Migration from CMVC 2.3 to TeamConnection 2.0

Document Number TR 29.2254

Clifford Meyers, Lee Perlov and Angel Rivera

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

ABSTRACT

This technical report provides a procedure and tools that aid in the migration from Configuration Management and Version Control (CMVC) to its successor product, TeamConnection. The objective is to provide structure, reduce errors and save time by selecting only appropriate data for the migration process.

ITIRC KEYWORDS

CMVC

TeamConnection

ABOUT THE AUTHORS

CLIFFORD J. MEYERS

Mr. Meyers is an advisory programmer and the technical team lead for IBM Global Services Distributed Configuration Management Services. He joined IBM in 1985 as support for AIX/370 and AIX/ESA before accepting his current assignment in 1993.

The IBM Global Services organization in Research Triangle Park, NC, provides consulting and support services for over 7,000 internal and external customers.

Cliff was also involved in the development of the original Motif GUI for CMVC and all of the tools in the appendices of this document.

LEE R. PERLOV

Mr. Perlov is a Staff Software Engineer 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.

ANGEL RIVERA

Mr. Rivera is a Staff Software Engineer in the TeamConnection development group and team lead for CMVC development. He is also in the development group for TeamConnection. 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

ABSTRACT . . . iii
ITIRC KEYWORDS . . . iii

ABOUT THE AUTHORS . . . v
Clifford J. Meyers . . . v
Lee R. Perlov . . . v
Angel Rivera . . . v

Figures . . . ix

Introduction . . . 1

Preparing a plan for the migration . . . 3
Day by Day Plan . . . 3
Day: TC - 2 days (server side) . . . 3
Day: TC - 2 days (clients side) . . . 4
Day: TC - 2 days (users and usability) . . . 4
Day: TC - 1 day (server side) . . . 5
Day: TC - 1 day (client side) . . . 5
Day: TC - 1 day (users and usability) . . . 5
Day: TC (server side) . . . 5
Day: TC (client side) . . . 6
Day: TC + 1 day (server side) . . . 6

Preparing CMVC family for migration . . . 7
What to do to CMVC before migration . . . 7
Summary of important CMVC preparation steps (from TR 29.2232) . . . 7
Important assumptions about the migration from CMVC . . . 8

Selecting the data to be migrated . . . 9
Whether to clone . . . 9
Whether to prune . . . 10
Disk space considerations . . . 10
Time considerations . . . 10

Constructing the migration script . . . 11
Sample migration script: migcmvc.lst . . . 11
Notes about the sample migration script . . . 13

Description of mkmiglst.ksh . . . 13
Output from mkmiglst.ksh: actual script to be used with migcmvc . . . 14
Output from mkmiglst.ksh: migration instructions . . . 15

Migrating the data . . . 19
Ways to avoid trouble . . . 19
Actual migration steps . . . 20

Verification after the migration . . . 21

CMVC and TeamConnection are different . . . 23

Our experiences migrating CMVC families to TeamConnection . . . 25

Appendix A. Shell scripts to aid migrate from CMVC to TeamConnection . . . 27
Details of the tools . . . 27
Complete a release prior to TeamConnection migration . . . 27
Reassign user's work and delete users . . . 27
Delete a release from a family . . . 28
Creating a migcmvc migration script file . . . 28
Import a release into CMVC before migration . . . 29
Obtaining the Tools . . . 30
IBM Intranet . . . 30
Internet . . . 30

Appendix B. Bibliography . . . 33
CMVC 2.3 . . . 33
TeamConnection 2 . . . 33
TeamConnection 1, useful to TeamConnection 2 users . . . 33
Related Technical Reports . . . 33

Appendix C. Copyrights, Trademarks and Service marks . . . 35

FIGURES

1. File migcmvc.lst, top part. 12
2. File migcmvc.lst, bottom part. 12
3. Sample output file (myfile) of mkmiglst.ksh, top part. 14
4. Sample output file (myfile) of mkmiglst.ksh, bottom part. 15
5. Sample instructions file of mkmiglst.ksh, top part. 16
6. Sample instructions file of mkmiglst.ksh, bottom part. 17

INTRODUCTION

The current TeamConnection migration facility is a very general purpose tool and it has considerable capability and flexibility. Unfortunately, it does not enforce a methodology and this may result in mistakes that could be avoided if the user follows a well defined procedure.

The objective of this technical report (TR) and the documented tools, is to provide guidance that can make the migration easier, faster and more efficient.

These are the fundamental steps to migrate a family from CMVC to TeamConnection:

1. Prepare a plan of attack

   For more details, see “Preparing a plan for the migration” on page 3.

2. Prepare the CMVC family prior to migration

   For more details, see “Preparing CMVC family for migration” on page 7.

