Evolution of a new VisualAge TeamConnection Family: Taking advantage of automation. Second in a series

Document Number TR 29.2357

Lee Perlov

VisualAge TeamConnection/CMVC Development
IBM Software Solutions
Research Triangle Park, North Carolina

ABSTRACT

VisualAge TeamConnection families evolve over time. As the family matures, things tend to change. All families can benefit from automation of tasks. This document, second in a series, continues to trace the growth of a typical VisualAge TeamConnection family and reflects on the impact of this automation.

ITIRC KEYWORDS

- VisualAge TeamConnection
- Family Administration
- User Exits
- Windows NT

ABOUT THE AUTHOR

Lee R. Perlov

Mr. Perlov is a Staff Programmer in the VisualAge TeamConnection/CMVC development group. He started working for IBM in 1985 in Gaithersburg, Md, 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.

Contents

Introduction

The VisualAge TeamConnection services team, one of the departments in our development group, started a VisualAge TeamConnection family using a PC running Windows NT about five months ago. The first installment of this series of technical reports discussed our initial decisions, the problems we encountered along the way, and what we did to improve the reliability of the family and our development process. We have recently added new users and responsibility for storing more products. In this document I focus on automation recently added to make this growing family easier to manage and more usable for a wider range of users.

Since our family has been up and stable for about five months, several of the VisualAge TeamConnection team members have been asked to move the data they currently store in our primary production family (where we manage the VisualAge TeamConnection source code) into the **dept_cfj** family. Also, we have several team members that would like to be able to extract our education materials for their use. These team members need limited access to the family. They have asked us to improve our automation and share more information about the family.

In order to meet the needs of new users, I have done the following to our family:

- Insured a reliable backup each night,
- Performed a vital records backup and reboot each weekend,
- Added a user exit that communicates helpful information to new users, and
- Added a new access group for users that will only be extracting files, but not updating them.

The evolution of a family is never really complete. So, I have included many of the normal administration tasks I have performed. Also, at the end of the document I have listed tasks I would like to perform in the future.

Overview of family changes

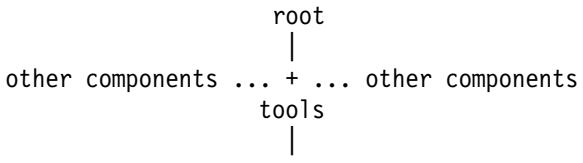
I am currently the system administrator as well as family administrator. So, I am performing most of the tasks listed below. The tasks performed on our family are organized by role, as defined in the **VisualAge TeamConnection Administrators Guide**.

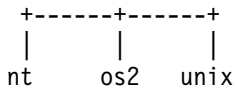
System Administrator's tasks

- Automate the periodic reboot procedures.
- Update the `Emergency Recovery Diskette`.
- Upgrade VisualAge TeamConnection to version 2.09. After updating to v209, there were several smaller tasks to perform:
  - Read README.txt. Since lots of software upgrades can cause problems, I always check with README.txt before upgrading other software.
  - Reinitialize Objectstore database (as directed in installation instructions).
  - Remove three out-of-date entries for VisualAge TeamConnection documentation from the `Start->Program->TeamConnection` menu. I used the Windows Explorer to find the `All User->Start Menu->Programs->TeamConnection` directory.
  - Delete all of the `.boo` files from the `TeamConnection\nls\doc\ENU` directory, since the BookManager versions of the documentation are out-of-date.
- Look at files that grow to clean up where necessary
  - Save and purge the `Event Viewer`
  - Clean out the `Recycle Bin`
  - Clean out the `TEMP` directory.
- Reconfigure the `Windows NT Server FTP server` to allow user accounts to ftp into the workstation. I also restricted the use of anonymous ftp.

Family Administrator's tasks

- Update start, stop and backup utilities.
- Add user exit that sends mail to new users.
- Review files (audit.log, teamc.log and files in `queue` directory) that grow and decide whether or not to prune any at this time.
- Add several new users.
- Create new `Extractor` authority group.
- Create new release and components as shown for gathering tools described in technical reports and class materials. Once again, the zipbuild builder is all that we need. New users that are not on our development team get `Extractor` access.





Build Administrator's tasks

My team shares responsibility for creating drivers and driver members, committing releases, etc. Also, the `zipbuild` builder is still the only builder we need, and no updates have been required recently.

