Comparison between CMVC 2.3 and TeamConnection 2

Document Number TR 29.2253

Angel Rivera, Lee Perlov and Sam Ruby

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

ABSTRACT

This technical report compares the functions of Configuration Management and Version Control (CMVC) and its successor product, TeamConnection. The objective is to provide to the CMVC user the relevant information about what is the same, what is different and what is new between CMVC and TeamConnection.

ITIRC KEYWORDS

CMVC

TeamConnection

First revision, May 1997

Added the following sections: “The Big Differences” on page 2, “New UNIX GUI” on page 27, and “Space used by family daemons” on page 36.
Updated the following section: “Space used by database” on page 36
Added references to other TeamConnection technical reports.

This document has now been reviewed and updated by the TeamConnection lead architect, Sam Ruby. As such, his name has been added to the author's list.

ABOUT THE AUTHORS

ANGEL RIVERA

Mr. Rivera is an Advisory Software Engineer and team lead for CMVC development. He joined IBM in 1989 and since then has worked since then 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 a B.S. in Electronic Systems Engineering from the Instituto Tecnológico y de Estudios Superiores de Monterrey, México.

LEE R. PERLOV

Mr. Perlov is a Staff Software Engineer in the 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 moved to RTP to work on library development and support, in 1992.

Mr. Perlov received a B.S. 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.

SAMUEL RUBY

Mr. Ruby is a Senior Software Engineer and the lead architect of TeamConnection. He was previously lead architect of the mainframe library product, SCLM.

His turn-ons include Diet Coke in the can and Dilbert.

CONTENTS

ABSTRACT . . . iii
ITIRC KEYWORDS . . . iii

ABOUT THE AUTHORS . . . vii
Angel Rivera . . . vii
Lee R. Perlov . . . vii
Samuel Ruby . . . vii

Figures . . . xi

Introduction . . . 1
Compatibility . . . 1
Migration from CMVC to TeamConnection . . . 2
The Big Differences . . . 2

Comparison of common functions . . . 3
Command names . . . 3
Configuration management . . . 4
Components and component hierarchy . . . 4
Access to components . . . 4
Notification mechanism . . . 5
Customized processes for components . . . 6
Release management . . . 6
Releases . . . 6
Approval List . . . 7
Environment List . . . 7
Customized processes for releases . . . 8
Change Control . . . 8
Overview . . . 9
Defects . . . 9
Features . . . 9
Version Control . . . 10
Files (CMVC) or Parts (TeamConnection) . . . 10
Tracks (CMVC) or Workareas (TeamConnection) . . . 11
Levels (CMVC) or Drivers (TeamConnection) . . . 12
Users and authentication of users . . . 12
Differences . . . 12
Architectural Issues . . . 12

Client/Server Design . . . 13
Customization aspects . . . 13
Integration with other products . . . 14
Query facility . . . 14
Migration utilities . . . 15
Year 2000 Compliance . . . 15
Supported platforms and databases . . . 15
Family server . . . 15
Clients . . . 16
Databases . . . 16
License handling . . . 16
Family administration . . . 17
Structure of a family account . . . 17
Suggestions . . . 18
Family administration tools . . . 19

New user functions of TeamConnection . . . 21
Sequential/Concurrent development . . . 21
Automatic pruning of releases . . . 21
Versioning . . . 22
Versioning in CMVC . . . 22
Versioning in TeamConnection . . . 23
Finding different versions of TeamConnection objects . . . 23
Parts are more than just Files . . . 24
File/Part links . . . 24
Changing the type of a file (text <-> binary) . . . 24
Workareas are more than renamed Tracks . . . 25
Multiple workareas per defect or feature . . . 25
Workarea performance . . . 25
teamc workarea -undo . . . 25
Driver has more options than Level . . . 26
teamc driver -freeze . . . 26
teamc driver -restrict . . . 26
Updated Intel GUI . . . 27
New UNIX GUI . . . 27
Authentication by means of login . . . 28
Planning for teamc release/driver -extract actions . . . 29
Build support . . . 29
Packaging support . . . 30
Object Repository and Information Model . . . 30

Changes that impact system administrators . . . 31
TCADMIN - System Administrator's GUI . . . 31

Other administrator tools . . . 31
User exits . . . 32
License handling . . . 32
License handling in CMVC . . . 32
License handling in TeamConnection . . . 33
Backup and Recovery . . . 33
What processes are started . . . 33
What processes are started in CMVC . . . 33
What processes are started in TeamConnection . . . 34
Use of disk space and memory . . . 35
Directory /tmp . . . 35
Disk space for the family account . . . 36
Use of AFS and NFS . . . 36
Memory usage . . . 37
Family daemons and security/integration issues . . . 37
Process privileges . . . 37
Access to data in the family account . . . 38
System integration issues . . . 38
Other issues . . . 38

Appendix A. Bibliography . . . 39
CMVC 2.3 . . . 39
TeamConnection 2 . . . 39
TeamConnection 1, useful to TeamConnection 2 users . . . 39
Related Technical Reports . . . 39

Appendix B. Copyrights, Trademarks and Service marks . . . 41

FIGURES

1. CMVC terms that have different name/function in TeamConnection . . . 3
2. CMVC family directory structure . . . 17
3. TeamConnection family directory structure . . . 18
4. Changes from CMVC to TeamConnection . . . 18
5. CMVC daemon process hierarchy . . . 34
6. TeamConnection daemon process hierarchy . . . 35

INTRODUCTION

The purpose of this technical report (TR) is to compare and contrast the product Configuration Management and Version Control (CMVC) Version 2.3 with its successor TeamConnection Version 2.0. The focus will be on enhanced capabilities in TeamConnection derived from the addition of an object-oriented methodology and object repository.

This TR is designed to provide enough information for current CMVC users to quickly become productive TeamConnection users. It is assumed that readers are familiar with the basic functions of CMVC.

The organization of this TR is as follows:

- Comparison of functions common to both products
- New functions affecting general TeamConnection users
- New functions and differences impacting family administrators

This technical report is not a replacement for the documentation of these products; therefore, the functions will be explained briefly in this TR. For more details, refer to the appropriate documentation listed in the bibliography, Appendix A, “Bibliography” on page 39.

The best way to find out how the client functions really work, is to run/review the following samples:

- In CMVC, /usr/lpp/cmvc/samples/README.demo2 and demo2.tar.
- In TeamConnection, /usr/teamc/samples/univunix. The univunix utility runs the TeamConnection commands in univunix.cmds which you can modify.

COMPATIBILITY

TeamConnection is the successor product to CMVC. TeamConnection has evolved and been extended to support object-oriented development. Both products have a lot in common; however, because of changes in TeamConnection these products are not compatible with each other. That is, you cannot use a CMVC client with a TeamConnection server or a TeamConnection client with a CMVC server.

A migration facility is provided with TeamConnection to move your CMVC family to TeamConnection. There are no utilities to migrate a TeamConnection family to CMVC.

Migration from CMVC to TeamConnection

The technical report "How to do migration tasks with CMVC", TR 29.2232, describes (in Chapter 6 "Preparing to migrate to TeamConnection") how to prepare a CMVC family to be migrated to TeamConnection.