   This step is addressed in the technical report, TR 29.2232, "How to do migration tasks with CMVC". For convenience, we include a summary in this TR.

3. Select the data to be migrated

   For more details, see “Selecting the data to be migrated” on page 9.

4. Construct the migration script

   For more details, see “Constructing the migration script” on page 11.

5. Migrate the data

   For more details, see “Migrating the data” on page 19.

6. Verify the migration and shut down the CMVC family

   For more details, see “Verification after the migration” on page 21.

7. Get familiar with the differences between CMVC and TeamConnection

   For more details, see “CMVC and TeamConnection are different” on page 23.

This technical report is not a replacement for the documentation of these products and therefore, the migration functions will be explained only briefly in this TR. For more details, refer to the appropriate documentation listed in the bibliography, Appendix B, “Bibliography” on page 33.

PREPARING A PLAN FOR THE MIGRATION

We strongly suggest a rigorous plan for CMVC to TeamConnection migration for each family. This will require the participation of your entire team.

The basic idea is to do a practice migration, test the new family, assess the results, discard the new family, then do the migration again for real.

Plan for your team to extensively exercise your new TeamConnection family after migrating from CMVC for a very limited period of time, such as one full day. Do everything that you do normally in your development and reporting efforts. You can get a feel for a day's activity by looking at your CMVC audit log for a given family.

At the end of testing, have a quick meeting to assess the results. Once you pass a day of testing your new TeamConnection family with no "show-stoppers", then discard the TeamConnection family, do the migration again for real, and start using your new TeamConnection family.

One key ingredient in the smooth transition of your team from CMVC to TeamConnection is training to get familiar with TeamConnection. It is assumed in this migration plan that your group schedules training sessions for the users, prior to migration.

DAY BY DAY PLAN

You have to begin by selecting an appropriate day in which all the members of your team will test the migrated family. Select an appropriate day. Let's call it: TC day.

Based on that target date, there are several activities that need to be done prior, during and after that date.

Day: TC - 2 days (server side)

Ensure that all users of the new TeamConnection family have the proper host lists and access in the current CMVC family. In that way, the new family will be ready to handle the authentication of users after the migration from CMVC.

If you have a pilot project with TeamConnection, you may want to delete ObjectStore and the TeamConnection code from the server host.

Remove any old versions of ObjectStore or TeamConnection so that the server is point to the current code.

Install TeamConnection and ObjectStore from the latest CD-ROM.

Ensure that there is enough temporary disk space for handling all family servers and build agents in /tmp/ostore. ObjectStore requires an initial 10 MB of temporary disk space and 8 MB for each family server and agent.

A rough estimate of the number of family servers (daemons) is the closest cubit root of the total number of users. For example, if the total number of users for the new family is expected to be 50 users, then the closest cubic root is 4 (for 64). This means that you will need at least 42 MB of temporary disk space in /tmp/ostore:

1 MB	+	(4		8 MB	)	= 42 MB
initial required space		number of daemons		temp space for each daemon		

Plan the backup and restore strategy for your new family.

Day: TC - 2 days (clients side)

Ensure that everybody in the team has installed the TeamConnection client.

Remove any old versions of TeamConnection, such as versions used in a pilot project.

Prepare a list of the team members to match which TeamConnection areas these users are going to test and from which operating system platform.

The idea is to ensure that all relevant TeamConnection functions will be covered and that all relevant platforms will be tested for each client.

If this list is not prepared, then you risk a less comprehensive test. For example, everybody might open a defect, but will add/remove access lists.

Day: TC - 2 days (users and usability)

Schedule training sessions for the users of the CMVC family to provide them with the essentials of how to use a TeamConnection family.

A starting point is to present the comparison between CMVC and TeamConnection. For more details, see “CMVC and TeamConnection are different” on page 23.

Review with key users how you use your CMVC family (such as component structure, naming convention) and decide what changes, if any, will be done on the migrated data.

Do not take for granted that the current setup in the CMVC family is perfect and keep in mind that it might be possible to do some fine tuning with the setup in the new TeamConnection family just after the migration but before overall production use.

For example, take advantage of this transition to reorganize the component hierarchy (if needed) or to make changes to the naming conventions (such as changing release names

from very long with upper and lower case letters, to no more than eight lower-case only letters).

Day: TC - 1 day (server side)

The latest snapshot from the CMVC family is taken and migrated into the new TeamConnection family.

The new TeamConnection family is started and a sniff test is done by selected team members to ensure that so far the migration went fine and that the family is ready for TC day.

The new TeamConnection family should be up for few hours to allow the users to test their access to the family.

Rehearse the backup and restore strategy for the new family.