Notes:

1. I always use the on-line version of the documentation in Adobe Acrobat format. It is much more current and accurate than the printed documentation.
2. Even though I perform most roles, I am fortunate in that anyone in my department can, and does, provide backup.

Automated shutdown and backup

As I stated in the first **Evolution of a new TeamConnection family...**, the automation of startup and shutdown, including a reboot, is useful. The simple examples I gave in that document were sufficient for a family with few users, but I quickly needed more. Here are the new tools I use, and how I invoke them.

Nightly tasks:

1. shutdown family
2. shutdown database
3. copy database (.tcd file)
4. restart database
5. restart family

Weekly tasks:

1. make copy of all VisualAge TeamConnection files
2. send copy to another system (vital records backup)
3. reboot
4. restart family

The backup is run from `tcstop.cmd` (see "tcstop.cmd" on page 8). When a full backup is required, tcstop calls `tcfull.cmd` (see "tcfull.cmd" on page 9). We ran `tcsched.cmd` (see "tcsched.cmd" on page 6) once to schedule all backups.

Since we are using Windows NT, we had two choices for handling reboot:

- We could put our server in a secured room, then configure the machine to automatically log onto a user account that runs `tcstart.cmd` (see "tcstart.cmd" on page 7) from the Startup folder as part of the login process.
- We could leave the server in the open and ask someone to log in manually after an automated reboot.

We decided it would be fine for the first person that came in Monday morning to log in and let the family restart. The server sends a note to several users reminding them to check the server and restart the family.

Environment Variables

VisualAge TeamConnection sets environment variables required by the family in the System environment list. This is how the environment variables already created by VisualAge TeamConnection, and used by the automation scripts, are set:

```
TC_FAMILY=dept_cfj
TC_DBPATH=e:\dept_cfj\dept_cfj
```

These are the new environment variables added for the automation tools:

TC_BACKSITE This is the name of the target system that receives a vital records backup of our family once a week.

TC_BACKUSER This is the user on the backup target system.

TC_BACKPASS This is the password for that backup target system. Although, since this is a security exposure, this user should have limited usage.

TC_STRING This is the complete parameter string used by the `teamcd` command. It starts all of the VisualAge TeamConnection processes at once.

Here is how we set these variables set on our production family. The system is fully backed up regularly, with the backup tape taken to another location for storage.

```
TC_BACKSITE=perlovnt
TC_BACKUSER=anonymous
TC_BACKPASS=dept_cfj@mal4.raleigh.ibm.com
TC_STRING=-a agntf.fil -p procf.fil -n tcnotes.exe %TC_FAMILY%@9000 3
```

Contents of `agntf.fil`:

```
-s @mal4@9002 -p bldsock -e NT
```

Contents of `procf.fil`:

```
-s @9002
```

tcsched.cmd

We schedule our tasks using the NT `at` job. `at` is a Windows NT command that schedules processes to be started at a later time.

Note: When using `at` jobs, it is important to check the Windows NT Event Viewer frequently to make sure that the processes are running properly.

```
@echo off
echo Schedule TeamConnection Tasks
echo Nightly incremental backup...
at \\%COMPUTERNAME% 23:30 /every:M,T,W,Th,F "tcstop" INCR
echo Nightly update of defect/feature age
at \\%COMPUTERNAME% 23:00 /every:M,T,W,Th,F "age"
echo Every Sunday, full backup and reboot...
at \\%COMPUTERNAME% 23:50 /every:Su "tcstop" FULL
echo Notify of reboot
at \\%COMPUTERNAME% 23:00 /every:Su "tcnotes"
 %TC_DBPATH%\reboot.notice %TC_FAMILY%
 mlincoln@us.ibm.com perlovl@us.ibm.com
```