Follow the instructions in the TeamConnection Administrator's Guide (Chapter 16 "Migrating to TeamConnection version 2") to perform the actual migration from a CMVC family to TeamConnection.

THE BIG DIFFERENCES

The main differences between CMVC and TeamConnection that CMVC users will experience when migrating to TeamConnection are shown below. There is more detail in the sections “Comparison of common functions” on page 3 and “New user functions of TeamConnection” on page 21.

The names of several objects have changed. For example, tracks are now workareas and levels are now drivers. For more details see “Command names” on page 3.

Workareas are much more important than tracks and behave differently. For more details see “Workareas are more than renamed Tracks” on page 25

- A workarea needs to be specified in all teamc part -checkin and -checkout commands. Even if you are just using TeamConnection to store parts in a single release without any defects or features, you need to create a workarea and periodically integrate and commit your changes.
- You will periodically be required to refresh your workarea before you perform actions like integrating the workarea, or checking parts in or out. For example, if you try to check out a file that another user has updated in a different workarea, but has not being committed, you will need to refresh from that workarea.

The versioning scheme is different. Version identifiers are now related to the workarea, and are not global. Furthermore, workareas, drivers and releases are versioned. For more details see “Versioning” on page 22

When performing queries, like filling in a filter window, you are often querying within a context instead of using the entire family. Therfore, you need to specify that context by entering a release, a driver or a workarea. If you do not provide enough context you will get an error or not be able to press the OK button. For more details see “Query facility” on page 14.

COMPARISON OF COMMON FUNCTIONS

COMMAND NAMES

When comparing TeamConnection and CMVC, the most obvious change is the naming of the client commands. All TeamConnection commands now begin with "teamc", for example, "teamc report". Most of the CMVC and TeamConnection commands are the same. However, the table below lists commands that have changed names:

Figure 1. CMVC terms that have different name/function in TeamConnection

CMVC	TeamConnection	Comment
File	Part	To better describe the range in granularity of the objects that can be managed.
Level	Driver	To conform with common industry terms
LevelMember	DriverMember	To conform with the change for Level
Track	Workarea	To conform with the functionality change in workarea.
cmvcd	teamcd	To reflect the name of the product
cmvc	teamcgui	To reflect the name of the product
stopCMVC	stopteamc	To reflect the name of the product

Notes:

1. The actions in the Authority table reflect the corresponding names, such as FileView in CMVC and PartView in TeamConnection.
2. In Windows and OS/2, where commands are limited to 8 characters some of the CMVC names have been abbreviated.
3. Some versions of the OS/2 client contain compatibility commands (for example, Report.exe). These will be removed at a later date.

CONFIGURATION MANAGEMENT

Both CMVC and TeamConnection provide support for the process of identifying, organizing, managing and controlling software modules. This is accomplished by components and a component hierarchy.

Components and component hierarchy

A component hierarchy allows software modules to be grouped for organization purposes and to control the access of users to those software modules. For example, an application can create components to separately manage access to server code, client code and documentation.

A component can have zero, one or more child components, as well as one or more parents. The top component of the hierarchy, "root", has no parent; consequently, all the rest of the active components are descendants of the "root" component. However, deleted components also do not have parents.

The access list and the notification lists are inherited from the parent component to its descendants.

Access to components

An access list associated with a component specifies an authorization level for each user, which determines the action that the user can perform on that component and its descendants. For example, a user with a "general" access has the CompView authority that allows the user to view the description of the specified component (or its descendants) but does not have the PartAdd authority that is needed to create a software module.

The above authorities can be restricted in a specific component. However, child components do not inherit these restrictions.

There are certain actions that all users are able to perform, regardless of their access list. These actions are called "base authorities", such as opening a defect and adding remarks to it.

There are other actions that each user has implicit authority for, such as modifying the properties for the user object in CMVC or in TeamConnection.

The users who have "superuser" authority are allowed to perform any action that is legal, given the current process configurations and the state of the TeamConnection object being

manipulated. For example, a superuser can cancel any defect that is not already associated with a workarea.

Differences

Some actions from the CMVC authority.ld have been renamed in the TeamConnection authorit.ld file; which is loaded into the database as the Authority table. For example, the FileView action has been renamed to PartView

Furthermore, some new actions have been added to TeamConnection to reflect the new functions.

Notification mechanism

A notification mechanism sends mail to team members as software modules change and as other actions are taken with objects in the product.

There are some explicit notifications that the user may want to receive and this is accomplished by registering to a notification list in a given component. For example, if a user wants to be notified when a new release is created in that component, then the user can add the "developer" notification list.

There is no support for finer granularity of events, such as the notification for all changes for one particular defect.

The actual notification is performed by means of the Notification Daemon (notifyd). This daemon runs a script which can be customized, for example, to use a mechanism other than sendmail.

Differences

Some actions from the CMVC interest.ld have been renamed in the TeamConnection interest.ld file; this file is loaded into the database to populate the Interest table. For example, the FileView action has been renamed to PartView.

Furthermore, some new actions have been added to TeamConnection to reflect the new functions.

Customized processes for components

The components are the objects from which defects and features (see “Defects” on page 9 and “Features” on page 9) are opened and manipulated. The process to handle the defects and features is customizable by component; that is, one component can have one process, and another component can have another process.

The process for a component could include the following subprocesses:

Design-Size-Review (DSR)
Verify subprocess

Depending on the subprocesses, it is possible to have the following records:

sizing records to identify the sizing activities during design

verification records to accept or reject the defect/feature.

RELEASE MANAGEMENT

Both CMVC and TeamConnection provide support performing a logical organization of objects that are related to an application. This is accomplished by releases.

Releases

A release provides a logical view of objects that must be extracted, built, tested and distributed together. Furthermore, a release can be linked to another release.

Both CMVC and TeamConnection provide support for handling serial development; that is, only one user at a time can have a lock on a file.

The release may have an approval list and environment list.

Differences

TeamConnection has added the following functionality to releases:

Concurrent development.

A release can be created to allow concurrent development, that is, more than one developer at a time can update a part.

For more details see “Sequential/Concurrent development” on page 21.

Collision records.

When using concurrent development, if two or more users are updating the same part, then a collision record is created upon the first checkin of that part (that is, the first one in wins, the second gets a collision record). The workarea receiving the collision record needs to resolve the differences prior to integration. We suggest using the TCMERGE tool, available on Windows and OS/2.

For more details see “Sequential/Concurrent development” on page 21.

TCMERGE: Graphical tool to merge different versions of a part.

To help the handling of collision records, TeamConnection provides a GUI utility called TCMERGE that shows the differences between two parts and allows the user to manually merge the parts into one part.

Currently, this tool is not available in UNIX.

Versioning releases.

In TeamConnection the releases can be versioned in the same manner as other versionable objects in TeamConnection.

For more details see “Versioning” on page 22.

Building and packaging releases.

By using the new build and packaging functions in TeamConnection, the releases can be built and packaged from within TeamConnection, without the need to extract the release and invoke the make utilities to build the code.

For more details see “Build support” on page 29.

Pruning.