Day: TC - 1 day (client side)

All team members ensure that they are ready for TC day:

They should be able to connect to the new TeamConnection family from the connectivity point of view. This can be checked by issuing the command "ping family" and if this is successful, then issue the command "teamc report -testServer"

They should have the proper host list and access list. This can be checked by issuing the command "teamc user -view userName", where userName is the name of each user.

Day: TC - 1 day (users and usability)

Deliver the training sessions for the users to become familiar with TeamConnection.

Day: TC (server side)

The new TeamConnection family is started with the desired number of family servers, which is roughly the cubit root of the total number of users.

For example, if you have 50 users, the closest cubit root is 4 (for 64):

```
teamcd yourFamily 4
```

The family administrator and selected members of the team should monitor the family, paying particular attention to not running out of disk space.

TeamConnection does NOT currently offer a "monitor" tool, however the family administrator can watch the activity in the TeamConnection family by viewing changes to the audit log (using "tail -f audit.log").

The family administrator may decide to stop the family for few minutes during the middle of the day and restart it again, just to see if there are any problems with this action.

At the end of the day, the TeamConnection family will be shut down.

Perform a backup of the family and then do a restore, to test your backup/restore strategy.

Day: TC (client side)

Every member of the team will spend time interacting with the TeamConnection family. First with the specific assignments and then later on in an exploratory mode.

Once there are several workareas and drivers, you should try to do a "normal" build. That is, extract the release and then start builds in all the platforms from the extracted code.

At the end of the day, you should have a short meeting to discuss the results and if necessary, plan to fix the main problems and repeat the test (at least, a scale down version of it, such as just for half a day).

Day: TC + 1 day (server side)

The new TeamConnection family should be deleted for the next test (if needed) or for the formal migration.

Perform the final migration for the family.

Start the family servers for the new family.

PREPARING CMVC FAMILY FOR MIGRATION

WHAT TO DO TO CMVC BEFORE MIGRATION

As stated in the introduction, this topic has already been covered by the technical report, TR 29.2232, "How to do Migration Tasks with CMVC" (see Appendix B, “Bibliography” on page 33 for more references). For convenience, we include a summary in this TR.

The TeamConnection migration utility will use CMVC client commands. The only requirement from TeamConnection is that the CMVC client be installed on the same system as the TeamConnection server for a migration to occur.

In principle, any CMVC client, version 2.2 or later, will be able to support TeamConnection migration. However, we recommend that you upgrade your version of the CMVC server to the latest supported version of CMVC 2.3 before migrating to TeamConnection.

The following shell script samples will help you to prepare the CMVC family for migration:

“Complete a release prior to TeamConnection migration” on page 27
“Reassign user's work and delete users” on page 27
“Delete a release from a family” on page 28

TR 29.2232 also provides procedures for performing very useful tasks, such as cloning a CMVC family.

SUMMARY OF IMPORTANT CMVC PREPARATION STEPS (FROM TR 29.2232)

Remove anything from the CMVC family you do not need anymore.

There are things that are not migrated but that should be cleaned up before running the TeamConnection migcmvc migration facility. For example, ensure that there are no Tracks in the "working" state.

Use the shell scripts documented in TR 29.2232 and this report, or manually clean up anything that will give you warnings or errors during the migration.

Make sure you are at the current version of CMVC, which is 2.3.0.22 at the time that we wrote this TR. Even though we can migrate from any CMVC client version 2.2 or later, we strongly recommend you use the most currently available version of CMVC.

Decide how you are going to stage your migration from CMVC to TeamConnection.

If you have multiple families, migration is more than one day's work, so you need to be ready to run both CMVC and TeamConnection in parallel while all the families are being migrated. See “Selecting the data to be migrated” on page 9 for more details.

IMPORTANT ASSUMPTIONS ABOUT THE MIGRATION FROM CMVC

Throughout this document, we will assume the following:

You should plan to do practice migrations because there are things you need to find out for yourself.

You should include in your plan enough time to do backups of the CMVC family at several points in the migration, such as just before the migration, after all data except files have been migrated, and after all data has been migrated.

Also, the migration facility is very flexible and thus, you can expect to decide to change options and try again.

You are migrating from UNIX to UNIX because all our tools are written in Korn shell.

Finally, you should learn the differences between CMVC and TeamConnection, and make sure the rest of the user community does too.

Although there is a lot common between the products, TeamConnection is different than CMVC. There are subtle differences that make TeamConnection much more powerful than CMVC.

If you do not understand the differences you will spend a lot of time fighting with the TeamConnection GUI and client commands, and you will not understand why things do not work the way you expected.

