EXIST

Problem:

I have a text file that is shown as locked from the CMVC Files window. When I try to check it in or to unlock it, I get the following error message:

```
/home/cmpc3db2/vc/0/0/0/0/s.12:
1232-005 '/home/cmpc3db2/vc/0/0/0/0/s.12' does not exist (ut4)
0010-046 An error occurred when the CMVC server software tried to
         unlock file text.txt in the
         version control file repository.
```

Answer:

Notice that the 1232-005 error message indicates that the s.12 file is missing. This is the file that contains all the SCCS change history and all the actual deltas. Therefore, the fix is to either restore the latest valid backup of the file s.12 or recreate a good facsimile of the missing file s.12 (and delete the CMVC file to avoid confusion):

As the CMVC administrator, change the directory to the one shown in the error message, in this case /home/cmpc3db2/vc/0/0/0/0.

Verify that indeed the SCCS file s.12 is missing. If it is not missing, then ensure that the file is owned by the CMVC family id and groupid, and that the file permissions are:

```
-r--r--r-- cmpc3db2 db2iadm2   s.12
```

If it is missing, then it is important to recreate a valid SCCS s.12 file. Ideally, you should try to find out if there is a backup version of the file or of the CMVC family and restore.

In case that a valid backup of s.12 cannot be restored, then you should try to delete the CMVC file to avoid confusion; however, the deletion cannot take place because the file is locked. Thus, it is necessary to "trick" SCCS with a dummy s.12 file to unlock the file and then to delete the CMVC file:

1. Copy another s.XX as s.12; it does not matter which s.XX is chosen, that is, the change history does not have to match.

2. Issue CMVC File -unlock to unlock the file.
3. Use CMVC File -delete to delete the file from CMVC; in that way, you can prevent further confusion with the file, in case that a user wants to checkout or extract a previous version of the file, which cannot be accessed anymore because the original s.12 file does not exist.
4. Delete the s.12 file, because it is not a valid version of the corresponding CMVC file; this s.12 file is a dummy file that helped to unlock the file; this s.12 file will not be used anymore.

CMVC 0010-046, SCCS 1232-005, FILE P.XX DOES NOT EXIST

Problem:

I have a text file that is shown as locked from the CMVC Files window. When I try to check it in or to unlock it, I get the following error message:

```
/home/cmpc3db2/vc/0/0/0/0/s.12:
1232-005 '/home/cmpc3db2/vc/0/0/0/0/p.12' does not exist (ut4)
0010-046 An error occurred when the CMVC server software tried to
         unlock file text.txt in the
         version control file repository.
```

Answer:

Notice that the 1232-005 error message indicates that the p.12 file is missing. This is the file that indicates that the file is locked in SCCS. Therefore, the fix is to recreate the missing lock file p.12.

As the CMVC administrator, change the directory to the one shown in the error message, in this case /home/cmpc3db2/vc/0/0/0/0.

Verify that indeed the SCCS lock file p.12 is missing. If it is not missing, then ensure that the file is owned by the CMVC family id and groupid, and that the file permissions are:

```
-rw-r--r-- cmpc3db2 db2iadm2   p.12
```

If the SCCS lock file p.12 is indeed missing, then recreate it as follows:

1. Issue the following line command:

   ```
   File -long -view text.txt -release rel1 | more
   ```

2. The important details are:

```
versionSID
nuVersionSID    1.1

locks:
    userLogin    newSID    checkOutDate        releaseName     fileNuPath
    ------------ --------- ------------------- --------------- ----------
    cmpc3db2     1.2       2000/04/07 09:09:50 rel1            text.txt
```

3. Another way to find out the future version (newSID) is to issue the following command:

   ```
   Report -view FilesOutView | more
   ```

   The output should look like this (showing only the relevant fields):

   ```
   path        releaseName     checkOutD  newSID       userLogin
   ----------- --------------- ---------- ------------ ----------
   text.txt    rel1            2000/04/07 1.2          cmpc3db2
   ```

4. Create a new file called p.12, as follows:

   ```
   1.1 1.2 cmpc3db2 00/04/07 09:09:50
   ```

   - In the first item, place the latest valid version of the file, either from versionSID or nuVersionSID. In this case is 1.1.
   - In the second item, place the value of newSID shown under the "locks" section. In this case is 1.2.
   - In the third item, place the name of the CMVC family. In this case is "cmpc3db2".