A release can be automatically pruned under certain conditions.

For more details see “Automatic pruning of releases” on page 21.

Approval List

The approval list contains the users who can approve the changes to be made in a release.

Environment List

The environment list contains the platforms or environments that need to be tested when the changes are incorporated into the release.

Customized processes for releases

The process to handle the releases is customizable, that is, one release can have one process, and another release can have another process.

The process for a release could include the following subprocesses:

Process Purpose

Track Require defects or features to be associated with all tracks (CMVC) or workareas (TeamConnection). In other words, you need a reason to make a change.

Approval Require approval for each track/workarea before changes can be made to the track/workarea. This is usually used to limit development activity as you approach delivery of the release.

Fix Used to identify components where files/parts are changed. Completing a fix record is a useful mechanism for developers to indicate that their track/workarea is ready to be integrated.

Level (CMVC) or Driver (TeamConnection) Levels/Drivers are used to incrementally integrate and commit a set of changes in a release. This makes incremental development and the delivery of beta drivers easy to manage and recreate.

Test Used as a verification method for someone other than the originator of a defect or feature. Useful when a test group or some other users need to evaluate a change prior to the originator taking final action.

Depending on the subprocesses being used, the following records may be used:

approval records which are used by those users that are in the approval list.

fix records which are used by those users who are responsible for the components where the changes will be incorporated.

test records which are used by those users who need to test the changes in the specified environments.

Differences

The CMVC Level subprocess is called Driver in TeamConnection.

CHANGE CONTROL

Overview

Both CMVC and TeamConnection keep track of any file changes you make and the reasons you make them. After changes are made, you integrate the changes and build the application.

Note: It is not necessary to make a change prior to integrating tracks (CMVC) or workareas (TeamConnection). An empty track/workarea can be promoted for documentation or record keeping purposes.

Both products ensure that the change process is followed and that the changes are authorized. The change control process is configurable and your team can decide how strict the change control should be, from loose to very tight. You can also adjust the level of control as you move through a development cycle.

Defects and features are used to identify the requirements for the file changes.

There is an audit trail associated with each defect and feature. The history information includes who opened it, who accepted it, a description of the problem or suggestion, etc.

The defects and features can be "aged" in order keep track of how long the defect or feature has been active.

Differences

TeamConnection also adds a timestamp to each entry in the audit log for more complete tracking of family activity.

Defects

A defect is opened (but not created) to report a problem.

The process to handle the defect depends on the process configuration for the component where the defect was opened. See “Components and component hierarchy” on page 4.

Features

A feature is opened (but not created) to request a design change to a future function.

The process to handle the feature depends on the process configuration for the component where the feature was opened. See “Components and component hierarchy” on page 4.

VERSION CONTROL

Both CMVC and TeamConnection provide support for the process of tracking the relationships among the versions of the various software modules that form an application. Version control with configuration management enables you to build your product using stable levels of code, even if the code is constantly changing.

Version control allows you to view the different snapshots of a file over time: as the file was created, as it was one month ago, as it is today.

The file changes need to be done in the context of a release. These changes are done by means of CMVC Tracks or TeamConnection Workareas, which in turn are integrated into a CMVC Level or a TeamConnection Driver. These integrated changes will eventually be committed by following the release process.

Differences

The implementation of versioning has been completely replaced in TeamConnection. For more details see “Versioning” on page 22.

Files (CMVC) or Parts (TeamConnection)

The software modules for your application are stored in files in CMVC (and parts in TeamConnection). Files are then created and modified to address defects and features. These files are associated with a component for the authorization of who can do what, and with a release for the actions of extraction, subsequent build and packaging.

The files are versioned and both products handle the versioning of text (such as a file that contains source code in C) and binary files (such as an executable file).

Both CMVC and TeamConnection provide expandable keywords that can be embedded in the files. This information identifies the context of the build (for example, which release the file came from), and when the files are extracted, these expanded keywords provide useful information to developers trying to determine where a source file or executable came from.

Differences

TeamConnection parts are more than CMVC files. The granularity was expanded to include fine-grained objects that are not extracted as files, such as VisualAge Generator objects. Further, TeamConnection parts have attributes that are used during build and packaging. For more details see “Versioning” on page 22 and “Parts are more than just Files” on page 24.

In CMVC, there are 2 sets of expandable keywords, one if the family is configured to use SCCS (the most widely used type) and another if the family uses PVCS. TeamConnection provides a new and more flexible syntax for providing the same keyword capabilities as SCCS.

The TeamConnection Part command does not have a -destroy action. Parts can still be deleted from a release, but not physically removed.

The TeamConnection Part command adds actions to support the building of parts.

The TeamConnection Part object uses in the lastUpdate attribute the date and time from the file system of the file that is being checked in. Thus, during the extraction, the file will have the same date and time that the file had during the most recent check-in. In contrast, in CMVC, the lastUpdate attribute changed for ANY action that affected either the contents or the attributes of the file, and this date and time was used during the extraction.

Tracks (CMVC) or Workareas (TeamConnection)

In CMVC, depending on the process of the release, if the track subprocess is activated, then you need to specify a CMVC Track that will identify the file changes with a defect/feature and a release. For more details see “Workareas are more than renamed Tracks” on page 25.

You may specify that a given Track/Workarea needs to be integrated with another one, and thus, they will become corequisites.

Differences

TeamConnection workareas are more powerful and useful than CMVC tracks due to the TeamConnection's object-oriented design.

Workareas in TeamConnection are the context by which you view a release (a logical view). As such, you must specify always a workarea.

Further, you may open multiple workareas for a defect or feature in one release. For more details, see “Workareas are more than renamed Tracks” on page 25.

TeamConnection add the concept of a prerequisite so that two workareas do not need to change the same part in order to establish a relationship that will insure they are integrated to the same driver.

Levels (CMVC) or Drivers (TeamConnection)

In CMVC, depending on the process of the release, if the CMVC Level subprocess is activated, then you need to integrate the CMVC Track into a CMVC Level by means of a CMVC LevelMember.

Similarly, in TeamConnection, depending on the process of the release, if the TeamConnection Driver subprocess is activated, then you need to integrate the TeamConnection Workarea into a TeamConnection Driver by means of a TeamConnection DriverMember.

Differences

TeamConnection Drivers are also more powerful and useful than CMVC Levels. See “Driver has more options than Level” on page 26 for more details.

USERS AND AUTHENTICATION OF USERS

Each user has a unique ID in CMVC or TeamConnection. Be default, it is used in combination with a hostname and a system login to control access to the family.

In OS/2 and in Windows 3.1, because there is no system login, a variable is used instead; in CMVC the variable is USER and in TeamConnection the variable is TC_USER.

Differences

The functionality of the user is the same, but the authentication of users has been expanded in TeamConnection to use passwords and to require users to login into TeamConnection, and this makes optional the use of the hostname and system login as part of the authentication. For more details, see “Authentication by means of login” on page 28.

ARCHITECTURAL ISSUES

Client/Server Design

Both CMVC and TeamConnection have a client/server architecture, in that the GUI and client communicate across a TCP/IP socket to a family server. The socket is identified by "FamilyName@FamilyHost@SocketNumber".

