Comparison between CMVC 2.3.1 and VisualAge TeamConnection Enterprise Server 3

Document Number TR 29.3096

Angel Rivera, Lee Perlov, John Suchy, Tim Orlowski and Sam Ruby

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

ABSTRACT

This technical report compares the functions of the latest version of CMVC 2.3.1 (Configuration Management and Version Control) and the latest version of its successor product, VisualAge TeamConnection Enterprise Server Version 3. 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 VisualAge TeamConnection.

This technical report provides the following appendixes:

- Differences between TeamConnection 2 and TeamConnection 3.
- Differences between CMVC95 and CMVC 2.3.1 and TeamConnection 3.

ITIRC KEYWORDS

- CMVC
- TeamConnection
- VisualAge TeamConnection

ABOUT THE AUTHORS

ANGEL RIVERA

Mr. Rivera is a Advisory Software Engineer in the VisualAge TeamConnection and CMVC development group. He joined IBM in 1989 and since then has worked in the development and support of library systems.

Mr. Rivera has an M.S. in Electrical Engineering from The University of Texas at Austin, and 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.

JOHN SUCHY

Mr. Suchy is a Software Engineer and the documentation team lead for the TeamConnection product. He has been a technical writer in the computer and telecommunication industries for 7 years.

Mr. Suchy received a B.A. from the University of Virginia in 1985 and is currently working on an M.S. in Technical Communication at North Carolina State University.

TIM ORLOWSKI

Mr. Orlowski is an Advisory Software Engineer in the VisualAge TeamConnection Development Group. He joined IBM in 1984.

Mr. Orlowski has an B.S. in Math/Computer Science from the University of Illinois.

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 latest version of CMVC 2.3.1 (Configuration Management and Version Control) with the latest version of its successor, VisualAge TeamConnection Enterprise Server Version 3. The focus will be on enhanced capabilities in TeamConnection derived from the addition of an object-oriented methodology and its object repository.

This TR is designed to provide enough information for current CMVC users to become productive VisualAge TeamConnection users quickly. 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. See “Comparison of common functions” on page 7.
- New VisualAge TeamConnection functions and differences affecting general users. See “New user functions of TeamConnection” on page 27.
- New VisualAge TeamConnection functions and differences affecting family administrators See “Changes that impact system and family administrators” on page 43.
- Customer feedback after migrating from CMVC to VisualAge TeamConnection. See “Customer feedback after migrating from CMVC to TeamConnection” on page 51.
- An appendix covering the differences between VisualAge TeamConnection Version 2 and VisualAge TeamConnection Enterprise Server Version 3. See Appendix A, “Differences between TeamConnection 2 and TeamConnection 3” on page 61.
- An appendix covering the differences between the internal product CMVC95 and the external products CMVC 2.3.1 and VisualAge TeamConnection 3. See Appendix B, “Differences between CMVC95 and CMVC 2.3.1 and TeamConnection” on page 67.

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 C, “Bibliography” on page 71.

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

- In CMVC, /usr/lpp/cmvc/samples/README.demo3 and demo3.tar.
- In VisualAge TeamConnection, /usr/teamc/samples/univunix. The univunix utility runs the TeamConnection commands in univunix.cmds that you can modify.

Because the CMVC servers are only available for Unix, there is a tendency in this TR to focus on the Unix capabilities of VisualAge TeamConnection.

To economize words in this TR, sometimes the term "TeamConnection" is used to refer to "VisualAge TeamConnection Enterprise Server".

ACKNOWLEDGEMENTS

We would like to thank all the people who gave us feedback when we were preparing this technical report, especially:

1. Clifford Meyers, who prepares the VisualAge TeamConnection installation files for Unix.
2. David Gauck, who prepares the VisualAge TeamConnection installation files for Intel.
3. Jim Palistrant, who interacts with our customers.
4. Robbie Wilderspin, who interacts with our customers in the United Kingdom.

We want to thank Dodde Stark for editing this technical report.

GETTING THIS TECHNICAL REPORT

The most up to date version of this technical report can be obtained from the IBM VisualAge TeamConnection Enterprise Server:

- Library home page by selecting the item Library at URL:

  http://www.software.ibm.com/ad/teamcon

- FTP site by accessing the URL:

  ftp://ftp.software.ibm.com/ps/products/teamconnection/papers

  See the file README.index.txt for details.

COMPATIBILITY BETWEEN CMVC AND VISUALAGE TEAMCONNECTION

VisualAge TeamConnection is the successor product to CMVC. VisualAge TeamConnection has evolved from CMVC 2.3 and has been extended to support object-oriented development.

The products have a lot in common; however, because of changes in VisualAge 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 how to prepare a CMVC family to be migrated to TeamConnection. (In **Chapter 6, Preparing to migrate to TeamConnection**).

The most up to date migration instructions are available in the softcopy VisualAge TeamConnection Installation Guide, in the (**Chapter 13 "Migrating to TeamConnection version 3**). The softcopy files, install.pdf, install.htm (master file), or install.txt are located in the following directories in the CD-ROMs for VisualAge TeamConnection:

```
softpubs/<lang> (for UNIX)
            or
nls\doc\<lang>  (for Intel)
```

THE BIG DIFFERENCES FOR USERS

Here are a few 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 7 and “New user functions of TeamConnection” on page 27.

- The names of several objects have changed. For example, the CMVC ***Tracks*** are now TeamConnection ***Workareas*** and CMVC ***Levels*** are now TeamConnection ***Drivers***. For more details see “Command names” on page 7.
- The versioning scheme is different:
  - CMVC Files are versioned by checkout/checking (file versioning); that is, the first time a file is created it has a version 1.1, then after the first checkin is 1.2, and so on.

    In contrast, TeamConnection Parts are versioned by the number of freezes in a workarea (system versioning); that is, the first time a part is created it has a version 1 based on the workarea, such as workarea1:1, then after the first checkin and explicit freeze is workarea1:2, and so on.
  - Version numbers in TeamConnection are named relative to the workarea, the driver and the release. For example, the part a.c can have a version number workarea1:3, or driver2:2 or release1:1. See “Version numbers of TeamConnection objects” on page 30 for more details.
  - The TeamConnection drivers and workareas are essentially specialized releases, in which they have a snapshot of all the versioned parts in a release.