     In case that you do not use the name of the CMVC family, you may get the following error message when issuing a CMVC command:

     ```
     1255-153 User name is not in the p-file. (un2)
     ```

   - For the rest, place the date and time stamp "checkOutDate" as shown under the "locks" section. However, SCCS expects the year to be specified with only 2 digits, so, in this case, the value is "00/04/07 09:09:50".

     Note that it does not seem to be critical that the date stamp should be the original one. It seems that any valid date/time stamp is sufficient.

5. Try again the CMVC File -checkin or -unlock command.

CMVC 0010-046, SCCS 1232-005, FILES S.XX AND P.XX DO NOT EXIST

Problem:

I have a text file that is shown as locked from the CMVC Files window. When I try to check it in or to unlock it, I get the following error message:

```
/home/cmpc3db2/vc/0/0/0/0/s.12:
1232-005 '/home/cmpc3db2/vc/0/0/0/0/s.12' does not exist (ut4)
0010-046 An error occurred when the CMVC server software tried to
         unlock file text.txt in the
         version control file repository.
```

I noticed that the p.12 and s.12 files in /home/cmpc3db2/vc/0/0/0/0/ do not exist.

Answer:

Perform the following tasks:

1. Recreate the s.XX file; see “CMVC 0010-046, SCCS 1232-005, file s.XX does not exist” on page 25.
2. Recreate the p.XX file; see “CMVC 0010-046, SCCS 1232-005, file p.XX does not exist” on page 26.

An example of the concrete sequence of steps to be performed is shown below. This assumes that the file s.14 is a valid SCCS file that is going to be used as a dummy file.

```
mkdir  -p  vc/0/0/0/0/
cp  s.14  vc/0/0/0/0/s.12
echo "1.1 1.2 cmpc3db2 00/04/07 01:01:01" > vc/0/0/0/0/p.12
File -verbose -unlock text.txt -release rel1
If successful, then do File -delete on that file and
remove the file vc/0/0/0/0/s.12
```

CMVC 0010-046, SCCS 1232-005, FILE S.BINARY DOES NOT EXIST

Problem:

I have a binary file that is shown as locked from the CMVC Files window. When I try to check it in or to unlock it, I get the following error message:

```
/home/cmpc3db2/vc/0/0/0/0/b.14/s.binary:
1232-005 '/home/cmpc3db2/vc/0/0/0/0/b.14/s.binary' does not exist (ut4)
0010-046 An error occurred when the CMVC server software tried to
         unlock file binary.exe in the
         version control file repository.
```

Answer:

Notice that the 1232-005 error message indicates that the b.14/s.binary file is missing. This is the file that contains all the SCCS change history and all the actual deltas. Therefore, the fix is to either restore the latest valid backup of the file b.14/s.binary or recreate a good facsimile of the missing file b.14/s.binary (and delete the CMVC file to avoid confusion):

As the CMVC administrator, change the directory to the one shown in the error message, in this case /home/cmpc3db2/vc/0/0/0/0/b.14.

Verify that indeed the SCCS file s.binary is missing. If it is not missing, then ensure that the file is owned by the CMVC family id and groupid, and that the file permissions are:

```
-r--r--r-- cmpc3db2 db2iadm2   s.binary
```

If it is missing, then it is important to recreate a valid SCCS s.binary file. Ideally, you should try to find out if there is a backup version of the file or of the CMVC family and restore.

In case that a valid backup of s.binary cannot be restored, then you should try to delete the CMVC file to avoid confusion; however, the deletion cannot take place because the file is locked. Thus, it is necessary to "trick" SCCS with a dummy s.binary file to unlock the file and then to delete the CMVC file:

1. Copy another b.XX/s.binary as b.14/s.binary; it does not matter which b.XX/s.binary is chosen, that is, the change history does not have to match.
2. Issue CMVC File -unlock to unlock the file.
3. Use CMVC File -delete to delete the file from CMVC; in that way, you can prevent further confusion with the file, in case that a user wants to checkout or extract a previous version of the file, which cannot be accessed anymore because the original s.binary file does not exist.
4. Delete the s.binary file, because it is not a valid version of the corresponding CMVC file; this s.binary file is a dummy file that helped to unlock the file; this s.binary file will not be used anymore.