The client consists of a command line interface and a graphical user interface (GUI). The GUI constructs and issues line commands. The line commands issued are stored in a log file (the location is set in the GUI's settings pages).

Differences

In TeamConnection, instead of having one executable file for each line command, such as with the CMVC "Report" command, a single executable teamc, contains all functions for specific command such as "teamc report". This approach allows to have the same name for UNIX and OS/2 commands. For example, in CMVC, the UNIX command "LevelMember" had to be renamed in OS/2 as "levelMem".

TeamConnection adds a second client/server interface, the Object Repository. This allows C++ programs to access data in the family server through an object-oriented application programming interface. See “Object Repository and Information Model” on page 30 for more details.

Customization aspects

The family can be customized in the following aspects:

Configuration types (config.ld)

The family administrator can add more new types for items that have multiple-choice values.

Configurable fields

It is possible to add extra fields to the following objects:

- Defects
- Features
- Users
- Files (CMVC) or Parts (TeamConnection)

For the family administrator there are several small changes. See “Family administration” on page 17 for more details.

User exits (only on the server side)

Both products allow you to extend their functionality to a certain degree by allowing the family server to invoke utilities developed by the family administrator when certain actions/conditions occur. For more details, see “User exits” on page 32.

Integration with other products

In CMVC and TeamConnection it is possible for other applications to use the command line interface to interact with the product.

Differences

In CMVC, a UNIX client GUI is provided that is integrated with the product Softbench. TeamConnection is not integrated to Softbench, but will integrate to other similar "framework applications" in the future.

In TeamConnection, API's are provided for other tools to invoke TeamConnection command line capabilities from a function library, and to access TeamConnection data through an Object Repository (see “Object Repository and Information Model” on page 30 Applications such as VisualAge Generator use both API's integrate to TeamConnection.

Query facility

Both CMVC and TeamConnection allow queries to the database that are based on SQL.

Differences

CMVC simply passes thru the SQL queries to the database management system (DBMS) of the relational database and the DBMS responds to the queries.

TeamConnection provides an SQL translation facility on top of an object-oriented database. Users should not see any differences in capabilities.

The TeamConnection report command now checks access lists and returns only the rows of data that a user is authorized to see.

CMVC and TeamConnection have basically the same common tables and views, with the exception of the new functions for TeamConnection (such as ParserView) and the renaming of some functions (such as Levels to Drivers).

The TeamConnection teamc report command offers two new options:

- -testClient verifies the configuration of the client without contacting the server.
- -userExitInfo provides information on user exit to superusers.

Migration utilities

CMVC and TeamConnection provide utilities to migrate to the most current versions of each product. Further, TeamConnection provides a utility to migrate from CMVC to TeamConnection 2.0. CMVC provides utilities to migrate from SCCS to CMVC.

Year 2000 Compliance

CMVC uses 2 digits to store the information about the year. Although CMVC does not parse or use this information directly, when the user requests sorting by date (the actual sorting is done by the database) it might be possible that the year 2000 might not be sorted properly.

TeamConnection uses 4 digits to store the information about the year. Thus, TeamConnection is fully compliant with the Year 2000 requirement.

SUPPORTED PLATFORMS AND DATABASES

Family server

Both products support family servers in the following platforms:

AIX Version 4
HP-UX Version 10

Note: The port to Solaris is being done at the time this TR is being prepared.

TeamConnection adds the following server platforms:

OS/2 Warp
Windows NT

CMVC supports the following "older" server platforms. This support will not be added to TeamConnection:

AIX Version 3
HP-UX Version 9
SunOS 4.1.3

Clients

Both products support clients (GUI and line commands) in the following platforms:

AIX Version 4
HP-UX Version 10
OS/2 Warp
Windows 3.1 (GUI only)
Windows 95 and Windows NT

Notes:

1. The port to Solaris is being done at the time this TR is being prepared.
2. The Windows 32 bit support in CMVC is beta only.

CMVC supports the following "older" client platforms. This support might be added to TeamConnection at a later date (depending upon demand):

AIX Version 3
HP-UX Version 9
SunOS 4.1.3

Databases

CMVC supports the following relational databases:

DB/2 Version 1 and 2
Oracle Version 7.x (excluding 7.3 and later)
Informix Version 5 and 7
Sybase Version 4.9 and 10

TeamConnection supports only the following object-oriented database:

ObjectStore (bundled with TeamConnection and not distributed separately).

License handling

CMVC uses NetLS to enforce that the maximum number of concurrent users complies with the number of licenses that the customer acquired. For more details, see “License handling in CMVC” on page 32.

TeamConnection, on the other hand, uses a utility named tclicmon (TeamConnection License Monitor) to monitor the audit.log of the family to see if the maximum number of concurrent users is within the bounds.

FAMILY ADMINISTRATION

CMVC and TeamConnection use a family to contain the objects and files associated with a project or a set of projects. For each family there is a database that is dedicated only to that family.

In UNIX, a system account is created for a family. In Intel, a directory is used for a family.

Structure of a family account

CMVC stores the file changes in a directory structure from the family file system and stores the information about the objects of the family inside the relational database.

In contrast, TeamConnection stores both the part changes and the information about the objects of the family inside the same database. As a result all updates happen within a single database transaction to improve data integrity.

The directory structure for CMVC and TeamConnection are similar; They are shown in Figure 2 and in Figure 3 on page 18; the comments describe the differences:

```
authority.ld
config.ld
cfgcomproc.ld
cfgrelproc.ld
config.ld
interest.ld
audit/log              Separate directory for audit log files
configField/
maps/                    The driver information is stored in the database
                           in TeamConnection
queue/messages         Separated addressing information and text
vc/                      The file data is stored in the database
                           in TeamConnection
```

Figure 2. CMVC family directory structure

```
authorit.ld
comproc.ld
config.ld
interest.ld
relproc.ld
audit.log                Audit directory removed
cfgField/
queue/                     Messages stored in one file in one directory
```

Figure 3. TeamConnection family directory structure

Notes:

1. The items marked with * indicate items that are needed to create a family.
2. The file names in configField (CMVC) differ from the ones in cfgField (TeamConnection).

The differences in the family directory structure between CMVC and TeamConnection are shown in Figure 4.

CMVC	TeamConnection
authority.ld	authorit.ld
cfgcomproc.ld	comproc.ld
cfgrelproc.ld	relproc.ld
configField/	cfgField/
queue/messages	queue/
maps	REMOVED
vc	REMOVED

Figure 4. Changes from CMVC to TeamConnection

Suggestions

In CMVC, it is strongly recommended (and is somewhat enforced by the tools) to have one family in a single system account.

In TeamConnection, it is also strongly recommended to have one family in a single system account; however, it is possible to handle several families in one system account.

Family administration tools

In TeamConnection, a GUI tool called TCADMIN is the utility that is used to create and to maintain a family. Most other tools have changed names. For more details see “TCADMIN - System Administrator's GUI” on page 31.

The format for the views of the configurable fields was changed from CMVC to TeamConnection.

NEW USER FUNCTIONS OF TEAMCONNECTION

This chapter elaborates on new TeamConnection functions of interest to general users.