For more details see “Versioning” on page 28

- Workareas are much more important than tracks and behave differently. For more details see “Workareas are more than renamed Tracks” on page 32.
  - A TeamConnection workarea needs to be specified in all "teamc part -checkin" and "teamc part -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 may periodically be required to link parts from other workareas (preferable) or 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 do a part link or a refresh from that workarea.
  - You can freeze a workarea in order to save intermediate changes of the changed parts in that workarea. However, only the last version of a workarea will be integrated to the release.
- When performing queries in TeamConnection, 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 a release, a driver, a workarea or a version. 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 20.
- A Web Client has been added in Version 3 to allow interacting with TeamConnection commands via a web browser. For more details see “New web client” on page 38.

THE BIG DIFFERENCES FOR FAMILY ADMINISTRATORS

Here are a few things that will be significant changes for CMVC Family Administrators migrating to VisualAge TeamConnection. There are more details in the sections “Family administration” on page 23 and “Changes that impact system and family administrators” on page 43

- The only database supported by VisualAge TeamConnection 3 is DB2 Universal Database (UDB) Version 5, and its installation files are bundled with the CD-ROMs for VisualAge TeamConnection. The installation utilities of TeamConnection will check if DB2 UDB is already installed in your system; if negative, then DB2 UDB will be installed in your system.
- In VisualAge TeamConnection all of the data is stored in the database, including files. The TeamConnection family administrator does not need to know SCCS in order to keep the database and file versions synchronized. In CMVC, in some rare cases it was necessary to know some SCCS commands in order to resynchronize the database and the file ver-

sions; for example, when the data was committed in the database, but due to a problem in the file system, the actual SCCS archive file was not created, and thus the database and file system become out of synch.

This also makes backup much easier, because in CMVC you need to backup both the database and the version control ($CMVC_FAMILY_HOME/vc) file system at the same time.

- Most of the administration commands have changed. See the **TeamConnection Administrator's Guide** for more details. Further, the TeamConnection Family Administration GUI (tcadmin) is provided for family administration.
- 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. For more details, see “Migration from CMVC to TeamConnection” on page 3.

THE BIG DIFFERENCES FOR PROJECT ADMINISTRATORS

The Lotus Notes federation with TeamConnection is a significant benefit to a large project because allows users to handle Requirement, Design and Test Case documents via Lotus Notes and still handle defects and features that are tracked in TeamConnection. For more details see “Lotus Notes integrated databases” on page 39.

The web interface enables a customer, with a little work to customize some html, to provide a very flexible 'helpdesk' system tracking all the way from the end user to development. For more details see “New web client” on page 38.

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 37 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 19 and “Object Repository and Information Model” on page 41 for more details.

- A Web Client that allows the interaction with TeamConnection commands via a web browser. This function was added in Version 3. For more details see “New web client” on page 38.
- A Lotus Notes interface that allows users to use Lotus Notes databases for requirements, test cases, development and other tasks (based on the generic template) to interact with TeamConnection features and defects. This function was added in Version 3. For more details see “Lotus Notes integrated databases” on page 39.
- The client in Version 3 supports the Source Code Control API to allow other applications to checkout and checkin parts. For more details see “Source Code Control API” on page 39.

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	tcstop	To reflect the name of the product
mkfamily, rmfamily, mkdb, rmdb, chfield, ch*	tcadmin	Consolidation of administration tools.
cmvccomp (OS/2)	tcmerge	Availability for all platforms.

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.

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). In CMVC, the notification daemons only supports the UNIX utility "sendmail". In TeamConnection, the notification daemon runs a script which can be customized, for example, to use a mechanism other than sendmail (such as Lotus Notes or Microsoft Exchange in Windows NT).

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 14 and “Features” on page 14) 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 27.

- 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 VisualAge TeamConnection Merge tool (tcmerge).

For more details see “Sequential and Concurrent development” on page 27.

- 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 or three parts and allows the user to manually merge the parts into one single part. You can select to view the differences and collisions, as well as view the composite output of the merged files.

- Versioning releases.

  The TeamConnection drivers and workareas are essentially specialized releases, in which they have a snapshot of all the versioned parts in a release. 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 28.

- 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 37.

- 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 “Pruning of releases” on page 27.

- Configurable fields.

  In TeamConnection Version 3 you can now add configurable fields to releases.

- Part shadowing.

  In TeamConnection Version 3 you can now exploit the part shadowing capabilities for releases. For details, see “Part shadowing capability” on page 39.

- Coupling (commonality of files between releases).

  It is possible to specify the type of coupling (commonality between the files) that a release has with other releases; this does not affect the commonality of files between workareas in

the same release. This is specified by the -coupling flag for release -create and release -modify:

- Default.

  You must use force to break links (same as VA TC V2 and CMVC).

- Loose.

  If a common file is checked in for one release and if the -common flag is not used, then the commonality will be broken. This is the same behavior as if using -force; in this coupling, -force is not necessary.

- LooseRestrict.

  A release cannot be kept common with any other release by the -common mechanism.

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.
- In TeamConnection it is possible to have a process in which the release could have the driver subprocess but not the track subprocess (this is not possible in CMVC), because a workarea is always needed in TeamConnection and these workareas can be placed in a driver. ;p. This is not recommended for production releases because there is no change history (defects/features) associated with the parts. In our experience, the customer sooner or later is going to ask why was this part changed? And without a change history it is not easy to find out the answer.

CHANGE CONTROL

Overview

Both CMVC and TeamConnection keep track of any file changes you make and the reasons you make them. Defects and features are used to identify the requirements for the file changes. After changes are made, you integrate the changes and build the application.

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.

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.

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.

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 8.

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 8.

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 28.

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 (unless this is the first change in the workarea).

  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.

- Part shadowing.

