Comparison between TeamConnection 2 and CMVC 2.3

Document Number TR 29.2321

Angel Rivera, Lee Perlov and Sam Ruby

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

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

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.

Mr. Ruby joined IBM in 1981, working on software tools for the Federal Systems Division.

CONTENTS

FIGURES

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 technical report supersedes "TR 29.2253 Comparison of CMVC 2.3 and TeamConnection 2". Based on customer feedback we reworked many sections of the previous document and added a lot of new material.

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 49.

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** 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 FOR USERS

Here are afew things that will be significant changes for CMVC users migrating to TeamConnection. There is more detail in the sections “Comparison of common functions” on page 5 and “New user functions of TeamConnection” on page 23.

- 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 5.
- The versioning scheme is different:
  - Files (parts) are NOT versioned by checkout/checkin, but by the number of freezes in a workarea.
  - Version numbers are named relative to the workarea (for example, the part a.c can have a version number work1:3).
  - Drivers and releases are essentially specialized workareas, and therefore versioned.

  For more details see “Versioning” on page 24
- Workareas are much more important than tracks and behave differently. For more details see “Workareas are more than renamed Tracks” on page 27.
  - 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 committed, you will need to refresh from that workarea.

  - You must freeze a workarea (explicitly or implicitly) to create new versions of parts changed in that workarea.

- When performing queries, like filling in a filter window, you are often querying within a context instead of the entire family. You need to specify that context by entering release, driver or 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 16.

THE BIG DIFFERENCES FOR FAMILY ADMINISTRATORS

Here are a few things that will be significant changes for CMVC Family Administrators migrating to TeamConnection. There are more details in the sections “Family administration” on page 19 and “Changes that impact system and family administrators” on page 35

- There is a single installation for the TeamConnection software and the ObjectStore database (which is used by TeamConnection). You do not need to be an expert at a database before administering TeamConnection families.

  Note: Object Design Inc. provides courses on the administration of ObjectStore.

- All of the data is stored in the database, including files. You will not need to know SCCS in order to keep the database and file versions synchronized. This also makes backup much easier.

- Most of the administration commands have changed. See the **TeamConnection Administrator's Guide** for more details. Further, a separate GUI is provided for family administration on the Intel operating systems. It is planned to port the GUI to the Unix operating systems.

- TeamConnection build administration requires maintaining pools of build machines so that TeamConnection can properly perform release builds. This means that TeamConnection requires more extensive planning and management for the network.

- The migration facility is different in TeamConnection than in CMVC. Further, we recommend you follow the process documented in the Technical Report, **Migration from CMVC 2.3 to TeamConnection 2.0**.

THE BIG DIFFERENCES FOR TOOLS INTEGRATORS

CMVC relied on framework tools such as **SDE/Workbench** to provide tool integration. SDE/Workbench provided build integration through "make", GUI integration and messaging, etc.

TeamConnection is actually a point of integration, and it provides:

- A generic build interface that allows for traditional builds, as well as packaging and distribution. See “Build support” on page 32 for more details.
- A client cache that allows client applications to extract and update data in the TeamConnection family without calling client commands. See “Object Repository and Information Model” on page 33 for more details.
- An API to allow client applications to perform TeamConnection tasks using function calls instead of client line commands. See “Integration with other products” on page 16 for more details.

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 11 and “Features” on page 11) 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 and Concurrent development” on page 23.

- 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 and Concurrent development” on page 23.

- 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, drivers and releases are actually specialized forms of workareas. As such, they are versioned objects that can be queried for change history, can be extracted, can be built, etc. The CMVC "current" release concept has been removed.

  For more details see “Versioning” on page 24.

- 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 32.

- Pruning.

  A release can be explicitly or automatically pruned to conserve space in the family account and to remove change history that is no longer needed.

  For more details see “Automatic pruning of releases” on page 23.

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 6.

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 6.

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 24.

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 only versioned as part of the workarea. What this means is that when you check out a part, change it, then check it back in, the previous version will be replaced.

  New versions of parts can also be created by freezing the entire workarea; in this case, only those parts that have changed in the workarea get new version numbers.

- TeamConnection parts cannot be explicitly destroyed. However, it is possible to reclaim space by pruning workareas in the release.
- 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 24 and “Parts are more than just Files” on page 26.