SEQUENTIAL/CONCURRENT DEVELOPMENT

CMVC provides only the mode of sequential development, in which a file can be locked only to one user at a time.

TeamConnection also provides this mode of sequential development, but it offers a mode of concurrent development in which a part can be locked by more than one user at a time, and then when the different versions of the parts are checked-in, there will be collision records that the users would have to resolve to ensure that one user is not deleting the code added by another user.

This concurrent mode is available only per release and it is specified only at the time a release is created. A release that is created with sequential mode cannot be changed later on to concurrent development.

AUTOMATIC PRUNING OF RELEASES

In CMVC, every checkin of a file created a new version number. This leads to saving versions of files that were not useful in the long term. TeamConnection lets you control the versions you keep during your development process without limiting the number of times you check your files/parts into a workarea.

By using release pruning and workarea freeze you can specify how many of the intermediate checkins you perform will be kept and how many can be discarded after you integrate and commit a workarea.

You can prune a committed branch (workarea or driver) of a release by using the command "teamc release -prune".

It is possible to specify that a release can be pruned automatically for committed branches (work areas or drivers); this is accomplished with the +autopruning flag in the command "teamc release -create" or "teamc release -modify".

For more information about this topic and for a graphical explanation of the branches, consult the Versioning Model in Page 82, Chapter 2 "TeamConnection", in the manual "Introduction to the IBM Application Development Team Suite".

VERSIONING

In CMVC, only files are versioned either by SCCS or PVCS. These utilities determine the version numbers based on the number of changes and take into account when links are broken between releases.

The entire architecture of versioning in TeamConnection is new. TeamConnection versions most of the objects you use in development: parts, workareas, drivers and releases. The scope of a version is based on your release and workarea, so that the version of a part now has a meaningful context.

Furthermore, in TeamConnection, the version control allows the users to provide control over which changes are available to everyone by processing the workareas; for more details, see “Workareas are more than renamed Tracks” on page 25.

Versioning in CMVC

CMVC uses the UNIX utility SCCS as the underlying mechanism for the versioning of the files. The product PVCS can be purchased for use with CMVC on AIX 3.x platform.

In SCCS, numbers begin with 1.1 and the second digit is incremented with each version checked in (for example, 1.2, 1.3, etc). When a release or file is linked and then a link is broken during a check-in, a branch occurs in the numbering. For example, if release_1 is linked to release_2 when file_a is version 1.3, both releases will point to version 1.3. If only release_2 checks in a new version of file_a, the link will be broken to release_1 and the new version number will be 1.3.1.1.

The approach in PVCS is similar, but not identical.

The important point is that the numbering scheme is global and managed by SCCS/PVCS. It does not show any practical relationship between changes and the workarea/release where the change occurred.

Space is saved when storing files in CMVC in the following manner:

SCCS/text files are stored as forward deltas by SCCS. This saves space by comparing each file to the previous version and only saving the changes. The drawback is that it takes longer to extract the most current version (it must be constructed from the previous changes).

SCCS/binary files are stored as backward deltas by CMVC. If the changes between the current and previous version are beyond a specified level they the whole current file is

stored. Backward delta means that the most current version is a whole file and previous versions are stored as deltas. As a result it is faster to extract the current version but slower to extract old versions.

SCCS has lots of limitations in the types of data it can store so lots of files that you would expect to be text are really stored as binary.

PVCS/all files are stored as forward deltas by PVCS.

One drawback the separate processing for text and binary files in SCCS is that files cannot be changed on the fly from one type to another. If you want to change the type, then you must delete and destroy the file and then recreate it with the new type.

Versioning in TeamConnection

In TeamConnection, all parts are managed and versioned in the database. Here is what we do to them:

Text files are compressed and versioned using forward deltas.

Binary files are compressed and each version is stored whole.

This allows you to change on the fly the type of the part, from text to binary and vice versa.

Finding different versions of TeamConnection objects

The version control of TeamConnection maintains different versions of the following objects:

Releases
Work areas (and driver members)
Drivers
Parts

When you want to find and retrieve previous versions of these objects, it is helpful to know how TeamConnection creates and deletes previous versions of each object.

When you first create an object, the initial version name is the object name suffixed with ":1". When you create a new work area called "myWorkArea", for example, its version is "myWorkArea:1". Subsequent versions are identified in numerical order: myWorkArea:2, myWorkArea:3, myWorkArea:4, and so on.

Versions of releases and drivers are identified similarly: myRelease:1, myRelease:2, myRelease:3; myDriver:1, myDriver:2, myDriver:3; and so on.

Unique versions of parts are identified by association with a specific version of a release, work area, or driver. Your family may have three different versions of a part called

"myPart", one associated with release myRelease:2, one associated with work area myWorkArea:1, and one associated with work area myWorkArea:2.

PARTS ARE MORE THAN JUST FILES

TeamConnection parts are more than just files. The most important changes are:

Parts can be more than just files (of type text and binary). Parts may also be type "none" indicating that they will not be extracted as files; this is used for object level integration with other tools.

Parts include attributes defining their behavior during build and packaging.

File/Part links

From a user's point of view, parts are linked between releases and links can be broken in the same manner in CMVC and TeamConnection. However, there are two differences:

In TeamConnection Version numbers are not dictated by the when links are broken, rather, the numbering is based on the workarea and release.

In TeamConnection parts are copied into each release, regardless of whether there is a link. So if a part is linked in two releases it still takes up twice the space as it would if it were in one release.

Changing the type of a file (text <-> binary)

in TeamConnection you can change the type of a file without the need to destroy the file and then to recreate it again.

Notes:

1. The "type" field in CMVC is called "fileType" in TeamConnection.
2. The main changes about the file type from CMVC File to TeamConnection Part are:

   File -type text --> Part -text
   File -type binary --> Part -binary

   new Part -type (for fine grain objects)

WORKAREAS ARE MORE THAN RENAMED TRACKS

A workarea provides a logical view of a release that includes all of your changes and excludes the changes of all other workareas that have not been promoted.

While a CMVC track is just a list of the changes for a given defect or feature within a release, a TeamConnection workarea lets you see the impact of a defect or feature without needing to compare the version numbers of every file that has uncommitted versions.

In other words, a CMVC Track just contains file changes, but a TeamConnection Workarea is a view of the entire release. You cannot build a Track, but you can build a Workarea because it includes all the parts.

In CMVC you could use a release process such as 'prototype' which would allow you to create and checkin files without the need to specify a track.

By contrast, in TeamConnection, you always have to specify the information about the workarea in order to create and checkin parts. However, in release processes such as 'prototype' it is not necessary to name a workarea after a defect or feature, that is, you can use any name that may help you identify the workarea (it has to be unique).

Multiple workareas per defect or feature

A significant improvement in TeamConnection over CMVC is the ability to create more than one workarea for a defect or feature within a release. This allows you to break up the work you are doing into pieces, even to commit the work a piece at a time. You can include parts from other workareas, including currently committed drivers or releases.

Workarea performance

For users of prototype releases it is important to periodically integrate a workarea into the release in order to maintain good performance.

teamc workarea -undo

Workareas are versioned objects; as a result, you need to undo new versions of workareas in TeamConnection just like you do files in CMVC.