SELECTING THE DATA TO BE MIGRATED

Moving your data from a CMVC family to a TeamConnection family will require careful planning. These are the questions you need to ask yourself:

Will all of the releases from my current CMVC family be migrated, or only some of them?

If not all of the releases will be migrated, are there any releases that need to be kept in CMVC as reference only, while other releases will be migrated to be used in future development?

How important is the history of each file change?

Could the history of each file change be left in CMVC (which means that the CMVC family needs to be active) as a reference family? In addition, this will save time during the migration and disk space with the new TeamConnection family.

Is there enough disk space to keep both CMVC and TeamConnection families at the same time?

Is there enough CPU to keep CMVC and TeamConnection families at the same time?

Are NFS mounts for holding the CMVC family data being used?

How much time can the CMVC family be placed in maintenance mode in order to perform the migration?

The answers to these questions are critical because they can help you determine whether you will need to:

Clone any CMVC families

Prune data from CMVC families (using our shell scripts)

Remove NFS mounts, add disk space, move around disk space, etc.

Limit the amount of data to be migrated to save time or disk space; for example, get a snapshot of a release instead of the entire file change history.

WHETHER TO CLONE

If you are going to keep some releases in a CMVC family around for reference AND you have other releases that are going to continue to stay in production in CMVC then you should consider cloning your CMVC family.

By duplicating the contents of a CMVC family in another family, you can make one read-only and continue work in the other.

Cloning a CMVC family is documented in the TR 29.2232.

WHETHER TO PRUNE

Pruning a CMVC family is a good way to save space after migrating releases to TeamConnection or cloning a family.

Pruning eliminates confusion about which release in which family is the one to be updated, and it saves disk space that may be needed by TeamConnection. See “Delete a release from a family” on page 28 for details.

DISK SPACE CONSIDERATIONS

CMVC and TeamConnection use lots of disk space, as does all software development, therefore, we recommend pruning any duplicate releases, for example:

After migrating a release to TeamConnection use the pruning tool to remove the migrated release from the CMVC family.

After cloning a CMVC family into a read-only CMVC clone family and a CMVC development family, use the pruning tool to make sure there is only one copy of each release across CMVC families.

The pruning tool will remove files from the vc directory of the CMVC family whenever possible, thus saving the most disk space. See “Delete a release from a family” on page 28 for details.

TIME CONSIDERATIONS

Time spent migrating is time that your families are down, therefore the less data you migrate the less down time for your families. This is one of the best reasons to migrate only the releases you plan to continue developing. Leaving the other releases in CMVC will get your TeamConnection family up and available much faster.

In order to save time and effort, once you have migrated the active releases, put the CMVC family in read-only mode and take it OFF of your backup list. There is no reason to backup a family that does not change.

CONSTRUCTING THE MIGRATION SCRIPT

The utility to migrate a CMVC family to TeamConnection, called "migcmvc", is a front end to a generalized SQL query and update mechanism. This tool should be used only during migration.

The migcmvc utility from TeamConnection provides a command line environment from which you can issue the migration commands. Because the order of the commands for migration is very important, TeamConnection provides an input file that contains migrate commands that can be piped to the migcmvc utility to perform a generic migration.

Each of the migrate commands can be customized using SQL syntax to fine tune the selection of data to be migrated. For more details in this sample migration script, see “Sample migration script: migcmvc.lst.”

In order to make migration easier, this technical report describes a tool called "mkmiglst.ksh" that generates a customized input file for the migcmvc utility. This input file is much more useful than the generic migcmvc.lst file shipped with TeamConnection. For more details on this tool, see “Description of mkmiglst.ksh” on page 13 and “Creating a migcmvc migration script file” on page 28.

SAMPLE MIGRATION SCRIPT: MIGCMVC.LST

The sample migration script, migcmvc.lst, which is packaged with later versions of TeamConnection is shown in the next 2 figures (although they represent a single complete file): Figure 1 on page 12 and Figure 2 on page 12.

```
set batchsize 1
set decache     1
migrate    Users
migrate    HostView
migrate    Authority
migrate    Interest
migrate    Cfgcomproc
migrate    Cfgrelproc
migrate    Config
migrate     CompView    where 1=1 order by addDate
migrate     bCompView where name='root'
migrate    ReleaseView
migrate    EnvView
```

Figure 1. File migcmvc.lst, top part.

```
migrate    AccessView
migrate    NotifyView
migrate    DefectView
migrate    FeatureView
migrate     LevelView where 1=1 order by commitDate
migrate    TrackView
migrate    FixView
migrate    ApproverView
migrate    ApprovalView
migrate    LevelMemberView
set top x:\dir of files in release if migrating only the current version
migrate    FileView
migrate    SizeView
migrate    TestView
migrate    VerifyView
migrate    NoteView
```

Figure 2. File migcmvc.lst, bottom part.