- The TeamConnection Part command adds actions to support the building of parts. When building a part that is defined as an "output", this causes the building of all of the "input" parts. Therefore, building the appropriate output part will build a complete executable.
- 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 keyword capabilities.
- The TeamConnection Part object retains in the attribute "last update", during check-in, the date and time of a file from the file system. 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 last update 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 27.

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 always specify 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 27.
- 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 28 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 allow the use of passwords and to optionally require users to login into TeamConnection. The use of passwords makes optional the use of the hostname and system login as part of the authentication. For more details, see “Password login to TeamConnection” on page 30.

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 33 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 19 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 36.

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, which has been withdrawn as a product, so TeamConnection is not integrated to Softbench. It is the intention that TeamConnection would integrate in the future to other similar "framework applications".
- In TeamConnection, API's are provided to access TeamConnection data through an Object Repository (see “Object Repository and Information Model” on page 33). Applications such as VisualAge Generator use the API's to integrate to TeamConnection. The VisualAge brand family includes several other products that integrate directly to TeamConnection; there is more integration work under development.

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. As a result, the syntax and error messages vary depending on the database used with CMVC.
- 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). As a result, TeamConnection reports in raw format often have more fields, and some of the fields (for example, versionSID) may be interpreted differently.

- 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 2.3.0 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 and CMVC 2.3.1 use 4 digits to store the information about the year. Thus, TeamConnection and CMVC 2.3.1 are 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 2.5 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 does not require that any particular database already be installed, instead it bundles the appropriate version of the ObjectStore database by Object Design Inc. (ODI), which is included with the price of TeamConnection.

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 36.

TeamConnection, on the other hand, provides 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 operating systems, where file security is based on a separate user login, it is recommended that each TeamConnection family be in a separate family login. The sample profile for families assumes a separate user login for each family.

Operating systems that run on the Intel platform, although some offer user login capabilities, do not benefit from this added file security. As such, OS/2 and Windows NT set up documentation only recommend separate directories.

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 on page 20 and in Figure 3 on page 20; 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 on page 21.

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

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 - Family Administrator's GUI” on page 35.

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 AND CONCURRENT DEVELOPMENT

CMVC and TeamConnection provide a sequential development mode, in which a file can be locked only to one user at a time.

TeamConnection also provides a concurrent development mode in which a part can be locked by more than one user at a time. In concurrent mode, parts can be checked out and back in without warnings that anyone has modified the file. However, when the workarea is refreshed, such as prior to adding the workarea to a driver, collision records are created for workareas that change a part already modified by another workarea. Collisions records must be resolved, usually through the use of TeamConnection's merge tool to combine the changes to a part in multiple workareas, prior to adding the workarea to a driver.

This concurrent mode is available on a per release basis and may be specified at the time a release is created; however, once a mode is established for a release, it cannot be changed.

Note: In concurrent mode, the workarea -check ignores prerequisites and corequisites.

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, files are versioned either by SCCS or PVCS. These utilities determine the version numbers used and reported by CMVC. Creating new versions of files increment these version numbers, then creating new releases, linking releases and breaking those releases cause branching that results in more complex version numbers for parts (for example, 1.2.1.4).

While these numbers show the relationship of all file changes across all releases that share the file, the numbers do not have a relationship to the tracks that manage the changes to the file. In other words, there are two different tracking mechanisms that need to be manually reconciled.

TeamConnection version numbers reflect the workarea used to change a particular part. For example, if part a.c is created in workarea Work1, then the first version of a.c will be named Work1:1. If more than one version of a part is checked in and frozen, the version numbers will increment sequentially (for example Work1:2, Work1:3, etc.).

The reasons that the parts are versioned based on a workarea is that the workarea is actually the object being versioned. Further, because drivers and releases are actually specialized forms of workareas, they too are versioned. The benefit of this is that you have access to all the frozen parts of all workareas, drivers and releases that are not pruned. This provides a consistent view of all current and frozen workareas, including their change history; this is something that has always been lacking in CMVC. For more details, see “Workareas are more than renamed Tracks” on page 27.

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 may be stored as reverse deltas using a reverse delta versioning scheme internal to CMVC. If the changes between the current and previous version are beyond a specified level then the whole current file is stored. Reverse 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 significant limitations with respect to the types of data that can be stored. As a result, any file that cannot be stored in SCCS is stored as a binary using the CMVC internal binary mechanism.
- 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 reverse deltas.
- Binary files are compressed and each version is stored whole.

