

IBM Configuration Management Version Control

SC09-1631-02

## **Server Administration and Installation**

Version 2 Release 3





IBM Configuration Management Version Control

SC09-1631-02

## **Server Administration and Installation**

Version 2 Release 3

**Note**

Before using this document, read the general information under "Notices" on page ix.

## **Second Edition (December 1994)**

This edition applies to Version 2 Release 3 of IBM Configuration Management Version Control/6000, program number 5765-207; IBM Configuration Management Version Control for Sun systems, program number 5622-063; IBM Configuration Management Version Control for HP systems, program number 5765-202; IBM Configuration Management Version Control for Solaris systems, program number 5765-397; and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Order publications by phone or fax. The IBM Software Manufacturing Company takes publication orders between 8:30 a.m. and 7:00 p.m. eastern standard time (EST). The phone number is (800) 879-2755. The fax number is (800) 284-4721.

You can also order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

A form for comments appears at the back of this publication. If the form has been removed, address your comments to:

IBM Corporation  
Department TC3  
PO BOX 60000  
CARY NC 27511-8519  
USA

You can fax comments to (919) 469-7718.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 1993, 1994. All rights reserved.**

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>Notices</b> . . . . .	ix
Trademarks . . . . .	ix
<b>About This Book</b> . . . . .	xi
Who Should Read This Book . . . . .	xi
What You Should Know . . . . .	xi
Organization of This Book . . . . .	xii
Highlighting Style . . . . .	xii
Related Publications . . . . .	xii
<b>Chapter 1. Network Overview</b> . . . . .	1
System Configuration . . . . .	1
User Interfaces . . . . .	2
Roles in CMVC . . . . .	2

---

## Part 1. Installing Your CMVC Products . . . . . 5

<b>Chapter 2. Hardware and Software Requirements</b> . . . . .	7
Requirements for the CMVC Servers . . . . .	7
CMVC/6000 . . . . .	8
CMVC for Sun Systems . . . . .	9
CMVC for HP Systems . . . . .	10
CMVC for Solaris Systems . . . . .	11
<b>Chapter 3. Installing the CMVC Servers</b> . . . . .	13
CMVC Packaging . . . . .	13
CMVC/6000 Packaging . . . . .	14
Prerequisite Tasks for Installing the CMVC Server Code . . . . .	14
CMVC for Sun Systems . . . . .	14
Installing the CMVC Server Code from Tape . . . . .	15
CMVC/6000 . . . . .	15
CMVC for Sun, Solaris, and HP Systems . . . . .	16
Installing the CMVC Server Code from CD-ROM . . . . .	16
CMVC/6000 . . . . .	17
CMVC for Sun Systems . . . . .	17
CMVC for Solaris Systems . . . . .	18
CMVC for HP Systems . . . . .	18
<b>Chapter 4. The NetLS Software</b> . . . . .	19
Installing the NetLS Software on the Server . . . . .	19
Understanding How CMVC Implements NetLS Licensing . . . . .	20

---

## Part 2. Configuring Your CMVC Environment . . . . . 21

<b>Chapter 5. Configuring Your Version Control System</b> . . . . .	23
Overview . . . . .	23
Your Version Control Environment . . . . .	23
File Versioning . . . . .	23
Binary Files . . . . .	24

Version Labels . . . . .	24
PVCS and SCCS Commands . . . . .	24
Customizing Your Environment to Use the PVCS Version Manager . . . . .	24
Installing PVCS Version Manager . . . . .	24
Resetting the PVCS Version Manager Executables . . . . .	24
Configuring the PVCS Version Manager Executables . . . . .	25
Using vconfig to Configure the PVCS Executables . . . . .	25
Registering Users to the PVCS License Administration Database . . . . .	25
Optional Customization of Your Environment . . . . .	27
PVCS Configuration Parameters . . . . .	27
Creating a Local Configuration File . . . . .	28
Modifying Your Local Configuration Files . . . . .	29
<b>Chapter 6. Creating a CMVC Family . . . . .</b>	<b>31</b>
Preconfiguration Planning . . . . .	31
Prerequisite Tasks or Conditions . . . . .	32
Configuring the CMVC Server . . . . .	32
Configuration as Root . . . . .	33
Configuration as the Familyname User . . . . .	34
Mandatory Environment Variables in Your Login Profile . . . . .	35
Optional Environment Variables in Your Login Profile . . . . .	39
Using the mkfamily Command . . . . .	41
Using the mkdb Command . . . . .	43
Using the rmfamily Command . . . . .	45
Using the rmdb Command . . . . .	46
<b>Chapter 7. Configurable Fields . . . . .</b>	<b>47</b>
Overview . . . . .	47
Properties of Configurable Fields . . . . .	47
Using Configurable Fields . . . . .	48
Default Configurable Fields Shipped by IBM . . . . .	48
Feature Table . . . . .	49
Defect Table . . . . .	50
Creating and Modifying Configurable Fields . . . . .	51
Running the chfield Program with the -source Option . . . . .	52
Running the chfield Program with the -object Flag Only . . . . .	53
Updating Reports . . . . .	55
<b>Chapter 8. Configuring Components and Releases . . . . .</b>	<b>59</b>
Planning Your Component Structure . . . . .	59
Planning Your Release Groupings . . . . .	59
Planning Your Processes . . . . .	60
Configuring Component Processes . . . . .	60
Configuring Release Processes . . . . .	61
Configuring Processes . . . . .	63
The cfgcomproc.ld and cfgrelproc.ld Files . . . . .	63
Conditions Applying to Configurable Processes . . . . .	65
Migration from CMVC Version 1 . . . . .	66
<b>Chapter 9. Configuring Authority Groups . . . . .</b>	<b>67</b>
Controlling Access Authority . . . . .	67
Base Authority . . . . .	67
Implicit Authority . . . . .	67
Explicit Authority . . . . .	67