Notes about the sample migration script

Actually, if you want to use the above migcmvc.lst input file, it is strongly recommended to perform the migration in the following steps.

1. Comment out the "migrate FileView" line in order to NOT migrate the files in the first round.

   The line to migrate the complete change history of a file, "migrate ChangeView", is not included in migcmvc.lst. If it is in your migration script it should be commented out also.

2. Stop the TeamConnection family and make a backup of the family.
3. Restart the TeamConnection family and perform the commented out line(s). mentioned above.

   By following this process you will have the flexibility to go back to a stable checkpoint in case that the migration of the files was not what you expected and you need to try again.

DESCRIPTION OF MKMIGLST.KSH

The mkmiglst.ksh tool can help you in preparing a script that can be used with the migcmvc utility, instead of using the sample migration file migcmvc.lst. For a summary of this tool, see “Creating a migcmvc migration script file” on page 28.

An example of using mkmiglst.ksh is shown below, in which we wish to migrate the releases "v203" and "v204":

```
mkmiglst.ksh -o myfile -r v2 3 -r v2 4
```

The mkmiglst.ksh tool will generate the following files, which are explained the following sections.

File: myfile.lst

This file contains the actual script that can be used with migcmvc.

File: myfile.lst.instructions

This file contains migration instructions on how to use myfile.lst with the migcmvc utility.

OUTPUT FROM MKMIGLST.KSH: ACTUAL SCRIPT TO BE USED WITH MIGCMVC

This file contains the actual script to be used as input to migcmvc.

The sample myfile.lst file is shown in Figure 3 on page 14 and Figure 4 on page 15. Note that because some lines are larger than 80 characters, we needed to show them in multiple lines in these figures; however, in reality they take only one line in the actual script.

```
# If you use release.delete.ksh set maxerrors to
set maxerrors 1
# You can increase decache to increase performance if you do not approach
# your paging space limit during tests
set decache 1
# This is the recommended batchsize
set batchsize 1
# Migrating individual tables
migrate Users
migrate HostView
migrate Authority
migrate Interest
migrate Cfgcomproc
migrate Cfgrelproc
migrate Config
migrate CompView
migrate bCompView where name = 'root'
migrate ReleaseView where name in (select releaseName from DefectView)
migrate EnvView
migrate AccessView
migrate NotifyView
migrate DefectView where state in ('open','returned','working') OR    \
   name in (select defectName from TrackView where releaseName in       \
   ('v2 3','v2 4') AND defectType = 'defect')
migrate FeatureView where state in ('open','returned','working') OR \
   name in (select defectName from TrackView where releaseName in       \
   ('v2 3','v2 4') AND defectType = 'feature')
migrate TrackView where state in ('commit','complete') AND \
   releaseName in ('v2 3','v2 4')
migrate FixView where releaseName in ('v2 3','v2 4') AND    \
   state = 'complete'
migrate ApproverView
```

Figure 3. Sample output file (myfile) of mkmiglst.ksh, top part.

```
migrate ApprovalView where releaseName in ('v2 3','v2 4')
migrate LevelView where releaseName in ('v2 3','v2 4') order by    \
    commitDate
migrate LevelMemberView where releaseName in ('v2 3','v2 4')
set top /home/cmvcsupt/extract/v2 3
! mkdir -p /home/cmvcsupt/extract/v2 3
! Release -extract v2 3 -committed -node keithp -root \
   /home/cmvcsupt/extract/v2 3 -uid 2 2 -gid 1
# We recommend commenting out the FileView command
# and running it separately, after the database has been backed up.
migrate FileView where versionSID = nuVersionSID AND    \
   releaseName = 'v2 3' AND dropDate is null
set top /home/cmvcsupt/extract/v2 4
! mkdir -p /home/cmvcsupt/extract/v2 4
! Release -extract v2 4 -committed -node keithp -root \
   /home/cmvcsupt/extract/v2 4 -uid 2 2 -gid 1
# We recommend commenting out the FileView command
# and running it separately, after the database has been backed up.
migrate FileView where versionSID = nuVersionSID AND    \
   releaseName = 'v2 4' AND dropDate is null
migrate SizeView where releaseName in ('v2 3','v2 4')
migrate TestView where releaseName in ('v2 3','v2 4')
migrate VerifyView where defectName in (select name from DefectView          \
   where state in ('returned','working')) OR defectName in (select name    \
   from FeatureView where state in ('returned','working')) OR defectName \
   in (select defectName from TrackView where releaseName in                  \
  ('v2 3','v2 4'))
migrate NoteView
migrate CoreqView
quit
```

Figure 4. Sample output file (myfile) of mkmiglst.ksh, bottom part.