  In TeamConnection Version 3 you can now exploit the part shadowing capabilities for releases. For details, see “Part shadowing capability” on page 39.

  See next bullet item for additional information about the CR/LF conversion.

- In CMVC, all text files were normalized to the Unix standard: only use line-feed or LF between each line, and delete the carriage-return or CR. Thus, only when using Intel workstations it is necessary to use the -crlf flag during a file extract or check out to convert the LF to the CR/LF pair.

  In contrast, in TeamConnection, files are checked in "as is", that is, they are not normalized to the Unix standard. By default, the text files are not normalized during extract or checkout. In order to normalize the text files to match the appropriate standard of the client, the -crlf flag should be used when extracting or checking out a file/part.

  However, in part shadowing, the flag -crlf means to explicitly add the CR/LF pair to all text while not using -crlf means to add only LF.

- 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 28 and “Parts are more than just Files” on page 30.
- 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 different syntax for the expandable keywords.
- The TeamConnection Part object uses two attributes to retain important date information:
  - Last update.

    This is the timestamp when the part was last changed.
  - File system timestamp.

    During check-in, the date and time of a part from the file system is stored in this attribute.

  In TeamConnection, during the extraction of a part, the file system timestamp will be used; this means that the part will have the same date and time that the part 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 32.

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 32.
- TeamConnection adds 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 33 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 95 (when the users bypass the login screen), 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 35.

ARCHITECTURAL ISSUES

Client/Server Design

Both CMVC and VisualAge TeamConnection have a client/server architecture, in that the client communicates 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, which is specified in the GUI's settings pages.

Differences

- One single executable file with all client commands.

  CMVC has one executable file for each command, such as Report. In contrast, VisualAge TeamConnection has a single executable "teamc" which contains all functions for specific command such as "teamc report".

  This approach allows to have the same name for Unix and Intel commands. For example, in CMVC, the Unix command "LevelMember" had to be renamed in Intel as "levelmem".

- Object Repository API.

  TeamConnection adds another 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 41 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.
  - A help text can be defined, which can be displayed from the GUI.
- Configurable fields

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

  - Defects
  - Features
  - Users
  - Files (CMVC) or Parts (TeamConnection)
  - Releases (only in TeamConnection)
  - Workareas (only in TeamConnection).

  For the family administrator there are several small changes. See “Family administration” on page 23 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 44.

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; however, because Softbench has been withdrawn from the market, then VisualAge TeamConnection is not integrated to Softbench.
- In TeamConnection Version 3, an interface was added for Lotus Notes, in which you can use Lotus Notes to manage requirements, test cases or development, and then manage the corresponding TeamConnection features or defects.

- In TeamConnection Version 3, an interface was added for Source Code Control (SCC) API, in which other applications such as PowerBuilder can checkout and checkin parts from TeamConnection.
- In TeamConnection, API's are provided to access TeamConnection data through an Object Repository (see “Object Repository and Information Model” on page 41). 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

CMVC and VisualAge TeamConnection Version 3 simply pass thru the SQL queries to the database management system (DBMS) of the relational database and the DBMS responds to the queries.

Differences

- 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.

To find our more details on the parameters and configurable fields that a family administrator can use in the user exits, a superuser can issue the command:

```
teamc report -userExitInfo
```

An example of the output is shown below:

```
Action Name: DefectModify
 User Exits 0 and 3:
  Parameter List:   defectname, newdefectname, severity, environmentname,
       prefix, reference, drivername, abstract,
       originator, answer, remarks, release,
       configFields, MessageBuffer, notesDB,
       notesID, effectiveUserID, TeamcUserID,
       VerboseFlag
 User Exit 1:
  Parameter List:   defectname, newdefectname, severity, environmentname,
       prefix, reference, drivername, abstract,
```

```
       originator, answer, remarks, release,
       configFields, dateoflastupdate, notesDB,
       notesID, effectiveUserID, VerboseFlag
 User Exit 2:
  Parameter List:   defectname, newdefectname, severity, environmentname,
       prefix, reference, drivername, abstract,
       originator, answer, remarks, release,
       configFields, notesDB, notesID, effectiveUserID,
       VerboseFlag
 Configurable Fields:   symptom, phaseFound, phaseInject, priority,
       target, pubsImpact, guiImpact, duration,
       solution, customerName, pmrNumber, testImpact,
       installImpact, testerlogin
```

This command does not give the list of existing user exits for the family.

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 3.

CMVC provides utilities to migrate from SCCS to CMVC.

Year 2000 readiness

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 is possible that the year 2000 might not be sorted properly.

VisualAge TeamConnection and CMVC 2.3.1 use 4 digits to store the information about the year. Thus, VisualAge TeamConnection and CMVC 2.3.1 are fully ready for the Year 2000.

SUPPORTED PLATFORMS AND DATABASES

Family server

Both products support family servers in the following platforms:

- AIX Version 4.2.1
- HP-UX Version 10.20
- Solaris Version 2.5.1

TeamConnection adds the following server platforms:

- OS/2 Warp
- Windows NT

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

- AIX Version 3.2.5
- HP-UX Version 9
- SunOS 4.1.3

Clients

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

- AIX Version 4.2.1
- HP-UX Version 10.20
- Solaris Version 2.5.1
- OS/2 Warp
- Windows 95 and Windows NT

Notes:

In VisualAge TeamConnection Version 3, there is no longer support for Windows 3.1.

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

- AIX Version 3.2.5
- HP-UX Version 9
- SunOS 4.1.3

Build Servers

The build function is only supported by TeamConnection (see “Build support” on page 37) and the supported platforms are:

- The same platforms for the TeamConnection family server, “Family server” on page 21.
- MVS OS/390 and MVS/OE (Open Edition).

Databases

CMVC supports the following relational databases:

- DB2 Versions 1 and 2
- DB2 Universal Database (UDB) Version 5
- Oracle Version 7.x (including 7.3)
- Informix Version 5 and 7
- Sybase Version 4.9, 10 and 11

The only database supported by VisualAge TeamConnection 3 is DB2 Universal Database (UDB) Version 5, and its installation files are bundled with the CD-ROMs for VisualAge TeamConnection. The installation utilities of TeamConnection will check if DB2 UDB is already installed in your system; if negative, then DB2 UDB will be installed in your system.