If a workarea has been frozen or refreshed from another workarea (refreshing causes an implicit freeze), then teamc part -undo is not enough to remove all changes from the workarea (as File -undo could do for a CMVC track). After using Part -undo on any files that have been

changed since the last freeze, use teamc workarea -undo to remove each freeze that occurred.

The "teamc report -view versionView" command reports each freeze that occurred (that is, reports each version of the workarea). From the GUI WorkArea window, select Selected->Show->Versions.

DRIVER HAS MORE OPTIONS THAN LEVEL

Because the CMVC map files have been replaced with state data in the database, drivers can now list all of the parts and their versions that are in a committed driver.

One side effect of the change from Levels to Drivers is that it takes longer to integrate workareas into drivers.

The "teamc driver" command has two new options that give you more flexibility.

teamc driver -freeze

Because a driver is really a specialized workarea, you can save a snapshot of the driver's contents prior to committing. Depending on how you have set the pruning of the release, this allows you to either temporarily or permanently version the changes in a driver. This is particularly useful if you do not commit drivers frequently.

teamc driver -restrict

Restrict is a state for drivers between integrate and commit. The purpose is to allow a release's build administrator to limit the ability of other users to change the contents of a driver while building and verifying the build.

The typical use of restrict is in conjunction with some automated promotion and build process. For example, at 7 PM every night a process is started that adds all workareas in a release that are in integrate state, that is, all the fix records are completed, to the driver by creating driver members. After adding the workareas to the driver, a build is started. Without the restrict state the only way to prevent changes to a driver while build is running is to either change all users access (affecting the entire release) or to commit the driver (preventing further changes).

By using restrict, the tool can commit the driver after a successful build or regression test. If the build or regression test fails, then notifications can be sent to the development team to make the necessary changes to complete the build and pass the regression test. At that point

the build administrator can promote the changes, build, test and commit the driver without anyone else changing the driver contents.

UPDATED INTEL GUI

The GUI for Intel operating systems has evolved significantly from the original CMVC GUI.

Examples of new features in the TeamConnection GUI are:

The TeamConnection Part command adds an -edit action that performs checkout, opens an editor session, then checks the part back in after the editor session is closed.

Better context-sensitive menu options when selecting an entry in a dialog by pressing the right-mouse button.

More options on the Settings dialog, such as selecting whether the read-only attribute should be set.

Note: This compares the TeamConnection V2 Intel GUI with the CMVC version 2.3.0.18 Intel GUI.

NEW UNIX GUI

CMVC users of the OS/2 or Windows GUI are already familiar with the TeamConnection Intel GUI. The TeamConnection Unix GUI is now based on the Intel design; as a result of using the Intel design, the following has changed for Unix GUI users:

You now have a settngs page for changing the defaults for options. You should not need to edit the Teamcgui X-Windows resource file to change the teamcgui defaults.

The task list queries now support the use of environment variables. As a result, the default tasks do not need to be customized to be useful.

There is a significant increase in the use of icons that are associated with each TeamConnection object. For example, there is a generic icon for defect, another for component, etc; then when you are showing a list of defects you will see the same generic icon in the left hand side, one icon for each actual component. Each icon provides additional information on the state of the objects, such as a defect in working state has a different icon than a defect that was canceled.

The GUI uses "containers" to show the components and drivers, and you can select a Tree-View which will show the icons with a plus or a minus sign. Then you can click on these signs to expand or contract a specific component. This provides an equivalent function to the CMVC Component Tree hierarchy.

A command window has been added. This allows you to use line commands without out opening a shell window.

After selecting an object (for example, a defect in the defect window), a right-mouse click provides a context sensitive menu of actions.

You an edit and invoke queries on each object window without opening the filter window, by using the footer in that window.

AUTHENTICATION BY MEANS OF LOGIN

TeamConnection provides several authentication methods, one of which requires users to use the command called "tclogin". With this command the user uses the TeamConnection License Manager in the client and logs in once into a TeamConnection family using a valid password. If the user is authorized, then the family server and the login manager keep track of the login status and allows the user to issue other TeamConnection commands.

The family administrator could setup the family to use the following methods for authentication.

Only use host entries, which is the default: HOST_ONLY

This is the only option in CMVC.

Only use the method of login with password: PASSWORD_ONLY

Either a login with password or a host entry, in that order: PASSWORD_OR_HOST

No authentication method at all: NONE

This security feature affects the role of the "Host List" for reaching the doors of the family and for getting in. However, once the user passes the door and gets into the family, then the "Access List" provide the proper authorization to perform certain tasks in TeamConnection. This authentication feature does NOT affect the role of "Access List".

While most companies have no reason to worry about someone reconfiguring their workstation/PC to have the same hostname as another machine on the network in order to acquire another user's privileges, it is sometimes a security issue. The authentication method of PASSWORD_ONLY in TeamConnection provides an option to use passwords instead of hostnames as a means of authentication. Because TeamConnection only sends encrypted passwords across the network, this is a vastly more secure form of authentication. This option makes it easier for users to access TeamConnection from mobile locations.

PLANNING FOR TEAMC RELEASE/DRIVER -EXTRACT ACTIONS

TeamConnection V2 extracts all files directly to the client. It is no longer necessary to set the userid (uid) and the group-id (gid), or to worry about the ability to mount a client directory to the server. However, planning is still required whenever you want to manage an extract to a system or to another user other than the client invoking the extract.

Our recommendation is to do an NFS mount or use any other tool at your disposal (for example, OS/2 NET USE) to place the target directory to the reach of the local host where the family daemon is running and then extract to the client as you normally would.

The other popular option would be to issue an rexec to the target machine so that a client installed on that machine can run the extract.

BUILD SUPPORT

The function that enables you to define the structure of your application and then to create it within TeamConnection from your input parts.

You can build applications for platforms in addition to the one TeamConnection runs on; currently, you can use TeamConnection to build applications on AIX, HP-UX, OS/2, Windows NT, Windows 95, and MVS. However, because the build function is homogenous, you need to plan to build each platform individually.

With TeamConnection's distributed build, managed components or releases can also be built for the enterprise client/server environment (that is, for the target platforms that may be different from the server platform used for development).

With TeamConnection, a development team can set up the build structure once and be able to identify the build rules depending on the task at hand. The build rules are associated with the objects being built so there is no need to search for related build steps in a file used by a separate build tool. The software configuration management services rebuild only those objects that need to be rebuilt, based on criteria such as date changes. This minimizes the system resources required to accomplish each build reliably.

The TeamConnection software configuration management services also determine which parts of a build need to be built in serial and which can be in parallel; it then can parcel out the work to the available machines.

Building objects from different technologies, such as procedural and object-oriented languages, through a single rule-based build process reduces integration risks and complexity.

PACKAGING SUPPORT

After you complete a build, you can run a specialized build to package and even distribute your product. Currently, TeamConnection supports a generic packaging tool called "gather" and the Netview/DM (Distribution Manager) facility that is now part of Tivoli.