This is the text of the message in reboot.notice:

```
Subject:  dept_cfj family preparing for server reboot

Hi,
As part of a full backup of my family I am about to
reboot.  Please log onto mal4 so that my startup
processes can run and the dept_cfj TeamConnection
family can restart.

Thanks,
mal4.raleigh.ibm.com
```

tcstart.cmd

```
@echo off
REM Start TeamConnection Daemons:
REM - Change to working directory
REM - Start teamcd with other processes
REM - Check environment variables
REM
SET OUTFILE=tcstart
REM
echo Checking TC_DBPATH ...
if %TC_DBPATH%. == . goto enverr
echo Checking TC_FAMILY ...
if %TC_FAMILY%. == . goto enverr
echo Checking TC_STRING ...
if "%TC_STRING%" == "" goto enverr
echo Changing directory ...
cd /d %TC_DBPATH%
REM
REM Starting Family Daemons
if exist %OUTFILE%.bak erase %OUTFILE%.bak
if exist %OUTFILE%.out ren %OUTFILE%.out %OUTFILE%.bak
echo Verifying family database ...
osverifydb %TC_FAMILY%.tcd > %OUTFILE%.out 2>&1
echo Starting TeamConnection family ...
echo on
teamcd %TC_STRING% >> %OUTFILE%.out 2>&1
goto end
REM
REM Error Path
:enverr
echo ERROR, Environment variables or files not set properly ...
echo        Not starting TeamConnection.
echo USAGE: tcstart
:end
```

tcstop.cmd

```
@echo off
REM Stop TeamConnection
REM - Stop TeamConnection processes
REM - Stop ObjectStore database
REM - Run incremental or full backup
REM - Check environment variables
REM
SET OUTFILE=tcstop
REM
echo Checking TC_DBPATH ...
if %TC_DBPATH%. == . goto enverr
echo Checking TC_FAMILY ...
if %TC_FAMILY%. == . goto enverr
echo Changing directory ...
cd /d %TC_DBPATH%
if %1. == FULL. goto shut
if %1. == INCR. goto shut
echo ERROR, Must specify FULL or INCR
echo USAGE: tcstop FULL or tcstop INCR
echo        FULL backup includes all copying all files to another system
echo        INCR copies family database file to a backup
goto end
REM
:shut
echo Shutting down family %TC_FAMILY% and ObjectStore
echo Kill TeamConnection daemons ...
kill -f teamcd
kill -f teamagnt
kill -f teamproc
kill -f notifyd
REM
echo Manually stopping ObjectStore
ossvrchkpt %COMPUTERNAME%
net stop "ObjectStore Server R4.0"
net stop "ObjectStore Cache Manager R4.0"
REM
if %1. == INCR. goto incr
REM Full Processing
echo Executing full backup procedure ...
call tcfull
echo Rebooting ...
echo Log in to execute tcstart.
reboot
goto end
:incr
REM
REM Incremental Processing
echo Executing incremental processing ...
echo Copying database to backup file
if exist %TC_FAMILY%.bak erase %TC_FAMILY%.bak
if exist %TC_FAMILY%.tcd copy %TC_FAMILY%.tcd %TC_FAMILY%.bak
echo Manually restarting ObjectStore
net start "ObjectStore Server R4.0"
net start "ObjectStore Cache Manager R4.0"
call tcstart
```

```
REM
:end
```

tcfull.cmd

```
@echo off
REM FullBackup:
REM - Zip all files
REM - Vital Records Backup to remote location
REM
REM
echo Checking environment ...
echo Checking TC_FAMILY
if %TC_FAMILY%. == . goto enverr
REM Full backup of family directory (current working directory)
echo Backing up family (%TC_FAMILY%) in current directory:
CD
REM
echo Checking for TC_BACKSITE
if %TC_BACKSITE%. == . goto enverr
echo Checking for TC_BACKUSER
if %TC_BACKUSER%. == . goto enverr
echo Checking for TC_BACKPASS
if %TC_BACKPASS%. == . goto enverr
echo Checking for TC_DBPATH
if %TC_DBPATH%. == . goto enverr
REM
REM
echo Changing to directory of TC_DBPATH: %TC_DBPATH%
cd /d %TC_DBPATH%
REM
echo Backup of family %TC_FAMILY% > header.txt
date <nul: >> header.txt
zip -r -z %TC_FAMILY%.zip *.* < header.txt
erase header.txt
REM
REM Vital Records Backup
echo Performing vital records backup to %TC_BACKSITE%
echo open %TC_BACKSITE% >> response.ftp
echo user %TC_BACKUSER% %TC_BACKPASS% >> response.ftp
echo cd %TC_FAMILY% >> response.ftp
echo binary >> response.ftp
echo erase %TC_FAMILY%.bak >> response.ftp
echo ren %TC_FAMILY%.zip %TC_FAMILY%.bak >> response.ftp
echo put %TC_FAMILY%.zip >> response.ftp
echo bye >> response.ftp
ftp -n < response.ftp
erase response.ftp
rem erase %TC_FAMILY%.zip
echo Full backup of %TC_FAMILY% complete.
goto end
REM
:enverr
echo An environment variable is not correctly set for tcfull.cmd.
echo Quitting.
:end
```