CMVC 0010-046, SCCS 1232-005, FILE P.BINARY DOES NOT EXIST

Problem:

I have a binary file that is shown as locked from the CMVC Files window. When I try to check it in or to unlock it, I get the following error message:

```
/home/cmpc3db2/vc/0/0/0/0/b.14/s.binary:
1232-005 '/home/cmpc3db2/vc/0/0/0/0/b.14/p.binary' does not exist (ut4)
0010-046 An error occurred when the CMVC server software tried to
         unlock file binary.exe in the
         version control file repository.
```

Answer:

Notice that the 1232-005 error message indicates that the b.14/p.binary file is missing. This is the file that indicates that the file is locked in SCCS. Therefore, the fix is to recreate the missing lock file b.14/p.binary.

As the CMVC administrator, change the directory to the one shown in the error message, in this case /home/cmpc3db2/vc/0/0/0/0/b.14.

Verify that indeed the SCCS lock file b.14/p.binary is missing. If it is not missing, then ensure that the file is owned by the CMVC family id and groupid, and that the file permissions are:

```
-rw-r--r-- cmpc3db2 db2iadm2   p.binary
```

If the SCCS lock file b.14/p.binary is indeed missing, then recreate it as follows:

1. Issue the following line command:

   ```
   File -long -view binary.exe -release rel1 | more
   ```

2. The important details are:

   ```
   versionSID
   nuVersionSID    1.1

   locks:
       userLogin    newSID    checkOutDate        releaseName     fileNuPath
       ------------ --------- ------------------- --------------- ----------
       cmpc3db2     1.2       2000/04/07 09:09:50 rel1            binary.exe
   ```

3. Another way to find out the future version (newSID) is to issue the following command:

   ```
   Report -view FilesOutView | more
   ```

   The output should look like this (showing only the relevant fields):

   ```
   path        releaseName     checkOutD  newSID       userLogin
   ----------- --------------- ---------- ------------ ----------
   binary.exe  rel1            2000/04/07 1.2          cmpc3db2
   ```

   It is a good educated guess that if the newSID shown is 1.2, then the previous current SID is 1.1.

4. Create a new file called b.14/p.binary, as follows:

   ```
   1.1 1.2 cmpc3db2 00/04/07 09:09:50
   ```

   - In the first item, place the latest valid version of the file, either from versionSID or nuVersionSID. In this case is 1.1.
   - In the second item, place the value of newSID shown under the "locks" section. In this case is 1.2.
   - In the third item, place the value of the CMVC family. In this case is "cmpc3db2".

     In case that you do not use the name of the CMVC family, you may get the following error message when issuing a CMVC command:

```
1255-153 User name is not in the p-file. (un2)
```

- For the rest, place the date and time stamp "checkOutDate" as shown under the "locks" section. However, SCCS expects the year to be specified with only 2 digits, so, in this case, the value is "00/04/07 09:09:50".

  Note that it does not seem to be critical that the date stamp should be the original one. It seems that any valid date/time stamp is sufficient.

5. Try again the CMVC File -checkin or -unlock command.

CMVC 0010-046, SCCS 1232-005, S.BINARY AND P.BINARY DO NOT EXIST

Problem:

I have a binary file that is shown as locked from the CMVC Files window. When I try to check it in or to unlock it, I get the following error message:

```
/home/cmpc3db2/vc/0/0/0/0/b.14/s.binary:
1232-005 '/home/cmpc3db2/vc/0/0/0/0/b.14/s.binary' does not exist (ut4)
0010-046 An error occurred when the CMVC server software tried to
         unlock file binary.exe in the
         version control file repository.
```

I noticed that the p.binary and s.binary files in /home/cmpc3db2/vc/0/0/0/0/b.14 do not exist.

Answer:

Perform the following tasks:

1. Recreate the s.binary file; see “CMVC 0010-046, SCCS 1232-005, file s.binary does not exist” on page 28.
2. Recreate the p.binary file; see “CMVC 0010-046, SCCS 1232-005, file p.binary does not exist” on page 29.

An example of the concrete sequence of steps to be performed is shown below. This assumes that the file s.binary is a valid SCCS file that is going to be used as a dummy file.

```
mkdir  -p  vc/0/0/0/0/b.14
cp  s.binary  vc/0/0/0/0/b.14/s.binary
echo "1.1 1.2 cmpc3db2 00/04/07 09:09:50" > vc/0/0/0/0/b.14/p.binary
File -verbose -unlock binary.exe -release rel1
If successful, then do File -delete on that file and
remove the file vc/0/0/0/0/b.14/s.binary
```

HOW TO FIX SOME COMMON ERRORS

This chapter describes how to fix some common errors when the database and the vc tree of a CMVC family are out of sync.