Output from mkmiglst.ksh: migration instructions

This file contains migration instructions on how to use myfile.lst with the migcmvc utility.

The myfile.lst file is shown in Figure 5 on page 16, and Figure 6 on page 17.

```
PRE-MIGRATION RECOMMENDATIONS
-----------------------------
1.  Create a superuser login and a host list entry for the
     TeamConnection family account in the CMVC family to be migrated.
2.  Backup the CMVC family and database.
3.  Cleanup and commit the work in progress in the target release(s)
     using the following shell scripts.
       reassign.work.ksh
       release.complete.ksh
       release.delete.ksh
4.  The TeamConnection account needs to have the following environment
    variables  set:
       CMVC_FAMILY=<source  family>
       TC_FAMILY=<target  family>
        PATH=PATH:<location of CMVC executables>
     Note that the NLSPATH variable needs to have both the CMVC and the
     TeamConnection paths for the message catalog.
5.  Create an extract directory in the TeamConnection family $HOME
     directory that is NFS mountable from the CMVC server:
       ${FAMILYHOME}/extract
6.  Copy CMVC ${FAMILYHOME}/bin contents to the TeamConnection
    directory  ${FAMILYHOME}/bin.
7.  Copy CMVC ${FAMILYHOME}/config/userExits file to the TeamConnection
    file        ${FAMILYHOME}/config/userExit.
8.  Copy CMVC ${FAMILYHOME}/configField/DefectConfigTbl to TeamConnection
    file        ${FAMILYHOME}/cfgField/Defect.tbl
9.  Copy CMVC ${FAMILYHOME}/configField/FeatureConfigTbl to TeamConnection
    file        ${FAMILYHOME}/cfgField/Feature.tbl
1 . You have to update the format files for the configurable fields in
     order to put the appropriate sequence number for each field.
```

Figure 5. Sample instructions file of mkmiglst.ksh, top part.

```
MIGRATION STEPS
---------------
1.  print "myfile.lst" | migcmvc migcmvc.out 2>&1 | at now
2.  For status: tail migcmvc.out
3.  To reference this instructions, see the file: myfile.lst.instructions


POST-MIGRATION RECOMMENDATIONS
------------------------------

1.  Create CMVC userExits and approval records to block any further
     work on the CMVC release.
2.  Execute the following TeamConnection commands to recreate the
     .ld  files:
        teamc Report -view authority -raw > authorit.ld
        teamc Report -view cfgcomp      -raw > comproc.ld
        teamc Report -view cfgrel       -raw > relproc.ld
        teamc Report -view interest    -raw > interest.ld
        teamc Report -view config -raw | awk -F\| '{ print "$1|$2|$3|$4|$5|"$6""
}
' > config.ld

end of instructions
```

Figure 6. Sample instructions file of mkmiglst.ksh, bottom part.

MIGRATING THE DATA

WAYS TO AVOID TROUBLE

You are likely to perform the migration task several times, and thus, we suggest several tasks to make each migration easier and more reliable:

Make sure the CMVC family is in maintenance mode. In that way, the users of the CMVC family will not be able to create or modify objects in the CMVC family, which could cause a conflict during the migration to TeamConnection.

Note: The migration script generated by mkmiglst.ksh includes a CMVC Release -extract. Currently, this command does not run when CMVC is in maintenance mode. The Release -extract command is included mostly for documentation. We suggest performing the extract before running migcmvc. However, if there a way for you to prevent user access to the CMVC family during migration (with CMVC in normal mode), the Release -extract can be run from the migration script.

Take note of how long it takes you to perform the practice migration so you know for how long CMVC will need to be in maintenance mode.

Note: If CMVC is not in maintenance mode and users have access to the family, then you risk migrating inconsistent data and corrupting your TeamConnection family.

Test the user exits in TeamConnection. There are subtle differences in behavior with respect to CMVC.

Qualify each command with the appropriate releases names.

Do not qualify users, hosts, components, access and notification, because they are global to the family.

Be sure you are running with at least the version 2.0.3 of TeamConnection. To find out the version, run the following command in the "$HOME_TEAMC/bin" directory:

```
what teamcd | grep teamc
```

You do NOT need to use dbcreate when migrating from CMVC. The migcmvc utility will create your family database. Therefore, you need to make sure any current family database, $TC_DBPATH/*.tcd, is deleted.

When running the migration step, we recommend redirecting your output to a file as reference.