Sharing materials with new users

Extractor Access Role

New users that only need to extract our educational materials are best served by a new authority group, `Extractor`. These users will be opening defects and features, adding remarks to these defects and features, and extracting output parts. All that I needed to do was:

1. start `tcadmin`
2. select the authority groups page
3. select the `general` authority group
4. change the group name from `general` to `extractor` and press the `new` button
5. then add the following actions for extractors.

Action	**TeamConnection command**
PartExtract	teamc part -extract
DriverExtract	teamc driver -extract
ReleaseExtract	teamc release -extract

UserCreate user exit

A user exit running on the UserCreate action that sends a note to new users helps make them productive more quickly. It also allows us to verify the user's e-mail address. To maintain flexibility, the user exit reads the bulk of the mail message from a `motd` (message of the day) file.

This is the update to `config\userExit` to invoke the user exit:

```
UserCreate 2 newuser.exe "" ENV=(login,usersfullname,sendmailaddress)
```

Below is the mail message that is sent to each new user. It is customized in the user exit program, `newuser.exe`, to provide information about their new user account.

```
Subject:  Welcome New TeamConnection User to %TC_FAMILY%
Message of the Day for dept_cfj family, Updated 2/4/98

Family: dept_cfj@mal4.raleigh.ibm.com@9000

There is a builder running on mal4:
   Port         bldsock@9006
   Environment: NT

For information on education package development, extract
README.txt from release class1_all.  Request access to component
training_class from perlovl@us.ibm.com

Other components and releases:
- Miscellaneous tools: (currently only Adobe Acrobat)
     Component: tools
     Release:   tools
- Other information will be released as it becomes available
```

Please send a note to perlovl@us.ibm.com to have an initial password or hostname set (if you have not already done so).