HOW TO DEAL WITH THE ERROR 0010-045 ON A RE-CREATED FAMILY

Problem:

I had an existing CMVC family for prototyping/pilot work, and now I want to remove the database and recreate the database again, but when I try to do ANYTHING with files, I get the following error:

```
0010-045 An error occurred when the CMVC server software tried to
         check in file CSDRAMCTL.net.v to the
         version control file repository.
```

Answer:

In this particular case when the CMVC family is recreated, one very likely cause for the error 0010-045 is that the vc file system is stale. For example, a user creates a test family, then does "rmdb" (but does not do "rmfamily"), then "mkdb". When a new file is created, SCCS will try to create a new archive on top of an existing (stale) one and the error message is displayed.

The solution is to perform the following commands:

1. rmdb
2. rmfamily
3. mkdb
4. restart the family and try again.

HOW TO DEAL WITH THE ERRORS 0010-113 AND 0010-045

Problem:

When checking in a file, the following error messages are shown:

```
1.3
diff: files too big, try -h
0 inserted
0 deleted
93140 unchanged
0010-113
*** WARNING: A Version Control Error Has Occurred. ***

         The previous version control files have been restored.
         Contact the family administrator for assistance.
0010-045 An error occurred when the CMVC server software tried to
         check in file CSDRAMCTL.net.v to the
         version control file repository.
```

Notice the error message "diff: files too big, try -h".

The original file is about 1 MB but the new checkin has now changed to about 2 MB.

Answer:

There are some things that might help:

1. Make a backup copy of the extracted file in the file system.
2. Extract the affected file again from CMVC, then check this copy back into CMVC. Specify the verbose/confirm action and notice that no changes were done to the file. This action should execute correctly. This will indicate that the SCCS archive file for the file is fine.

   In this checkin action failed, then something is not right with the SCCS files.

   If the checkin was fine, then may want to "undo" it, to avoid having a file change that actually did not have any changes in the file.
3. If the checkin of the unmodified file was successful, then view the original backup copy and check if there are ANY lines that have more than 512 characters. If so, then this will cause a problem with SCCS, which cannot handle text lines larger than 512 bytes.

   If this is the case, then you have to decide if you really want to have lines with more than 512 characters. If so, then you will need to delete and destroy this text file and commit the changes, then create a file that is binary, which does not have the 512 byte restriction per line. By default, the type of a file is text; thus, you need to ensure that the binary type is explicitly specified.

   By the way, it is not possible to change the type of a file in CMVC from text to binary because text files use forward deltas and binary files use backward deltas. The only way to change the type is to delete/destroy the file and then to create it again with the desired type.
4. If there are no lines that are larger than 512 bytes, then at this point the most likely cause is due to the AIX utility "diff" which has some limitations on very large files (I could not find

the information about these limits, though). In this case, the workaround is the same: to change the type of the file from text to binary.

If you are using HP-UX 10.20, then you must ensure that you have the latest patches for SCCS (see “What are the necessary patches for SCCS for the Year 2000” on page 4).

HOW TO DEAL WITH THE ERROR 0010-350 AND 0010-045

Problem:

During the checkin of a binary file, the transaction was interrupted and now the database and the vc tree are not in sync. Specifically, when the next attempt to checkin the file is done, then the following errors are displayed:

```
0010-350 The error (No such file or directory) occurred when the CMVC
         server software
         processed function vcCopy() on the file with path
         name /home/sa27e/vc/0/2/3/3/b.36/p.binary.

         Check that the path name, the file permissions, and
         the directory permissions are correct.  If problems
         persist, contact the system administrator or the family
         administrator.
0010-045 An error occurred when the CMVC server software tried to
         check in file ASIC/sa27e/gds2/io/BC2550T_PM_B.gds.Z to the
         version control file repository.
```

Answer:

1. Take a look at the directory shown in the error message. For example:

   ```
   $ cd /home/sa27e/vc/0/2/3/3/b.36
   $ ls -al
   total 880
   drwxr-xr-x   2 sa27e    asicfams     512 Nov 04 00:51 .
   drwxr-xr-x 102 sa27e    asicfams    2048 Sep 13 16:43 ..
   -rw-r--r--   1 sa27e    asicfams  427651 Sep 13 16:32 1.1
   -rw-rw-rw-   1 sa27e    asicfams    5573 Nov 04 00:51 1.2d_temp
   -r--r--r--   1 sa27e    asicfams     210 Nov 04 00:51 s.binary
   ```

   In this case, the p.binary file does not exist.