You can 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 that define their behavior during build and packaging. For example, a COBOL program will be associated with a different parser than a C program, as well as a different builder that calls a different compiler with different parameters.
- Parts that are type "output" will be checked in after a build. The number of output files that can be stored in a release is set for that release.
- Parts that are "temporary" are discarded after the build, without being checked in.
- Tools may define subclasses of parts which may contain other attributes, relationships and dependent objects.

File/Part links

From a user's point of view, parts are linked between releases and these links can be broken in the same manner in CMVC and TeamConnection. However, in TeamConnection, the version numbers are not dictated by the when links are broken, rather, the numbering is based on the workarea and release.

In TeamConnection the database information for a part is copied into each release, regardless of whether there is a link. So if a part is linked in two releases, then the database information about the file will be stored twice (in each release), but the actual contents of the file will be

stored only once. That is, a part linked in two releases does not take the double amount of space as if the part were not linked.

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

Because text files are subject to some character manipulation (such as replacing line feeds with carriage return/line feed in OS/2 and Windows, and the insertion of keyword values), there are times when a file that has been stored as text needs to be changed to binary, or vice versa.

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. There is a syntax change from CMVC to TeamConnection:

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

3. The Part -type flag has a new meaning in TeamConnection: it is used to define fine-grained 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 'proto-

type' 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 can undo new versions of workareas in TeamConnection. You can also undo individual part changes.

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, it 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 may take 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 GUI for UNIX. The CMVC GUIs for OS/2 and Windows include most of the features provided by the TeamConnection 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, then 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 settings page for changing the defaults for options like the Component name. You should not need to edit the Teamcgui X-Windows resource file to change the teamcgui defaults.
- The task list queries support the use of environment variables. As a result, the default tasks that already include environment variables 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 on the left hand side for each defect. Each generic icon is augmented with minor graphics to provide 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 can edit and invoke queries on each object window without opening the filter window.

PASSWORD LOGIN TO TEAMCONNECTION

TeamConnection adds another method of authentication that is complementary to the entries in the Host List table. Through the **tclogin** command a client can enter a password in order to log the client session into TeamConnection. If the user is authorized, then the family server and the login manager keep track of the login status and allow the user to issue other TeamConnection commands.

The family administrator can configure the family to use one of the following authentication methods:

- 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

The password option solves several problems associated with using host lists:

- Users that access a family from a large number of workstations require less effort to manage a password that a long list of host table entries.
- The potential security issue of one workstation taking over another workstation's IP address in order to gain another user's access to a family, is eliminated by passwords.
- Multiple users can access the family from the same workstation and user account without inadvertently using another's TeamConnection User ID.
- A user's login can be explicitly terminated and not taken over by another user. A user's access is terminated whenever the family server is recycled, the user logs off by using tclogin or the user logs out of the the workstation.
- It is easier for a family administrator to temporarily disable a user account with passwords than deleting a user's host list entries.

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. A single build pool can include systems running any or all of the operating systems and TeamConnection will route the build to the appropriate machine.

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.

While distribution is part of the build subsystem, the **teamcpak** utility can be run separately.

Although the build function is heterogeneous, from a practical perspective the packaging function is essentially homogeneous. Currently, the executable files are operating specific, as are most of the installation utilities used for packaging (for example, InstallShield, installp, etc.). The increasing popularity of languages like Java may allow for heterogeneous packaging in the future.

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 AND FAMILY 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 - FAMILY 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 - Family Administrator's GUI” on page 35.

TeamConnection now supports running TeamConnection actions from within a user exit. As with CMVC, there is always the limitation that the action cannot access the same data as the command, in such a way as to cause a database deadlock. You will need to test to determine what is an acceptable action in any user exit.

The database can be accessed using SQL through the **tcselect** utility that is provided with the family server.

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 just LF between lines.

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 a lot of overhead to each CMVC command. Because of this, the use of NetLS is one of the major causes for dissatisfaction 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 whole releases from a TeamConnection family. As stated earlier, workarea pruning can reduce the amount of data in a specific 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 daemon processes. This second set of daemon processes are called "parents". After all the parent processes have started, the grand-parent process 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.