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 44.

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 and in Figure 3; 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/        * Definition of the configurable fields
maps/               * The driver information is stored in the database
                      in TeamConnection
queue/messages      * Separated addressing information and contents.
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: only one file
cfgField/           * Definition of the configurable fields
config/             * Place for the Security and userExit files
queue/              * Messages stored in one file in one directory
                    * The addresses are in the i* files and
                    * the contents are in the m* files.
tctmp/              * Temporary 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 25.

CMVC	TeamConnection
authority.ld	authorit.ld
cfgcomproc.ld	comproc.ld
cfgrelproc.ld	relproc.ld
configField/	cfgField/
queue/messages	queue/
maps	REMOVED
vc	REMOVED
<none>	tctmp/

Figure 4. Changes from CMVC to VisualAge 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 43.

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. 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 the VisualAge TeamConnection Merge tool (tcmerge) 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 can be specified at the time a release is created. The mode can be changed from serial to concurrent, but it cannot be changed from concurrent to serial.

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

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.

There are two ways to accomplish the pruning:

- Manual.

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

- Automatic.

  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". See “TeamConnection 1, useful to TeamConnection 3 users” on page 71.

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.). See “Version numbers of TeamConnection objects” on page 30 for more details.

The releases are the objects that are versioned, which means that the drivers, workareas and parts are also 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. For more details, see “Workareas are more than renamed Tracks” on page 32.

Versioning in CMVC

CMVC uses the Unix utility SCCS as the underlying mechanism for the versioning of the files. The product PVCS is only used with CMVC on AIX 3.x (which is now discontinued).

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.
- In PVCS all files are stored as forward deltas by PVCS.

One drawback of 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.
- The release versions use reverse deltas.
- The file versions within a workarea and driver use forward deltas.

You can change the type of the part, from text to binary and vice versa, by checking it out and then checking it in and specifying the new type.

For more information about this topic, consult the Versioning Model in Page 82, Chapter 2 "TeamConnection", in the manual "Introduction to the IBM Application Development Team Suite". See “TeamConnection 1, useful to TeamConnection 3 users” on page 71.

Version numbers 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.

Furthermore, in TeamConnection there is commonality between workareas when a part is linked or refreshed from another workarea in the same release.

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. You can change the type of the part, from text to binary and vice versa, by checking it out and then checking it in and specifying the new type.

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 '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 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.

Miscellaneous features

- Configurable fields.

  In TeamConnection Version 3 you can now add configurable fields to workareas.

- Part shadowing.

  In TeamConnection Version 3 you can now exploit the part shadowing capabilities for workareas. For details, see “Part shadowing capability” on page 39.

- Extraction.

  In TeamConnection Version 3 you can now extract the parts associated with a workarea (full or delta).

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 VISUALAGE TEAMCONNECTION GUI

The VisualAge TeamConnection GUI has evolved significantly from the original CMVC GUI for Unix. Actually, some of the new features in the TeamConnection GUI were added to the CMVC GUIs for OS/2 and Windows.

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 V3 Intel GUI with the CMVC version 2.3.0.18 GUI for Intel.

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.
- 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 in the client 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:

- HOST_ONLY: only use host entries, which is the default.

  This is the only option in CMVC.

- PASSWORD_ONLY: only use the method of login with password.
- PASSWORD_OR_HOST: either a login with password or a host entry.
- NONE: no authentication method at all.

  This method should be used with extreme care because every user has the potential to become a superuser.

  This method should be used as the last resource in emergency situations, such as when migrating a family from one host to another and no provisions were made (such as adding a new host list entry for the new host) before the migration. In this case, shutdown the family, change the authentication method for the new family to NONE, restart the family, add the proper new host list entry, shutdown the family, change the authentication method to a more restrictive type, and restart the family.

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 than 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 workstation.
- It is easier for a family administrator to temporarily disable a user account with passwords than deleting a user's host list entries.

Note: The authentication type can be changed at any time by using the tcadmin utility or by modifying the file $FAMILY_HOME/config/security. It is necessary to restart the family in order for the new teamcd daemons to use the new authentication type.

PLANNING FOR TEAMC RELEASE/DRIVER/WORKAREA -EXTRACT ACTIONS

When doing a teamc release/driver/workarea extract, TeamConnection 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 (like with CMVC). 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.

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

Another option would be to issue a 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, Solaris, OS/2, Windows NT, Windows 95, MVS OS/390 and MVS/OE. 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 Tivoli Software Distribution facility.

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.

NEW WEB CLIENT

The Web Client is a TeamConnection Client that provides a subset of the functions available in the Intel or UNIX client. The name Web Client is a slightly misleading name, because it really is a specialized Web Server built into the TeamConnection family server.

The Web Client provides access to a TeamConnection family from a frame-enabled web browser (such as Netscape Navigator or recent versions of Microsoft Internet Explorer), without having to install any TeamConnection components on the client system.

The function missing from the GA version of the Web Client is the ability to manipulate part contents, that is, it is not possible to check-in, check-out, create or extract parts, although it is possible to view part contents.

Note: This ability will be added to the code in future fixpaks.

To access a TeamConnection family using the Web Client, all that is required is to know the hostname of the family server (or its IP dotted decimal address) and the port number of the family. The URL to be used is:

```
http://hostname:port
```

To disable the Web client, set the following environment variable in the TeamConnection server:

```
TC_WWWDISABLED
```

LOTUS NOTES INTEGRATED DATABASES

The VisualAge TeamConnection Integrated Notes Database feature provides a documentation facility to support software development. A software development group can use this database to communicate with TeamConnection objects from within Lotus Notes documents. The TeamConnection Integrated Notes Database links the technical documents to the TeamConnection objects involved in software enhancement and maintenance.