2. Verify with the user that the file was "locked" according to CMVC. You can perform the command and see the "locks" section:

   ```
   File -long -view fileName -release releaseName
   ```

3. Verify that the file is shown as checked-out:

   ```
   Report -view FilesOutView
   ```

4. Rename specific files prior to the fix with the extension "wrong.new". These renamed files will not be referenced by SCCS or CMVC and can be deleted later on:

```
mv 1.2d_temp  1.2d_temp.wrong.new
mv s.binary   s.binary.wrong.new
```

5. Restore the s.binary file from a backup.

   In this case, the file only referenced the 1.1 version of the file.

6. Created a p.binary file using the command "get -e p.binary" while in the binary file directory.

7. Retry the file checkin. This time it should be successful.

8. Now the binary directory appears as follows:

```
$ ls -al
total 1768
drwxr-xr-x   2 sa27e    asicfams     512 Nov 05 10:21 .
drwxr-xr-x 102 sa27e    asicfams    2048 Sep 13 16:43 ..
-rw-r--r--   1 sa27e    asicfams  427651 Sep 13 16:32 1.1
-rw-r--r--   1 sa27e    asicfams  447943 Nov 05 10:21 1.2
-rw-rw-rw-   1 sa27e    asicfams    5573 Nov 04 00:51 1.2d_temp.wrong.new
-rw-r-----   1 sa27e    asicfams       0 Nov 05 10:11 binary
-r--r--r--   1 sa27e    asicfams     210 Nov 05 10:20 s.binary
-r--r--r--   1 sa27e    asicfams     210 Nov 04 00:51 s.binary.wrong.new
```

HOW TO DEAL WITH THE ERROR 0010-511

Problem:

When trying to extract a file, the following error messages are shown:

```
0010-511 An error occurred when the CMVC server software tried to
         get file /radcmvc/radfam1/vc/0/7/2/1/s.71.

         Contact the family administrator for assistance.
```

Answer:

There are several possibilities for this error:

- To extract a file, the CMVC server daemon (cmvcd) requires read and execute access as the owner of the files in the vc tree. The extracted file is sent to the CMVC client via TCP/IP. Therefore, the directories in the vc tree should have as MINIMUM the permissions 750 or 740 (755 is fine).
- Ensure that the directories and files under the vc tree are owned by the CMVC family user id.

- Take a closer look at ALL the preceding messages, which may indicate other problems, such as:
  - A SID that does not exist; in this case, ensure that you specify the correct one:

    ```
    /adfam1/vc/0/7/2/1/s.71: get: The SID you specified does not exist.
    Use the sact command to check the p-file for existing SID numbers. (cm20)

    or

    /test/vc/0/0/0/0/s.02 /u/cmvctest/vc/0/0/0/0/z.02' does not exist (ut4)

    or

    /home/fam1/vc/0/1/0/1/s.86: get: Cannot lock the specified file.
            Check path name and permissions or
            wait until the file is not in use (cm4)
    ```

    In this case, specify the correct SID.
  - There are not enough resources to handle new processes in the host:

    ```
    /cmvc/postal/vc/0/3/2/5/s.25: get: Cannot create another process at this
    Try again later or use local problem reporting procedures. (co20)
    ```

    In this case, stop other applications, or increase resources or reboot the machine.
  - Wrong permissions or ownership:

    ```
    /home/fam1/vc/0/0/0/0/s.11 /home/fam1/vc/0/0/0/0/s.11  unreadable(ut5)

    or

    /afs/haifa/u/cmvcfam1/vc/0/0/0/0s.04
    directory /afs/haifa/u/cmvcfam1/vc/0/0/0/0' unwritable (ut2)

    or

    1255-064 You are not authorized to make deltas(co14)
    ```

    Modify the file permissions and ownership to the correct ones.
  - Problem during an "undo":

    ```
    /cmvc/as400/vc/0/0/1/9/s.41
    1255-047 The SID does not exist. (cm20)
    ```

    Also, there is an R next to version 1.15 of the SCCS file:

    ```
    ^Ah02601
    ^As 00000/00000/00246
    ^d R 1.15 93/05/28 12:54:10 as400 15 14
    ```

It looks like something bad happened during an undo. The R that is located to the left of version 1.15 of the SCCS file means this delta was Removed. Can you verify if version 1.14 of the file is the "latest"?