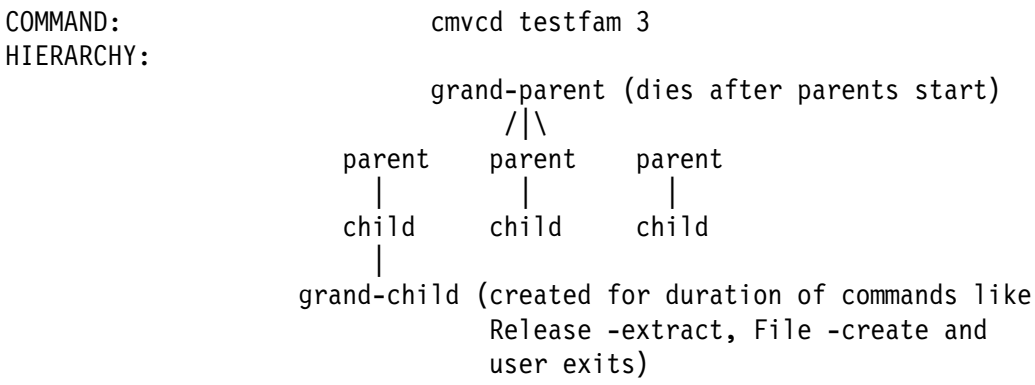

Figure 5. CMVC daemon process hierarchy

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 process stays around to keep an eye on all of the children processes. Should a child process die for any reason, the parent process will start another child process. Because the parent process does not open the database, it does not use 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 39.

```
COMMAND:   teamcd -a buildsystem@2000 -p pool1 -e HP -s 3 -n testfam 3
HIERARCHY:
                              parent (stays around to manage children)
                              //|\\
         parent    parent    parent    build agent  build processors
           |         |         |       (possibly on (possibly on
         child     child     child      a different  a different
           |                             system)      system)
           |
       grand-child (created for duration of a user exit)
```

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 on Unix

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 100 Mb) = 900 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 most 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" while it is being 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, DFS 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. Also, DFS (Distributed File System) is an industry standard component of DCE (Distributed Computing Environment) and a replacement for AFS.

The CMVC development and support teams have always discouraged their use for the contents of family accounts or databases, because NFS can be unreliable and because AFS/DFS 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.

ObjectStore databases can be distributed across multiple systems as long as each system that manages an ObjectStore database is running the **osserver** process. This requires that the ObjectStore server be installed on each system managing one or more databases.

Please note that the TeamConnection databases will not work in AFS, and currently do not work in DFS file systems.

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.

Initially, you should not need to set this value. If however, you get errors indicating "Address Space Full", then you can set OS_AS_SIZE in order to overcome this problem. If you find yourself setting OS_AS_SIZE, you should probably also review your system requirements and consider upgrading the memory and/or swap space on that system.

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 need NOT be 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 and this eliminates the possibility of users with no root authority to "steal" or "misuse" the root authority.
- 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, files are stored in the database in a way that is not directly accessible to end users. 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.

CUSTOMER FEEDBACK

Now that there has been time for customers to move to TeamConnection Version 2, they have provided feedback on how TeamConnection allows for process improvements over CMVC. Here are some of the most significant process improvements:

Note: Where a specific customer made the process improvement, the heading begins with "Customer scenario:".

CUSTOMER SCENARIO: ASSIGNING MULTIPLE WORKAREAS

Multiple workareas makes it easier for people to work on the same problem without getting in each other's way and maintain better records of their changes.

Because many defects require the participation of more than one person, several customers have been pleased by the ability to create and assign a workarea to each contributing person. For example, a workarea can be created for the developer changing the GUI, another for the developer changing the database code, and a third for the technical writer updating the documentation. Each person makes changes in their own workarea without interfering with the others. As a result, when all of the workareas are added to drivers (not necessarily the same driver), there is an accurate and easy way to follow the change history for each person's work.

In CMVC, the solution was usually to create extra fix records or to change the ownership of fix records as different people work on their portion of the defect. Of course, the CMVC solutions were really only workarounds because a fix record is automatically created for each component impacted (which does not always correlate to who is doing the work).

CUSTOMER SCENARIO: ALLOWING INCREMENTAL DEVELOPMENT IN ONE FEATURE

Multiple workareas for a feature or a defect also allows for easier incremental development.

A single team member, or a group, can open multiple workareas for a feature or defect, then add their changes incrementally, until the solution is complete. Team members can check changes into workareas, one workarea at a time, and have them committed to drivers or directly into releases without the hassle and confusion of using multiple features and defects.

In TeamConnection, a feature or defect remains in working state as long as at least one workarea is still in the fix state. Any number of workareas can be integrated, committed and completed prior to moving the feature or defect to test, verify or complete.

Further, TeamConnection allows a team member to freeze the state of a workarea at any time, without needing to commit that workarea.