The example below redirects both normal output and error messages to the same file, in the correct order:

```
migcmvc 2>&1 | tee migcmvc.out
```

If this is a practice run, we also recommend timing your test:

```
time migcmvc 2>&1 | tee -a migcmvc.out
```

ACTUAL MIGRATION STEPS

To actually migrate the CMVC family to TeamConnection, follow the instructions provided with the output file generated from mkmiglst.ksh which has a file extension of "instructions".

For an example, see Figure 5 on page 16 and Figure 6 on page 17.

VERIFICATION AFTER THE MIGRATION

After the migration has been performed, it is necessary to verify that the new TeamConnection family has valid data. You can perform the following verifications:

First of all, you will need to review the output file from migcmvc, called "migcmvc.out", to see if there are any errors or warnings reported by the migration utility.

Run CMVC "Report" and TeamConnection "teamc report" commands to ensure that the number of records match for different objects, such as users, components.

You will need to use the "-where" clause to limit the number of records found by CMVC to match TeamConnection (as defined by the .lst file generated by the tool).

You may create your own shell scripts that have pairs of commands, the first item in the pair for CMVC and the second for TeamConnection.

To verify the migration of files (parts), do a release or driver extract from the TeamConnection family.

After the parts have been extracted, select some parts at random an verify that their contents are valid.

If the migration script was not properly setup, it might be possible that the part objects (the information about the parts) were migrated properly, but the actual data (parts) associated with those objects were NOT migrated; this could happen if CMVC_TOP was not set for the migration of FileView, or if the files were not extracted first.

One way to roughly verify that all the files were extracted correctly from the TeamConnection family is to use the "du -s" command on the top directory of the extracted files. You will need to extract also the files from CMVC into another directory and use also the "du -s" command to compare the number of blocks.

CMVC AND TEAMCONNECTION ARE DIFFERENT

In order to become proficient in using your new TeamConnection family, it is important to understand the differences in behavior between CMVC and TeamConnection.

The technical report, TR 29.2253, **Comparison between CMVC 2.3 and TeamConnection 2** highlights the differences between these products and describes briefly the new functions in TeamConnection.

The best overview of TeamConnection, which is equivalent to **CMVC Concepts** is the Redbook, **Introduction to the IBM Application Development Team Suite**.

The TeamConnection User's guide is also a good reference.

OUR EXPERIENCES MIGRATING CMVC FAMILIES TO TEAMCONNECTION

When we prepared the plan to migrate our development CMVC family that contains the TeamConnection code, we had many decisions to make:

Do we migrate all of our releases or just the active ones?

We decided to only migrate the active releases.

Do we migrate all of the change history or just the currently committed versions of each file.

We choose to keep our CMVC family around, running in maintenance mode, and only migrate the most recently committed files. This minimized the amount of time we were down for migration and preserved in CMVC the historical information.

Do we clean up the CMVC family before the migration?

We decided to do a minimum cleanup of the CMVC family because it appeared that we would need to break the family into at least two TeamConnection families, one for further TeamConnection development and another for some of the CMVC code that is still being maintained.

As a result of the above decisions, here is how our migration went:

7000+ files (mostly text source files)

2 releases (most recent changes, with no change history)

Migrated from RS/6000 970

Migrated to RS/6000 c20

It took 3 hours to migrate

We did lots of practice runs, wrote the tools referenced in this document, and tuned what we were migrating along the way.

APPENDIX A. SHELL SCRIPTS TO AID MIGRATE FROM CMVC TO TEAMCONNECTION

All the tools used here are written in Korn shell and are available via the Internet or through IBM's intranet. The tools are updated periodically, so we are not providing the source in this document.

You can request help for each tool by entering the "-?" parameter, for example:

release.delete.ksh -?

DETAILS OF THE TOOLS

Complete a release prior to TeamConnection migration

The name of the shell script is:

release.complete.ksh

Help contents:

```
USAGE: release.complete.ksh -f <FAMILY> -r <RELEASENAME>
WHERE: FAMILY          is the CMVC family to process
       RELEASENAME is the active CMVC release
NOTE : script must be run by a SUPERUSER
```

This utility will complete work in a given release so that all file changes are committed.

Reassign user's work and delete users

The name of the shell script is:

reassign.work.ksh

Help contents:

```
USAGE: reassign.work.ksh -f <FAMILY> -s <FROMUSER> -t <TOUSER>
WHERE: FAMILY          is the CMVC family to process
       FROMUSER       is the CMVC login to be deleted
       TOUSER         is the CMVC login to assign work to
NOTE : script must be run by a SUPERUSER
```

This utility will reassign incomplete work from one active CMVC login to another active CMVC login. Incomplete work can be defined as:

defects owned/originated
features owned/originated
verification
components
releases
levels
environments
test records
size records
approval
approver
tracks
fix records
notifications deleted
access deleted
login deleted

Delete a release from a family

The name of the shell script is:

release.delete.ksh

Help contents:

```
USAGE: release.delete.ksh -f <FAMILY> -r <RELEASENAME> -d <DEFECT> -w
WHERE: FAMILY          is the CMVC family to process
       RELEASENAME is the active CMVC release
       DEFECT          is the CMVC defect for the track
NOTE : script must be run by the family account SUPERUSER
       w option define what sccs and mapfiles can be deleted
```

This utility will delete work in a given release. It will also identify a list of vc (SCCS/PVCS) files that are not related to any other release that can be removed to save disk space.