newuser.exe

```
/*
*********************************************************************

SAMPLE NAME:  newuser.c

USAGE:        newuser

COMPILATION:  icc -o newuser newuser.c

ENVIRONMENT VARIABLES:
                      TC_FAMILY
                      TC_DBPATH

DESCRIPTION:  This user exit is designed to be registered for UserCreate, exit 2.
              Here is the recommended entry in the config/UserExit file
                 UserCreate 2 newusers "" ENV=(component,release) # New User Mail Message
              This program extracts the current contents of %TC_DBPATH%/motd, adds a
              standardized subject line and some framing, then invokes tcnotes.exe to
              mail the document using Lotus Notes.

NOTES:        1. The tcnotes.exe utility is required as the mail exit, and notifyd must
                 be running.
              2. The functions envInitReadEnvFile and envGetFromEnvFile are derived
                 from the TEAMCENV.C sample provided with TeamConnection.
              3. The function userHelp is derived from the help function in VIEWEXIT.C

*********************************************************************
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* This is based on a limit used for actions in TeamC */
#define maxParmName 40

/* Function Declarations */
void userHelp(void);
char *getEnvVal(char *inValue);
FILE *initFile(char *envFileName, char *parms);
int envGetFromEnvFile(FILE *envFile, char *parameterName, char *outValue);

/*-------------------------------------------------------------------------\
|  userHelp:                                                               |
\-------------------------------------------------------------------------*/
void userHelp(void)
{
   fprintf(stderr,"newuser usage:\n");
   fprintf(stderr,"\tnewuser                                -  This help message\n");
   fprintf(stderr,"\tnewuser UserCreate_ParameterList ...  -  Send motd to new user\n");
   return;
}

/*-------------------------------------------------------------------------\
|  getEnvVal:                                                              |
\-------------------------------------------------------------------------*/
char *getEnvVal(char *inValue)
{
   char *outValue;

   outValue = getenv(inValue);
   if (outValue == NULL)
   {
      fprintf(stderr, "newuser: Error, %s environment variable must be set\n",
         inValue);
      return NULL;
   }
```

```
   return outValue;
}

/*-------------------------------------------------------------------------\
|  initFile:                                                               |
|  - Return File Handle                                                    |
|  - temporary file automatically deleted when daemons killed              |
\-------------------------------------------------------------------------*/
FILE *initFile(char *envFileName, char *parms)
{
   FILE *envFile;
   /* Open temporary file */
   envFile = fopen(envFileName, parms);
   if (envFile == (FILE *)0)
   {
       fprintf(stderr,"newuser: Error, could not open file \"%s\"\n",
          envFileName);
       return (FILE *)0;
   }

   return (envFile);
}

/*-------------------------------------------------------------------------\
|  envGetFromEnvFile:                                                      |
|  - Write an entry to environment file                                    |
|  - Write binary:                                                         |
|    size of parameter, parameter string, size of value, value string      |
\-------------------------------------------------------------------------*/
int envGetFromEnvFile(FILE *envFile, char *inName, char *outValue)
{
   int nNameLength;
   int nValueLength;
   char parameterName(maxParmName+1“; /* allow for maximum in TeamC (15 + NULL) */
   char parameterValue(16001); /* allow for max in TeamC 16000 for remarks + NULL */

   /* Search for parameter identified by inName */
   if (*inName == '\0')
   {
      fprintf(stderr,"newuser: Error, require parameter name\n");
      return 1;
   }
   else /* Searching for one entry */
   {
      fread(&nNameLength, sizeof(int), 1, envFile);
      fread(parameterName, sizeof(char), nNameLength, envFile);
      *(parameterName+nNameLength)='\0';
      fread(&nValueLength, sizeof(int), 1, envFile);
      fread(parameterValue, sizeof(char), nValueLength, envFile);
      *(parameterValue+nValueLength)='\0';

      while ((strcmp(inName, parameterName) != 0) && (!feof(envFile)))
      {
         fread(&nNameLength, sizeof(int), 1, envFile);
         fread(parameterName, sizeof(char), nNameLength, envFile);
         *(parameterName+nNameLength)='\0';
         fread(&nValueLength, sizeof(int), 1, envFile);
         fread(parameterValue, sizeof(char), nValueLength, envFile);
         *(parameterValue+nValueLength)='\0';
      }
      if (!feof(envFile))
      {
         /* Return the value of the parameter */
         strcpy(outValue, parameterValue);
      }
      else
      {
         strcpy(outValue, "");
         fprintf(stderr, "newuser: Error, parameter \"%s\" not found\n", inName);
         return 1;
      }
   }
```

```
   return 0;
}

/*-------------------------------------------------------------------------\
|  main:                                                                   |
|  - Get parameters from envFile for UserCreate User Exit 2 action         |
|  - Read %TC_DBPATH%/motd and create a new user mail message              |
|  - Use Lotus Notes mail exit to mail new user message                    |
\-------------------------------------------------------------------------*/
int main(int argc, char *argv())
{
   FILE *envFile, *userFile;
   char *family, *dbpath, *userFileName;
   char envLogin(31), envFullName(63), envMailAddr(159);
   char command(1000);
   int  rc = 0;

   /* If no parameters, print help text */
   if (argc < 3)
   {
      userHelp();
      exit (1);
   }

   /* Check environment variables */
   family       = getEnvVal("TC_FAMILY");
   dbpath       = getEnvVal("TC_DBPATH");
   if (family == NULL || dbpath == NULL)
   {
      fprintf(stderr, "newuser: Error, Could not get one or more environment variable values\n");
      exit (1);
   }
   /* Parameter 1, UEParameters string, is not currently used               */

   /* Parameter 2, envFile, must be set                                     */
   /* Access parameter env file.                                            */
   if (strlen(argv(2)) == 0)
   {
      fprintf(stderr, "newuser: Error, envFile must be specified in config/UserExit\n");
      exit (1);
   }
   envFile = initFile(argv(2), "rb");
   if (envFile == (FILE *)0)
   {
      exit (1);
   }
   /* Get parameters needed for message from envFile */
   rc =  envGetFromEnvFile(envFile, "login", envLogin); /* Login ID */
   rc += envGetFromEnvFile(envFile, "usersfullname", envFullName); /* Full Name */
   rc += envGetFromEnvFile(envFile, "sendmailaddress", envMailAddr); /* mail address */
   if (rc > 0)
   {
      fprintf(stderr, "newuser: Error, Could not get one or more parameters from \"%s\"\n",
         argv(2));
      exit (1);
   }

   /* Construct mail message */
   /* - Create temporary file */
   /* - Write subject line */
   /* - append motd */
   userFileName = tmpnam(NULL);
   userFile = initFile(userFileName, "wb");
   if (userFile == (FILE *)0)
   {
      exit (1);
   }

   fprintf(userFile, "Subject:  Welcome New TeamConnection User to %s\n", family);
   fprintf(userFile, "Welcome %s, your new userid is %s\n", envFullName, envLogin);
   fprintf(userFile, "-------------------------------------------------\n");
```

```
   fclose(userFile);

   sprintf (command, "type %s\\motd >> %s", dbpath, userFileName);
   rc = system(command);
   if (rc > 0)
   {
      fprintf(stderr, "newuser: Error, Could not execute command: %s\n", command);
      exit (1);
   }

   /* Send file to user */
   sprintf(command, "tcnotes.exe %s %s %s", userFileName, family, envMailAddr);
   rc = system(command);
   remove (userFileName);
   if (rc > 0)
   {
      fprintf(stderr, "newuser: Error, Could not send new user note\n\
command: tcnotes %s %s %s\n", userFileName, family, envMailAddr);
      exit (1);
   }

   return (0);
}

/* End of File */
```