In CMVC, the only available solution was to:

1. Let a feature or defect go to test, verify or complete,
2. Explain to the tester or person opening the feature/defect that they have to wait for additional changes.
3. Open another feature/defect, typically using some naming convention that reflected a relationship between the old and new feature/defect,
4. Work all subsequent features/defects until done.
5. Update the original feature/defect and tell everyone that the change was really done.

This is a cumbersome process that TeamConnection has made obsolete.

IMPROVED COMMUNICATION BETWEEN TEAM MEMBERS

TeamConnection encourages better communication between team members by forcing active workareas to be refreshed from one to another when changing the same parts in a release.

Whenever team members need to make changes to the same parts there is contention for these parts. Even though the first person to check out a part and check it back into their workarea "wins" (that is, there is no need to take any special action in order to integrate their workarea into a driver), other team members can also update these parts before the workarea containing the part is committed.

The second team member that tries to check out the part to their workarea will get a message indicating that they must refresh from the first workarea in order to change the part (the first workarea is identified in the message) prior to checkout.

This refresh establishes an explicit prerequisite and tells the second developer whom to talk to (that is, the owner of the first workarea) in order to resolve issues like "when will your workarea be added to a driver?".

Note that this works particularly well when the release is in "serial" mode. In many cases, the customer sees TeamConnection's "parallel" mode and initially wants to use that mode for all the releases, believing it will allow team members to work more independently and quickly. However, once they see the way these notifications force communication, and how serial mode avoids the need for manually merging of changes, they generally choose serial mode as their preferred release process.

CMVC does little to encourage communication other than sending notifications to component owners (unchanged in TeamConnection) and identifying conflicts with Level -check.

SHARING PART CHANGES BEFORE PROMOTING

TeamConnection allows the sharing of parts that have changed but that have not been committed yet by using "teamc workarea -refresh".

This eliminates the confusion caused by the CMVC "current" version concept. Using TeamConnection's ability to refresh any workarea from any other workarea in a release, including the release itself or a driver, the CMVC concept of the "current" version of a part in a release is no longer needed.

Further, the issue of how best to manage a "current" version is eliminated. Each version of a part is associated with a specific workarea; when the workarea is integrated and a driver member is created, then that workarea IS the driver. When that driver is committed, the workarea IS the release. Therefore, each version of a part is clearly associated with a set of changes, not just the history of that part (which rarely provides much useful information). Further, workarea refreshes clearly define the prerequisites without external checking.

Finally, each team member can build the contents of their workarea without picking up any unwanted changes, as happens so often in CMVC when extracting the "current" release. As a result, multiple team members can simultaneously be working on changes that involve the same part without confusion, and without inadvertently using parts changed by others that CMVC would have included in the "current" pool.

CUSTOMER SCENARIO: MAINTAINING A REFERENCE DIRECTORY

Because a driver is also a workarea, it is possible to extract the contents of the driver and to list the part changes included in the driver.

For example, a customer built a process around CMVC that maintained a "reference directory" containing all of the changes in the tracks that had been added to an active level; then the developers built from this reference directory for their integration testing. However, it was not easy to determine if the reference directory was up to date, or even if it contained the right versions of files.

In contrast, when using TeamConnection, they have been able to alter their process to allow developers to add workareas to a driver, then let the tools update the contents of the reference directory. As a result, they know exactly what work is ready to be tested and that the

testing done in this reference directory is a valid integration test. Finally, they can build the driver using the TeamConnection build facility.

SAVING THE OUTPUTS OF A BUILD WITH THE BUILD FUNCTION

Since the TeamConnection build function checks the output files (that is, executables) back into the release, it is possible to recreate an old release simply by extracting those executables. As a result, upgrading systems with new operating systems, compilers, etc. does not prevent a release from being recreated exactly as it originally was.

The issue of saving too much data and running out of disk space in the family account is addressed though pruning options on the release.

ENCOURAGING MORE FREQUENT CHECKIN OF PARTS

Programmers, and other team members, often delay checking out, changing and checking in a part until they feel they are done with their changes (using extracted copies of the part). The most common reason for this is simple: no one wants someone else to see their incomplete work.

Unfortunately, this generally misleads other team members into believing that no one is interested in any particular part, since it is has not been updated in any active release. As a result, when a part is finally checked out it might be different than the version that was extracted and lots of manual merging of changes and retesting is necessary.