Creating a migcmvc migration script file

The name of the shell script is:

mkmiglst.ksh

Usage (split into 2 lines):

```
mkmiglst.ksh:
Purpose: Create a CMVC to TeamConnection migration listfile
Usage: mkmiglst.ksh [-h] [-f|-c]
       [-o <outputFile>] [-r <release>] ...
```

Help:

```
mkmiglst.ksh:
Purpose: Create a CMVC to TeamConnection migration listfile
Usage: mkmiglst.ksh [-h] [-f|-c]
        [-o <outputFile>] [-r <release>] ...
Description:
Generates a migration listfile to copy releases from CMVC to TeamConnection;
this migration listfile is input to the migcmvc migration tool.
This listfile contains the steps necessary to migrate all data related to
specified releases.    This listfile contains comments to assist the user
to manually customize the listfile for the migration process.
Instructions are displayed for using listfile with migcmvc; the instructions
are also saved in a file, myfile.instructions (by default)
If -o OutFile is used, the instructions are saved to OutFile.instructions.
Options:-o: Specify filename for migration listfile.    Default=myfile.lst.
         -f: Default. Migrate only FileView (snapshot of committed release).
         -c: Migrate ChangeView (complete history of release).
         -r ReleaseName: The release to be migrated.    Use -r for each release.
              If no releases are specified then the user is prompted for them.
Sample: # Listfile is myfile.lst, migrate FileView for two releases
         mkmiglst.ksh -o myfile.lst -r v2 3 -r v2 4
```

This utility generates a migration script file to be used as input for the migcmvc utility. An example of this file is shown in Figure 3 on page 14 and Figure 4 on page 15.

When the migration script is generated, instruction messages are sent to the screen; these messages are stored in the file with a prefix of ".instructions". An example of these messages is shown in Figure 5 on page 16, and Figure 6 on page 17.

Import a release into CMVC before migration

The name of the shell script is:

file.import.ksh

Help contents:

```
USAGE: file.import.ksh -f <FAMILY> -a <RELPATH> -m <MAPFILE> -r <RELEASE>
WHERE: FAMILY          is the CMVC family to process
       RELPATH         is the absolute path to the file tree
       MAPFILE         is the pathname component mapfile
       RELEASE         is the is the active CMVC release
NOTE : script must be run by a SUPERUSER
       MAPFILE format pathname|compname
       Assuming empty RELEASE
```

This utility will load files into CMVC from a given root directory. It requires that the user should provide a mapfile that maps the pathnames of the files to the components where the files are to be loaded into.

OBTAINING THE TOOLS

The tools described in this technical report can be downloaded as follows:

From the IBM intranet (only for IBM employees).

From the Internet (open to everyone).

Note: The tools referenced in this document are available from ftp sites instead of being included here so that we can make updates as necessary.

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 the section "Tools to migrate from CMVC to TeamConnection".

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/fixes/tr-tools
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/fixes/tr-tools
4. binary
5. get fileName
6. quit

APPENDIX B. BIBLIOGRAPHY

CMVC 2.3

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

TEAMCONNECTION 2

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

SC34-4551 TeamConnection, Administrator's Guide
SC34-4552 Getting Started with the TeamConnection Clients
SC34-4499 TeamConnection, User's Guide
SC34-45 1 TeamConnection, Commands Reference
SC34-45 TeamConnection, Quick Commands Reference

TEAMCONNECTION 1, USEFUL TO TEAMCONNECTION 2 USERS

A few publications that have not been updated for TeamConnection 2.0 are still useful.

SG24-4648 Introduction to the IBM Application Development Team Suite
83H9677 Staying on Track with TeamConnection Processes (Poster)

RELATED TECHNICAL REPORTS

TR 29.2232 How to do Migration Tasks with CMVC
It has a chapter on how to prepare a CMVC family for migration.

TR 29.2253 Comparison between CMVC 2.3 and TeamConnection 2

APPENDIX C. 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, IBM, CMVC, TeamConnection, RS/6	IBM Corporation
ObjectStore	Object Design, Inc.
UNIX	X/Open Co., Ltd.

END OF DOCUMENT