A single database template is provided that you use to define the database that will assist you with all phases of your development process. From the template, databases can be produced that can assist with requirements tracking, design and development documentation, and test case management, as well as other purposes.

This function was added in Version 3.

SOURCE CODE CONTROL API

Source Code Control (SCC) API, in which other applications such as PowerBuilder, Visual Basic, Visual C++ and Content Manager can checkout and checkin parts from TeamConnection, among other functions.

Any tool that this SCC API should work; however, the ones listed above are the only ones officially supported

This support was added in Version 3.

PART SHADOWING CAPABILITY

TeamConnection now includes support for shadows. A shadow is a collection of parts in a file system that reflects the contents of a workarea, driver, or release. You can use shadows to build your product, or to provide a place outside the library where developers can go to search through code. Shadowing is similar to extracting in that the purpose of each is to provide a set of objects from a TeamConnection version or version context.

In part shadowing it is important to understand the following characteristic with respect to the CR/LF conversion: the flag -crlf means to explicitly add the CR/LF pair to all text (for Intel) while not using -crlf means to add only LF (for Unix). For more details, see the differences subsection in “Files (CMVC) or Parts (TeamConnection)” on page 15.

This support was added in Version 3.

VISUALAGE TEAMCONNECTION FOR SMALLTALK BRIDGE

VisualAge TeamConnection for Smalltalk provides a repository with operational support tailored specifically for highly-interactive, prototyping environments that emphasize iterative development, such as VisualAge Smalltalk or VisualAge Generator.

A bridge from VisualAge TeamConnection Enterprise Server Version 3 to VisualAge TeamConnection for Smalltalk provides access to the software configuration management (SCM) support provided by VisualAge TeamConnection for Smalltalk along with the scalable, enterprise-level support provided by VisualAge TeamConnection Enterprise Server's ability to manage all development artifacts (not just source code), to share information in a common model, and to integrate multiple tools and multiple languages across the enterprise on a single baseline that extends the capabilities of software development groups.

This bridge provides essential integration for VisualAge tools which use VisualAge TeamConnection for Smalltalk as their day-to-day operational library.

The bridge supports the import and export of VisualAge Generator objects (parts) to an from VisualAge TeamConnection Enterprise Server.

VisualAge TeamConnection Enterprise Server can be used to manage artifacts (parts) that need to be shared with languages that are not based on a universal virtual machine or tools for purposes of build management, problem tracking and other configuration management functions. These artifacts can be exported from VisualAge TeamConnection for Smalltalk to VisualAge TeamConnection Enterprise Server through the bridge and stored as TeamConnection parts.

Objects stored in a TeamConnection database can be queried and retrieved back into the VisualAge TeamConnection for Smalltalk development environment as needed. The units of storage in TeamConnection include exported VisualAge for Smalltalk and VisualAge Generator components (such as applications and configuration maps) and large grained objects (files). Small-grained objects, such as VisualAge Generator data items, are imported and exported as constituents of applications. The data items in an application are exported to VisualAge TeamConnection Enterprise Server in a form that makes their definitions available to other tools through the data model.

PART/RELEASE MERGE FUNCTION

TeamConnection provides a merge function to the part and release commands. Parts and releases from multiple sources can now be merged into a new single source.

In addition, an Automerge tool has been introduced to provide automatic merging of files. In

most cases, the user needs only to resolve conflicts. The actual "script writing" and execution for merging of files is now automatic.

This support was added in Version 3.

TRANSLATION SUPPORT ATTRIBUTES FOR THE PART COMMAND

The following fields related to the translation of parts related to National Language Support are provided to designate the translatability of parts for the actions part -create, part -modify, and part -mark:

- translation

  This field indicates if the part is part of the translation process.

- translation state

  This field indicates if the part is ready or notReady to be translated.

New versions of a part have an initial translation state of notReady. The most recently committed part version is marked by default as notReady.

The transition of a part to the ready state is not automatic, and it must be done manually with the part -mark command. No translation states are currently defined for parts in other languages that are the result of translation.

This support was added in Version 3.

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

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: name, location, port number, mail exit,
- Initial superuser and its password
- Security: authentication level, minimum password size, maximum number of invalid attempts.
- Configurable types (config.ld)
- Configurable fields and their report and table formats.
- Release processes
- Component processes
- User exits
- Authority groups
- Interest groups

Notes:

1. Security (Authentication Level).

   For more information, see “Password login to TeamConnection” on page 35.

2. Configurable fields

   In CMVC, chfield actually interacts with the database and validates the configurable field definition files in configField. In contrast, in TeamConnection, tcadmin 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". Furthermore, 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 43.

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.

For user exits that modify the contents of files as they are extracted, checked in, etc. there is one minor change with respect to the CR/LF pair (Intel standard carriage-return/line-feed): in CMVC the text files were stored normalized to the Unix standard (only LF), while in TeamConnection, the text files are not normalized.

As a result, the temporary file on the server that is accessible by a user exit may contain CR/LF pairs or just LF between lines. For more details see “Files (CMVC) or Parts (TeamConnection)” on page 15.

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

The family administrator has to use the DB2 facilities for backup and restore of the database.

There is more documentation on backup and recovery in the VisualAge TeamConnection Administrator's Guide and the DB2 Administration Guide manuals.

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.