It appears that SCCS thinks that version 1.14 is the latest but CMVC thinks that version 1.15 is the latest.

To fix the problem, do:

1. Edit the SCCS file to change the 'R' to a 'D'.
2. Run 'admin -z' on the file to recompute the SCCS checksum.

– Possible lack of enough file system space:

```
get: error, unable to write workfile "<stdout>" (disk full?).
```

Ensure that you have enough file system space for the directory of the CMVC family.

0010-682: UNABLE TO CHECKOUT A LOCKED FILE WITH FORCE OPTION

Problem:

A user already has checked out a file that is common between 3 releases:

```
New Path Name      Release Current Committed Locked by
-------------      ------- ------- --------- ---------
admin/admincmd.doc dmb_10  1.4     1.4       xyz
admin/admincmd.doc dmb_12  1.4     1.4
admin/admincmd.doc dmb_31  1.4     1.4
```

The file is common to all three releases and locked in the first release. The user is trying to check-out the file from the third release by using the 'force' option and gets the following message:

```
0010-682 An error occurred in SCCS when the CMVC server software
         tried to check out file admin/admincmd.doc.

         The version control failed to create a lock for
         this file within the version control file tree.
```

Answer:

It is recommended to perform the following actions:

1. Identify the location of the lock for that file, in this case it is a p.binary.
2. Make a backup copy of the p.binary, just in case.
3. Use the SCCS command:

```
unget -r1.4.1.1 s.binary
```

Note that the important aspect here is to edit the p.XX file remove all the entries starting from the 2nd line; that is, leave only the first entry.

4. Use File -unlock -release dmb_10
5. Verify that p.binary is gone and that no locks show for the file.
6. Verify everything is correct by checking the file out again.

HOW TO DEAL WITH THE ERROR 0010-781

Question:

I got the following error message when trying to checkin a file:

```
0010-781 The CMVC server has determined there is inadequate
            file space to receive a file of size XXXX bytes.
```

Answer:

A file is temporarily stored in /tmp on the CMVC server before it is checked into the CMVC server.

The /tmp directory on the server must be large enough to hold 2 times the size of the SCCS archive (s.XX) file for a SCCS or PVCS delta file.

If this file is binary (that is, non-ascii characters, long lines or you selected binary), make sure you have 3 times the size of the file in /tmp on the server.

Remember that more than one user can simultaneously check in files and /tmp must be large enough to hold all these files.

CMVC ERROR MESSAGES 0010-788 AND 0010-045

Problem:

When I try to check in a file in CMVC, I am getting the following error messages:

```
0010-788 The CMVC server could not create the version control
         recovery file.
         Contact the family administrator for assistance.
0010-045 An error occurred when the CMVC server software tried to
         check in file os2/ui/ui.mak to the
         version control file repository.
```

Answer:

CMVC is having problems in creating a recovery file for dealing with SCCS. Some recommendations are:

- There might not be enough available space in the $HOME file system of the CMVC family. Specifically, under $HOME/vc, which is the place where the SCCS archive files are stored and which are handled by CMVC. You can use the following command to find out the available space for the file system where the CMVC family resides:

  ```
  AIX, Solaris:  df -k
  HP-UX:         bdf
  ```

- There might not be enough available space in the /tmp file system. As a general rule, the size for /tmp should be 3.2 times larger than the largest file stored in CMVC. For example, if you have a large binary file with a size of 50 MB, then you will need 160 MB of free space in /tmp in both the CMVC server and the CMVC client. You can use the following command to find out the available space in the /tmp file system:

  ```
  AIX, Solaris:  df -k /tmp
  HP-UX:         bdf   /tmp
  ```

- Look at the syslog (its location is mentioned in /etc/syslog.conf) and see if there are other messages issued around the same time that the 0010-788 was issued. For example, some times the error 0100-350 is shown too; for more details on this error, see “How to deal with the error 0010-350 and 0010-045” on page 35.

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
IBM, AIX, CMVC DB2 Universal Database	IBM Corporation
Oracle	Oracle Corp.

END OF DOCUMENT °