Once CMVC users hear about TeamConnection's concurrent release process, managers and team members believe that the collision resolution process and merge tool used by TeamConnection will address their problems. Actually, there is often an even easier way.

In TeamConnection, checking in a part merely replaces the previously stored copy of the part, eliminating it from the part history. In order to save a version, the workarea must be frozen. This means that if you check in a part 3 times with remarks and if you do not freeze the workarea, then only the remarks for the last check in will be kept and the remarks of the 2 previous stored copies are eliminated.

The result is that only the versions of parts that are worth saving will be preserved when the workarea is added to the driver. What this means to each team member is that they can check out, change and check in as much as they want without having their incomplete work saved and visible to others. On top of that, workareas (including their change history) can be manually or automatically pruned from the release when their information is no longer of use.

SURVIVING A SECURITY AUDIT

As with all large companies, someone does EDP (Electronic Data Processing) audits. In IBM's case, the standards they audit against are defined in a document called ITSC 201. In order for CMVC to meet the ITSC 201 requirements, many specific exemptions were required (for example, use of NFS mounts to OS/2 and Windows) and several specific CMVC options were required (for example, use of CMVC_NFS_DISABLE). As we have found out, IBM is not unique.

TeamConnection resolves these security issues:

- No processes should run as root.

  CMVC runs the "cmvcd" daemons with "root" authority in order to do NFS mounts, and then to perform the setuid() function so that the files extracted to the server are owned by a user and not by the CMVC family id.

  Since TeamConnection performs extracts across sockets, there is no need to setuid to another user and therefore no need for "teamcd" to be a root process. Thus, TeamConnection has no root/setuid processes.

- CMVC allows users to see the contents of the SCCS archives in the family accounts "vc" directory structure. This is required in order to perform extracts from a setuid process. The workaround is to not let anyone other than the family administrator log onto the system running CMVC, making the machine a dedicated CMVC server.

  TeamConnection stores all part data in the family database. Since TeamConnection also implements authentication for each row of data in every table, including through the use of the "teamc report" command, parts are not accessible through TeamConnection unless you are authorized.

  Further, since the ObjectStore databases are controlled by the family account and are only accessed by the TeamConnection daemons, there is no need to set permissions in such a way for another user on the system to have the ability to read the contents of that database. Therefore, there is no reason to prevent other users from logging onto the TeamConnection server machine in order to protect the part data.

- TeamConnection client will own all files it writes.

  As with Part -extract, all other -extract actions use the same socket mechanism to transfer data. This insures that the client will write and own all files, eliminating the need to set any permissions for sharing files.

- TeamConnection adds password protection on top of host/userid.

  The "trusted host/userid" mechanism can be defeated through malicious means. As a result, it is not always good enough to satisfy the requirements of an auditor. In those

cases, or merely because it could be more appealing to you, TeamConnection provides a password scheme.

This password scheme is secure because:

- The password is encrypted on the client (that is, no unencrypted password is ever passed across a socket).
- If the server goes down, or the user logs out, the "token" for that user expires.
- The password is entered on a separate prompt so that it does not show up in a "ps" command (to show the processes that are running).

APPENDIX A. BIBLIOGRAPHY

CMVC 2.3

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

SC09-1596-01 IBM CMVC Client Installation and Configuration
SC09-1597-01 IBM CMVC User's Reference
SC09-1631-02 IBM CMVC Server Administration and Installation
SC09-1633-00 IBM CMVC Concepts
SC09-1635-01 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
SC34-4501 TeamConnection, Commands Reference
SC34-4500 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

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

29.2147 SCLM Guide to TeamConnection Terminology
29.2196 Using REXX command files with TeamConnection MVS Build Scripts
29.2231 TeamConnection Interoperability with MVS and SCLM
29.2235 Using REXX command files with TeamConnection MVS Build Scripts for PL/I programs
29.2253 Comparison between CMVC 2.3 and TeamConnection 2
29.2254 Migrating from CMVC 2.3 to TeamConnection 2
29.2266 TeamConnection frequently asked questions: National Language Support (NLS) and Double-Byte Character Sets (DBCS)
29.2267 TeamConnection frequently asked questions: how to do routine operating system tasks

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 DB2/6000, DB2, CMVC VisualAge, TeamConnection, NetView	IBM Corporation
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, Visual Basic	Microsoft Corporation
X Window System	Massachusetts Institute of Technology
PVCS	InterSolv, Inc.

AFS	Transarc Corp
Intel	Intel Corp.

END OF DOCUMENT