Future enhancements

VisualAge TeamConnection and our family are both still growing and evolving. We always keep our eyes open for new ideas or features in VisualAge TeamConnection that can help us do our jobs better.

Here are some manual tasks that I perform periodically that should be automated:

- Cleaning out event logs and system files that grow.
- Archiving and cleaning out family files that grow.
- Logging in after a reboot.

Here are some other tasks that would increase the usability of our family:

- Notification of events via e-mail and paging.
- Failure monitoring and notification.
- System monitoring and notification.

There are still some things we haven't decided, even after five months. These are the ones we resolved:

- Question: What should our driver naming convention be and how should we open our drivers?

  Answer: We guess at an end date, then create a new driver as soon as we commit the last driver. Since committing a few workareas at a time is faster than doing lots of workareas, we create new drivers about once a week. When we are near the end of a development cycle, we can create as many as two in a day.

- Question: Why not move our family to a Unix server, where we already have an administrator assigned?

  Answer: We don't have any other production families running on Windows NT in our lab. Also, every one of us that takes a laptop computer to customer sites is running Windows NT. We really needed to gain long-term experience managing a VisualAge TeamConnection family on Windows NT.

These are the questions we are STILL pondering:

- Question: The port number used by `dept_cfj` is too common, should we change it?

  Guess: The currently used port number is 9000. This is a very common number, and there is a real risk that someone will put an application on our server or a client machine that uses that port number. We still may change the number.

- Question: Do we have a need for build processors running on other operating systems?

  Guess: We are moving our technical reports to this family. These documents are still written in a scripting language. We will probably want to add a builder on a system, perhaps MVS, that runs a scripting tool. We may also add a packaging builder that uses `tar` on AIX.

Finally, I hope that sharing our experiences will help you better utilize VisualAge TeamConnection.

Appendix A. Bibliography

VisualAge TeamConnection Publications

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

SC34-4551 TeamConnection, Administrator's Guide
SC34-4552 Getting Started with the TeamConnection Clients
SC34-4499 TeamConnection, User's Guide
SC34-4501 TeamConnection, Commands Reference
SC34-4500 TeamConnection, Quick Commands Reference

Related Redbooks

The following IBM redbooks provide practical advice about VisualAge TeamConnection from software specialists:

SG24-4648 Introduction to the IBM Application Development Team Suite
SG26-2008 TeamConnection Family and Application Development
SG24-4610 TeamConnection Workframe Integration Survival Guide

Related Technical Reports

The following technical reports provide hints for using VisualAge TeamConnection:

29.2333 Evolution of a new TeamConnection Family: Common do's and don'ts
29.2267 TeamConnection frequently asked questions: How to do routine operating system tasks
29.2253 Comparison between CMVC 2.3 and TeamConnection 2

Appendix B. Copyrights, Trademarks and Service marks

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

TRADEMARK, REGISTERED TRADEMARK OR SERVICE MARK	COMPANY
AIX, OS/2, IBM, CMVC, VisualAge TeamConnection	IBM Corporation
ObjectStore	Object Design, Inc.
UNIX, USL	UNIX System Laboratories, Inc.
Microsoft, Windows NT	Microsoft Corporation

END OF DOCUMENT °