OBJECT REPOSITORY AND INFORMATION MODEL

TeamConnection integrates an object-based repository with software configuration management functions to provide a managed environment for data-related assets at all levels of business and application information.

With TeamConnection, team members of different disciplines interact with the repository to work with information in a form that makes sense for their task. The repository provides a central point of controlling and translating information by allowing access to the data through the integration of tools.

A repository provides multiple task-driven views of stored information, as well as, "inherited" services to all the objects that it manages. Tool integration with the TeamConnection repository provides the ability to decompose the processes of each discipline in a team, analyze them, and transform their managed data-objects into their conceptual, logical, and physical implementations, each with their respective views.

Underlying all of the views (conceptual, logical, storage) are the object and class definitions required to support these views. This set of definitions represents the Information Model. The TeamConnection repository architecture includes a set of APIs and an open extendible information model so that tools can share data, store their unique data, and use the common software configuration management and repository services available in TeamConnection.

This capability allows for an open repository that can be extended through TeamConnection services by both tool vendors and developers of in-house tools. Specifically, TeamConnection's information meta-meta model provides a common language for representing various tool-specific meta-models. A tool can reuse or extend TeamConnection classes to define a tool model reflecting attributes, relationships, and behaviors of the objects to be created and used.

The information model and the functions for the tool builder are provided separately in the Tool Builder's Development Kit.

CHANGES THAT IMPACT SYSTEM ADMINISTRATORS

There are significant differences in the way CMVC and TeamConnection work and use system resources. The following sections address some of these differences. Much of this information is not fully documented in either the CMVC or TeamConnection administration manuals.

TCADMIN - SYSTEM ADMINISTRATOR'S GUI

Note: The TCADMIN tool is not yet available in UNIX.

While TeamConnection still allows you to create, configure and manage your families using commands, there is a new GUI, TCADMIN, that makes all of the configurable elements of a TeamConnection family obvious and easy to manage.

Also, you can keep track of all of your families at once by and select which family to modify from the main dialog.

For each family you can specify to start/stop the daemons and to change the following characteristics, which are presented as a notebook:

Family information
Initial superuser
Configurable fields
Component processes
Release processes
User exits
Security
Authority groups
Interest groups

Other administrator tools

In CMVC, chfield actually interacts with the database and validates the configurable field definition files in configField. In contrast, in TeamConnection, chfield does not interact with the database and relies only in the configurable field definition files in the cfgField directory.

USER EXITS

The CMVC user exit paradigm is supported in TeamConnection. However, the parameters on almost all commands have changed. Further, the possible values for many of the commands have changed. For example, the file "type" in CMVC is either "text" or "binary" but in TeamConnection the values are "text", "binary" and "none". Even worse, in several commands, the values are passed as integers (that is, 0, 1, 2) instead of text strings.

TeamConnection adds significant ease of use on top of the CMVC paradigm. You can cause a parameter file to be created that contains the values of specific parameters in a binary files. The sample program teamcenv.c allows you to extract the values from the files.

TeamConnection adds a family administration GUI that supports the setting up of user exits; see “TCADMIN - System Administrator's GUI” on page 31.

The most significant limitations of TeamConnection user exits are that it does NOT support issuing ANY TeamConnection commands in a user exit. Also, there is NO way to directly access the database using SQL.

For user exits that modify the contents of files as they are extracted, checked in, etc. there is one minor change. In CMVC, all text files were normalized to the UNIX standard (carriage-return, CR, between each line). In contrast, in TeamConnection, files are checked in "as is", then normalized by the client during extract or checkout. As a result, the temporary file on the server that is accessible by a user exit may contain CR/LF (Intel standard carriage-return/line-feed) or CR between lines. Further, tabs and/or spaces are altered on the client on the way out.

LICENSE HANDLING

License handling in CMVC

In CMVC, the licenses for the UNIX clients are managed by the use of the product NetLS (also known as iForLS or iFor). This product enforces the number of concurrent users that can work with CMVC. However, NetLS could be difficult to install, to administer and to maintain, specially in complex networks. Further, an inefficiently running NetLS can add up to one minute to each CMVC command. Because of this, the use of NetLS is one of the major dissatisfactors for some CMVC customers. By the way, the licenses for the OS/2 and Windows clients do not use NetLS.

License handling in TeamConnection

TeamConnection chose a simpler approach for the licenses: the honor system. That is, the customers will use only the licenses for which they are entitled to. In order to help them with this task, TeamConnection provides a License Monitor utility that will scan the audit log of a family server and will report the highest number of concurrent users. The number of concurrent users is defined as the number of unique combinations of TeamConnection user id, system login, and host name that use the family server in a period of 15 minutes (this default value can be modified).

BACKUP AND RECOVERY

ObjectStore provides a utility, osbackup, for backing up individual databases. You do not need to stop TeamConnection in order to run the backup, although it is recommended so that you can backup all databases (for each release) at one time, insuring that the backup is consistent for the entire family. There is more documentation on backup and recover in the TeamConnection Administrator's Guide.

At the present time there is no facility for removing data from a TeamConnection family. It is anticipated in the future that it will be possible to break the links for a release that uses its own database (a new option when creating a release).

WHAT PROCESSES ARE STARTED

What processes are started in CMVC

CMVC has a fairly complex hierarchy of daemons, where a single daemon (called "the grand-parent") is started, followed by the number of daemons requested; for example, "cmvc testfam 3" would start 3 daemons. This second set of daemons are called "parents". After all of the parents have started, the grand-parent is terminated.

Next, each parent starts a "child" daemon that listens for CMVC clients requesting that a CMVC command be processed. So, from the example above, 1 grand-parent starts 3 parents which in turn start 1 child each producing 3 children, but when the grand-parent terminates, then there will be 6 processes left running.

When a command is issue that needs to temporarily change to your user ID, like "Release -extract", a grand-child is spawned for the duration of the command (with your user ID as the owner). Grand children are also generated for all file operations, where changing to the family account is necessary before running SCCS commands.

Also, when calling user exits it is necessary to spawn a grand-child running as the family account in order to prevent a user exit from getting access to root authority.

All this was done so that CMVC could handle almost any sort of failure without reducing the number of daemons available to service your requests.

The process hierarchy for CMVC daemons is shown in Figure 5.

COMMAND:

cmvcd testfam 3

HIERARCHY:

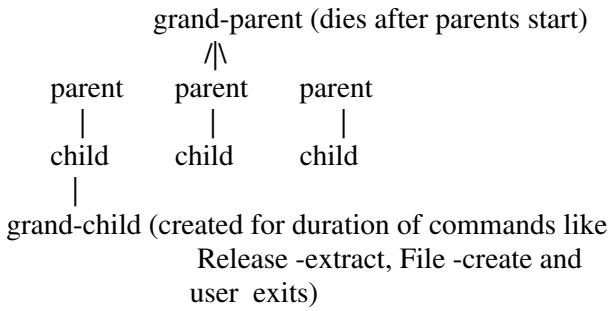

Figure 5. CMVC daemon process hierarchy

What processes are started in TeamConnection

TeamConnection has a much simpler architecture. A single parent spawns one child for each family daemon you request. Also, the parent will spawn a build agent and build processor daemons as requested in the startup command.