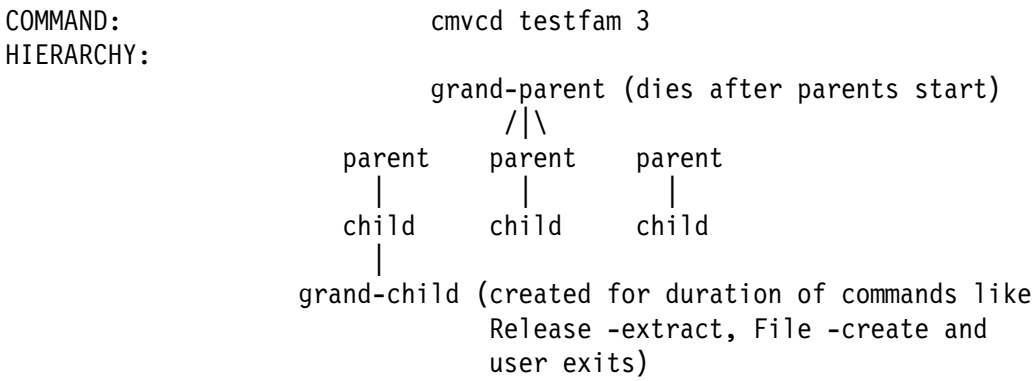

Figure 5. CMVC daemon process hierarchy

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 server daemon 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 47.

```
COMMAND:   teamcd -a buildsystem@2000 -p pool1 -e HP -s 3 -n testfam 3
HIERARCHY:
         parent ----+---------+------> build server
           |         \         \       (possibly on
         child     child     child      a different
           |                             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 tcstop 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 or TMPDIR 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 and $FAMILY_HOME/tctmp on a per process basis.

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 file system. This allows you to overcome most file system 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. The exception is the set of *.ld files (which contents is loaded into the database by tcadmin) and the contents in the bin, config and cfgField directories in the family HOME directory.

The "bulk data" table resides in its own table space. Thus, it has the potential for expansion by adding more devices.

Space used by family daemons

The TeamConnection temporary directory, /tmp and $FAMILY_HOME/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 TeamConnection and 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.

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

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 AFTER MIGRATING FROM CMVC TO TEAMCONNECTION

Now that there has been time for customers to move to VisualAge TeamConnection (both Version 2 and Version 3), 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:".

AUTOMATIC THE SOFTWARE BUILD AND DISTRIBUTION PROCESSES

The VisualAge TeamConnection build facility is a general purpose facility. When used with the shadow facility and a Web page, the software build and distribution processes can be automated and can be managed easily.

For example, we wanted to take full advantage of VisualAge TeamConnection by adding a few things to the family setup to ensure that a given application would build reliably and automatically. Also, we wanted to track the results of the build.

In addition, once we built the application, we needed to distribute it to team member; some of whom did not have access to our VisualAge TeamConnection family. In order to always distribute the proper version of the application, we shadow the committed release to a directory on the server and use a web page to point to the shadow. This allows anyone with access to our web page to quickly select the files that compose our application and then to download them into the client workstation.

Additionally, the web page used provides customized access to the VisualAge TeamConnection family through the VisualAge TeamConnection Web interface. Included in this customized web page is a reference to a second web page the reports the activity of the family server. Although this is not directly related to building our application, it is useful information for family administrators who monitor the build.

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 allowing them to link parts from other workareas or by refreshing a workareas from another when changing the same parts in a release. The linking of parts is preferable than the refreshing of workareas because the refreshing action will get all the changed versions from the source workarea and this may have unexpected side effects.

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 link the part or refresh from the first workarea in order to change the part (the first workarea is identified in the message) prior to checkout.

After the second team member checks in the part that as linked from another workarea, then the original workarea becomes a prerequisite. Thus, the second developer will need to talk to 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 "concurrent" (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; the main drawback is that when a part is checked out by several developers at the same time, then there is a lot of time involved in resolving the collisions (conflicts) when merging the changes during the checkin process.

However, once these customers see the way these notifications and prerequisites force communication, and how serial mode reduces the need for manually merging of changes, these customers 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 part -link" (preferable) or "teamc workarea -refresh".

This eliminates the confusion caused by the CMVC "current" version concept. Using TeamConnection's ability to do a part -link or 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).

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 DB2 UDB 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.

- 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. DIFFERENCES BETWEEN TEAMCONNECTION 2 AND TEAMCONNECTION 3

This appendix provides a summary of the differences between VisualAge TeamConnection Version 2 2 and VisualAge TeamConnection Enterprise Server Version 3. For more details, see the files "readme.txt" and "relnotes.htm" in the CD-ROM for VisualAge TeamConnection.

MAIN DIFFERENCES BETWEEN TEAMCONNECTION 2 AND 3

The main differences between VisualAge TeamConnection Version 2 and Version 3 are listed below:

- Due to limitations in scalability, the database management system was replaced from ObjectStore (in Version 2) to DB2 Universal Database Version 5 (in Version 3).
- Because of the change in database management system, in Version 3 you cannot longer create a separate database for each release.
- The Solaris platform is now supported in Version 3 for the server and the client.
- A Web Client has been added in Version 3 to allow interacting with TeamConnection commands via a web browser. For more details see “New web client” on page 38.
- In TeamConnection Version 3, an interface was added for Lotus Notes, in which you can use Lotus Notes to manage requirements, test cases or development, and then manage the corresponding TeamConnection features or defects. For more details see “Lotus Notes integrated databases” on page 39.
- In TeamConnection Version 3, an interface was added for Source Code Control (SCC) API, in which other applications such as PowerBuilder can checkout and checkin parts from TeamConnection. For more details see “Source Code Control API” on page 39.
- The tcmerge utility in Version 2 was available only on Intel. However, in Version 3, the tcmerge utility was rewritten in Java and thus, it is available in all our supported platforms.

  Furthermore, the paradigm of the tool was changed from 2-file merge into 3-file merge in order to support concurrent development.
- The tcadmin utility in Version 2 was available only on Intel. Now in Version 3, the tcadmin utility was rewritten in Java and thus, it is available in all our supported platforms.

  It now includes a utility that monitors family usage and has an expanded user exit support.
- Bridge for VisualAge TeamConnection for Smalltalk. For more details see “VisualAge TeamConnection for Smalltalk bridge” on page 40.
- Tivoli Software Distribution Packaging support

The Tivoli Software Distribution (TSD) packaging tool supports automated distribution between a single Tivoli Software Distribution server and Tivoli-managed clients.

- Part shadowing into a file system. For more details see “Part shadowing capability” on page 39.
- Configurable Fields updates in Version 3.
  - Family Administrators can now define configurable field types that allow users to specify multiple values for a configurable field or require users to enter a value that meets specific criteria, such as a 6-digit number or a 2-character string. They can also define general help text for configurable field types and specific help text for each value defined for a configurable field type.
  - It is possible to add extra configurable fields to the following objects:
    - Releases.
    - Workareas.
  - Parts: propagate values when doing a -link action.
- tcstop utility to stop the TeamConnection family daemons

  A new utility, tcstop.exe, has been provided to stop any daemons associated with a TeamConnection family. The syntax is as follows:

  ```
  tcstop Family
  ```
- Part/Release Merge function. For more details see “Part/Release Merge function” on page 40.
- Translation support attributes for the Part command For more details see “Translation support attributes for the Part command” on page 41.
- The new build server in Version 3 replaces the build agent and the build processor in Version 2.
- New build server for MVS and MVS/OE. It uses a single build server component to build OS/390 applications.

MISCELLANEOUS DIFFERENCES BETWEEN TEAMCONNECTION 2 AND 3

- The TeamConnection report command in Version 2 checks the access lists for a user and returns only the rows of data that a user is authorized to see. This function is planned to be added back to Version 3.
- A new shipped authority group called "admin" has been added. It includes User and Host actions that were formerly only allowed by a superuser, including the following actions:
  - UserCreate

  - UserDelete
  - UserModify
  - UserRecreate
  - UserView
  - HostCreate
  - HostDelete

  This new authority group can help to reduce the number of superusers because previously, some customers had to give superuser authority to team leaders in order to create users and host lists.

- New environment variables have been added:
  - The environment variable TC_MODPERM controls whether the file attribute of a file on a workstation is changed to read-only or not after a teamc part -create, -checkin, or -unlock command. If it equals off or OFF, the file attribute will not be changed.
  - The environment variable TC_BACKUP controls whether a backup file of a file on a workstation is created during a teamc part -checkout or part -extract command. If it equals off or OFF, the file attribute will not be changed.
  - The environment variable TC_CATALOG allows the specification of a message catalog file. With this variable, the exact location of a message catalog can be specified to a TeamConnection daemon.
- Users can use the keyword "null" to change to null (0 characters) certain fields, such as the release name in Defects, and configurable fields.
- "hold" attributes have been added for workarea state transitions. Use of these attributes will cause the current state of the workarea to be retained when processing the indicated commands, rather than the state moving to the normal next state. This change affects work area fix, test, and commit.

  For example, you can create a new release process with all the possible subprocesses in $HOME/relproc.ld or via tcadmin:

```
track_all|approval
track_all|fix
track_all|driver
track_all|test
track_all|track
track_all|trackfixhold
track_all|trackcommithold
track_all|tracktestthold
```

- Users can receive customized messages when they attempt to run commands that are not "read-only" against a Team Connection family that is running in maintenance mode.
- E-mail addresses are treated by TeamConnection as case sensitive.
- An abstract has been added to notifications for all workarea related actions.

- New user exit parameters have been added and documented.
- A message buffer is now available to user exits on errors. The message buffer (msgBuff) will always be null for exit point 0.
- The "+" and "-" characters can be used as the first character of a parameter value. The + or - must be doubled in a command to differentiate between the parameter value and a new flag. Only one + or - will be placed in the database for the value.
- Users can have separate INI resource files to be used with teamcgui for each of the families they work with.
- The action "teamc report -testServer" can now return to users the names and values of certain environment variables that are set on the Team Connection server.
- The action "teamc report -general" was added to allow users to customize their reports.
- The action "teamc workarea -extract" was added to allow users to extract only those files changed for specified workareas. In addition to a delta extract, you can do a full extract of parts in the workarea.
- The action "teamc driver -extract" allows a full extract of an uncommitted driver.
- The actions "teamc test -create" and "teamc test -delete" were added to explicitly create and delete test records.
- An -ignore flag has been added to the action "teamc driver -commit" to ignore any missing prerequisites that are identified during the prerequisite and corequisite checking phase.

  Warning

  We consider this feature to be potentially dangerous to the consistency of your baselines. Thus, we suggest that until you fully understand all the ramifications of ignoring the prerequisites when committing a driver, you should NOT give authority to the following action, even to the projectlead authority group (see authorit.ld file), which has it by default:

  ```
  DriverCommitIgn
  ```

- A -driver flag has been added to the action "teamc driver -check". This change essentially performs the action "teamc workarea -check -driver" for each workarea in the driver being checked; this action determines the requisites of the workarea as if the specified committed driver was the last committed driver.
- An exclude flag has been added to workarea -check to list work areas that should be removed from the input list to satisfy any prereq conditions.
- User configurable fields have been added to Part -create.
- In Version 3, the attributes "answer" and "release" have been added for Feature.

The answer field is required on the feature -accept and feature -return commands as it is for Defects. The release field is optional on the feature -open command and holds the release name, as it does for Defects.

- Ownership of an "original" verification record of a defect or feature will be changed when the originator (originlogin) field of the defect or feature itself is modified. The "duplicate" verification records are not affected.
- Support for list and regular expression configuration "kinds" has been added in the configurable field types (config.ld).

APPENDIX B. DIFFERENCES BETWEEN CMVC95 AND CMVC 2.3.1 AND TEAMCONNECTION

The product CMVC 2.3 is available for customers outside and inside IBM. In contrast, the product CMVC95 is available only to the internal community in IBM.

CMVC95 was produced in order to provide a quicker mechanism to deliver features requested by the internal community in IBM. Because the CMVC95 users are also part of the target audience for this TR, this appendix contains the listing of the main differences between CMVC95 and CMVC 2.3.1 and TeamConnection.

Many of these features have been incorporated into VisualAge TeamConnection Enterprise Server V3. For details see “CMVC95 features that are in TeamConnection 3, but not in CMVC 2.3.1.”

However, there are some features in CMVC95 that have not been incorporated into VisualAge TeamConnection or CMVC 2.3.1. For details see “CMVC95 features that are not in TeamConnection 3 or in CMVC 2.3.1” on page 69.

CMVC95 FEATURES THAT ARE IN TEAMCONNECTION 3, BUT NOT IN CMVC 2.3.1

The following CMVC95 features have been incorporated into VisualAge TeamConnection Enterprise Server V3, but are not part of CMVC 2.3.1:

- Allow use of + or - as first character of an input parameter in a command.
- Allow multiple INI resource files for the teamcgui to better handle the tasks across multiple families.
- Provide extended support in Report -testServer by returning extra environment variables defined by the family server.
- In the monitor, show an indicator of the processing phase.
- Control whether or not, the configurable fields can be modified by other than the owner.
- Add answer and release as fixed fields for Features.
- Provide "hold" attributes for the state transitions for workareas and drivers.

  For example, you can create a new release process with all the possible subprocesses in $HOME/relproc.ld or via tcadmin:

```
track_all|approval
track_all|fix
track_all|driver
track_all|test
track_all|track
track_all|trackfixhold
track_all|trackcommithold
track_all|tracktestthold
```

- Provide loosely coupled releases.
- Display a complete list of common releases during the checkin of a common file. In CMVC, only one common release is displayed at a time, which is cumbersome when a file is common to many releases.
- Add "admin" authority group to allow non-superusers to be able to create, delete and modify users and host lists.
- Customize message to be displayed to the users when the family server is in maintenance mode.
- Add a base driver parameter to Driver -check.
- Users can use the keyword "null" to change to null (0 characters) certain fields, such as the release name in Defects.
- The following environment variables have been added:
  - The environment variable TC_MODPERM controls whether or not the file permissions are changed to "read-only" after a checkin.
  - The environment variable TC_BACKUP controls the creation of backup files in the workstation when doing an extract or checkout.
  - The environment variable TC_CATALOG specifies the location of the message catalog.
- The following commands and actions from CMVC95 have been added to VisualAge TeamConnection Version 3:
  - File/Part configurable fields for -create and -link actions.
  - Level -build (Driver -restrict) action.
  - Prereq command.
  - Release -configInfo action.
  - Release -extract Component action.
  - Release configurable fields for -create and -modify actions.
  - Test -create action.
  - Test -delete action.
  - Track/Workarea configurable fields for -create and -modify actions.
  - Track/Workarea -configInfo action.
  - Track/Workarea -check (exclude, full) action.
- The following commands and actions from CMVC95 have been added to both VisualAge TeamConnection Version 3 and CMVC 2.3.1:
  - The Report -general command to allow users to get customized reports.

This function was also implemented in CMVC 2.3.1, in order to allow the migration of a CMVC 2.3 family to VisualAge TeamConnection Version 3.

CMVC95 FEATURES THAT ARE NOT IN TEAMCONNECTION 3 OR IN CMVC 2.3.1

The following CMVC95 features have not been incorporated into VisualAge TeamConnection Enterprise Server V3 or into CMVC 2.3.1:

- Become command (DCE authentication).
- Defect -recall action.
- Defect -transfer action.
- Family command.
- File -promote action.
- Level -extract Package action.
- Package command.
- Release -create (type, parentRel, parentLvl) action.
- Release -extract Package action.
- Release -modify parentLvl action.
- Report -exec action.
- Track -change action.
- Track -extract Package action.
- User DCE authentication fields for -create and -modify actions.

APPENDIX C. 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
```

For the latest list of updates to the publications, see the file:

```
/usr/lpp/cmvc/doc/README.pubs
```

VISUALAGE TEAMCONNECTION ENTERPRISE SERVER 3

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

```
SC34-4551  VisualAge TeamConnection, Administrator's Guide
SC34-4552  Getting Started with the TeamConnection Clients
SC34-4499  VisualAge TeamConnection, User's Guide
SC34-4501  VisualAge TeamConnection, Commands Reference
SC34-4500  VisualAge TeamConnection, Quick Commands Reference
SC34-4558  Staying on Track with VisualAge TeamConnection Processes (Poster)
```

TEAMCONNECTION 1, USEFUL TO TEAMCONNECTION 3 USERS

The following publication from TeamConnection 1 that has not been updated for VisualAge TeamConnection 3 is still useful:

```
SG24-4648  Introduction to the IBM Application Development Team Suite
```

TECHNICAL REPORTS ABOUT VISUALAGE TEAMCONNECTION

There are several technical reports that are available from the IBM VisualAge TeamConnection Enterprise Server Library home page by selecting the item Library at URL:

http://www.software.ibm.com/ad/teamcon

A partial list of these technical reports is shown below (for the complete and most up to date version, see the above URL).

- VisualAge TeamConnection Version 3: how to do routine operating system and DB2 tasks.
- VisualAge TeamConnection Version 3: NLS and DBCS.

DB2 UNIVERSAL DATABASE

For a complete and up-to-date source of DB2 documentation, use the DB2 Product and Service Technical Library, in English only, on the World Wide Web at: http://www.software.ibm.com/data/db2/library

APPENDIX D. 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
PDF, Acrobat Reader	Adobe Systems Incorporated
NetLS, Network Licensing System, iForLS, iFor	Gradient
HP-UX, SoftBench	Hewlett-Packard Company
AIX, OS/2, IBM, MVS DB2, DB2 UDB, Universal Database VisualAge, TeamConnection, CMVC VisualAge Generator	IBM Corporation
INFORMIX	Informix Inc.
Intel	Intel Corp.
PVCS	InterSolv, Inc.
Lotus Notes	Lotus Development Corporation
X Window System	Massachusetts Institute of Technology
Windows, Windows NT, Visual Basic Visual C++	Microsoft Corporation
Netscape	Netscape Communications Corp.
ObjectStore	Object Design, Inc.
OSF, Motif, DFS, DCE	Open Software Foundation, Inc.

ORACLE	Oracle Corp.
PowerBuilder	PowerSoft Corporation.
InstallShield	Stirling Technologies, Inc.
NFS, SunOS, Solaris Java	Sun Microsystems Inc.
SYBASE	Sybase Inc.
Tivoli	Tivoli Systems, a subsidiary of IBM
AFS	Transarc Corp
Unix	X/Open Co. Ltd.

END OF DOCUMENT