The parent stays around to keep an eye on all of the children. Because the parent does not open the database, therefore it uses no database resources.

TeamConnection is not a setuid process and does not need to change user IDs, thus, it only needs to spawn grand-children for such things as user exits.

The process hierarchy for TeamConnection daemons is shown in Figure 6 on page 35.

COMMAND:

teamcd -a buildsystem@2 -p pool1 -e HP -s 3 -n testfam 3

HIERARCHY:

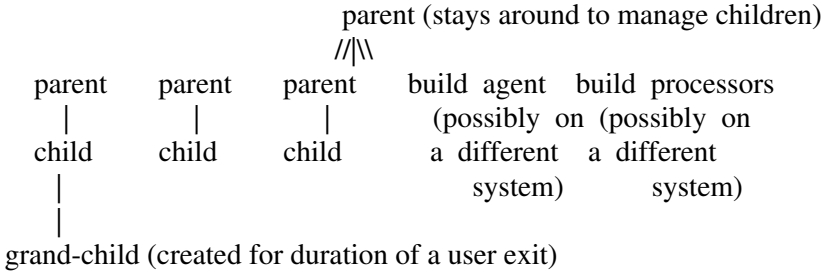

Figure 6. TeamConnection daemon process hierarchy

Note: In order for processes in TeamConnection to start and stop properly, you should start all processes using the options on the teamcd command. This will ensure that stopteamc will find all processes and terminate them.

USE OF DISK SPACE AND MEMORY

Directory /tmp

CMVC requires that /tmp (or the directory pointed to by the TMP variable) must have free space equal to three times (3X) the largest file to be checked into the family. This is pretty vague, because you rarely know how big your largest file will be.

TeamConnection uses /tmp on a per process basis. The use of /tmp by the database ObjectStore can be changed by setting the OS_CACHE_DIR variable in Unix and OS/2, and OS_TMPDIR for Windows installations.

The amount of space used by each process is determined by the value of OS_CACHE_SIZE (the default for OS_CACHE_SIZE is 8 Mb). For example, if you have 3 families, each running 2 daemons plus one build agent each and OS_CACHE_SIZE is set to 100 Mb, you will need 900 Mb free in /tmp:

((6 child daemons + 3 build agents) X OS_CACHE_SIZE of 1 Mb) = 9 Mb

Disk space for the family account

In CMVC, almost all of the disk space is allocated to the vc directory structure. This is where the files are stored. When a file system becomes too big, we recommend that you select a subdirectory (such as, $HOME/vc/0/2) and symbolically link that directory into another filesystem. This allows you to overcome any filesystem size limits. The next largest directory is where you store the database backup or export.

Space used by database

In TeamConnection, all files and other data are stored in the database. In order to distribute the space consumed by a family, we recommend that you specify a separate database for each large release.

Because ObjectStore allows you to physically distribute the databases for your releases across systems, you have greater flexibility with space allocation than you did with CMVC. There are directions for distributing in the TeamConnection Administrator's Guide

Space used by family daemons

The TeamConnection temporary directory, tmp/tctmp, stores a temporary copy of each part of type tcpart that is checked in or out. The other files stored in the temporary directory tend to be small, so no significant space needs to be allocated.

Use of AFS and NFS

Network distributed file systems such as AFS (formerly known as Andrew File System) and NFS (Network File System) are commonly used for the home directories of users. Furthermore, DFS (Distributed File System) is an industry standard component of DCE (Distributed Computing Environment) and a replacement for AFS and it is expected to become a popular new alternative.

The CMVC development and support teams has always discouraged their use for the contents of family accounts or databases, because NFS can be unreliable and because AFS tokens can expire; therefore, we strongly recommend that the database and contents of family accounts must reside physically on the systems running the CMVC daemons. We do understand that customers have successfully moved less frequently used files in the vc directory structure to NFS mounted drives, but we hope that is limited to a last resort.

TeamConnection databases do not work in AFS or DFS filesystems. However, as already indicated, ObjectStore supports distributing databases across systems and this configuration involves the use of NFS.

MEMORY USAGE

In CMVC, most of the memory is used by the database processes; in TeamConnection, it is the same.

While ObjectStore has different defaults for maximum values for each operating system (for example, in HP-UX is about 300 Mb per process, AIX is 1 Gb per process), this value is controlled by the OS_AS_SIZE variable.

You only need to be concerned about OS_AS_SIZE if you get errors from transactions reporting "Address Space Full".

FAMILY DAEMONS AND SECURITY/INTEGRATION ISSUES

Process privileges

The CMVC daemon processes run with the setuid bit on and are owned by root. This allows CMVC to become a specific user (other than the family account) as needed, as well as running the mount command which is owned by root.

TeamConnection does not need to use the mount command or change the user id, therefore the TeamConnection daemons do not use the setuid bit and are NOT root processes.

The following CMVC security/integration issues are eliminated by this TeamConnection architecture:

There is no need for TeamConnection daemons to be owned by root, thus, eliminating the possibility of a user or system administrator "stealing" root.

The functionality of the CMVC Release/Level extract was changed in TeamConnection Release/Driver extract so that, NFS mounts are NOT needed in TeamConnection. Thus, this change eliminates the need for use of the mount command or to change to a user ID. Another benefit is the elimination of mount failures due to differences in TCP/IP of various operating systems.

Access to data in the family account

In CMVC, it is necessary for all of the files and directories in the vc directory to have read access for group and other. Even though it is not possible to determine which file is which without access to the database, this is still a security issue.

In TeamConnection, the files are stored in the database so that they are much more difficult to access. Further, you can now be more restrictive in the file and directory permissions you use for your family account.

System integration issues

The TeamConnection daemon does not have to interact with SCCS. The greatest benefit is that all the work happens in a single database transaction, which improves data integrity. This also eliminates 2 environment variables, as well as a long list of miscellaneous restrictions on the format of "text" files, as well as a lengthy list of obscure SCCS error messages.

Other issues

The teamc report command now has access controls. It is no longer possible for a user without permissions via component access lists to see the data through the report command.

APPENDIX A. 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

Add TC TRs here

TC TRs

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, MVS DB2/6 , DB2, CMVC VisualAge, TeamConnection,	IBM Corporation
NetView/DM, Tivoli	Tivoli Systems, a subsidiary of IBM
ObjectStore	Object Design, Inc.
UNIX, USL	UNIX System Laboratories, Inc.
NetLS, Network Licensing System, iForLS, iFor	Gradient
OSF, Motif, DFS, DCE	Open Software Foundation, Inc.
PostScript	Adobe Systems Incorporated
INFORMIX	Informix Inc.
ORACLE	Oracle Corp.
SYBASE	Sybase Inc.
NFS, SunOS, Solaris	Sun Microsystems Inc.
HP-UX, SoftBench	Hewlett-Packard Company
Windows, Windows NT, Windows 95	Microsoft Corporation
X Window System	Massachusetts Institute of Technology
PVCS	InterSolv, Inc.

| AFS | Transarc Corp. |
+---------------------+--+
| Intel | Intel Corp. |
+---------------------+--+

END OF DOCUMENT