Restricted Authority . . . . .	67
The CMVC Superuser Privilege . . . . .	68
Grouping Actions into Authority Groups . . . . .	68
Configuring CMVC Actions into Authority Groups . . . . .	69
Editing the authority.ld File . . . . .	69
Using the chauth Script to Reload the Authority Table . . . . .	70
<b>Chapter 10. Configuring Interest Groups . . . . .</b>	<b>73</b>
Controlling Notification of Actions . . . . .	73
Automatic Notification . . . . .	73
Explicit Notification . . . . .	73
Subscribers . . . . .	73
Grouping Actions into Interest Groups . . . . .	74
Configuring Interest Groups . . . . .	74
Editing the interest.ld File . . . . .	75
Using the chintr Script to Reload the Interest Table . . . . .	75
<b>Chapter 11. Modifying Your Configuration Table . . . . .</b>	<b>77</b>
The Configuration Database Table . . . . .	77
Configuration Types . . . . .	78
Modifying the Config Table . . . . .	78
Editing the config.ld File . . . . .	83
Using the chcfg Script to Reload the Config Table . . . . .	84
<b>Chapter 12. Providing User Exits . . . . .</b>	<b>87</b>
Configuring User Exits . . . . .	87
Editing the userExits File . . . . .	87
Writing User Exit Programs . . . . .	88

---

**Part 3. Working with CMVC . . . . . 93**

<b>Chapter 13. CMVC Server Daemons . . . . .</b>	<b>95</b>
Starting the CMVC Server cmvcd Daemons . . . . .	95
Prerequisite Tasks . . . . .	95
Starting cmvcd . . . . .	95
Starting cmvcd in Maintenance Mode . . . . .	96
Starting the CMVC Server notifyd Daemon . . . . .	97
Starting the CMVC Server Daemons on System Reboot . . . . .	97
Stopping the CMVC Server Daemons . . . . .	98
Stopping the CMVC Server Daemons on System Shutdown . . . . .	98
Recycling the CMVC Server Daemons . . . . .	99
<b>Chapter 14. Ongoing Maintenance . . . . .</b>	<b>101</b>
Audit Log . . . . .	101
Mail . . . . .	101
Mail Addressing . . . . .	102
Addressing Using a Central Database of Names and Addresses . . . . .	102
Addressing Using Domain Name Addressing . . . . .	102
Mail Queue—Processing Interval . . . . .	102
Restored Mail . . . . .	102
Aging Defects and Features . . . . .	103
The Age Shell Script . . . . .	103
The resetAge Program . . . . .	104

Monitoring the Performance of Your CMVC Server	104
Tuning Your CMVC Server	105
Monitoring the CMVC Server Daemons	105
Monitoring the Activity of the CMVC Server	108
Determining Which Users Issue Time Consuming Reports	108
Monitoring the Number of Requests Serviced	109
Monitoring Server Daemon Problems	109
Cleaning Up Shared Memory	109
Version Control Path Finder Tool	110
Managing Level Maps	111
Level Map File	112
Maintaining the Maps Directory	112
Errors	112
<b>Chapter 15. The CMVC Audit Log</b>	<b>115</b>
Managing the CMVC Audit Log	115
Prerequisite Tasks or Conditions	115
Cleaning Up the Log File	115
Format of the CMVC Audit Log	116
<b>Chapter 16. Bringing SCCS Files under CMVC Control</b>	<b>121</b>
Options for Bringing SCCS Files into CMVC	121
The SCCS File Migration Function	121
The SCCS File Import Function	122
Deciding to Migrate or Import SCCS Files	122
Preliminary Requirements and Planning	123
Prerequisites	123
Comparing SCCS Files and CMVC Files	123
Planning Your Component and Release Structures	123
Migration and Import Requirements	125
The SCCS File Migration Function	125
Stage 1: Running the Filemap Shell Script	126
Stage 2: Running the Filemigrate Shell Script	130
Stage 3: Running the Commands in the file.migrate File	131
Post-Migration Activities	132
The SCCS File Import Function	133
Stage 1: Running the Filemap Shell Script	133
Stage 2: Running the Fileimport Shell Script	133
Stage 3: Running the Commands in the file.import File	134
Post-Migration Activities	135
<b>Chapter 17. Backup and Recovery</b>	<b>137</b>
Backing Up the CMVC Server	137
Backing Up CMVC Family Database Tables, Views, and Indexes	138
Recovering CMVC	140
<b>Chapter 18. Archiving and Restoring</b>	<b>141</b>
Archiving and Restoring CMVC Data	141
Archive Functions	143
Restore Functions	143
Archive and Restore Preparation	143
Level Archive Prerequisites	144
Release Archive Prerequisites	145
Restore Prerequisites	146



Archive and Restore Limitations . . . . .	146
Archive and Restore Procedures . . . . .	147
Using the cmvcarchive Program . . . . .	147
Using the cmvcrestore Program . . . . .	149
Resetting Operating System Semaphores . . . . .	150
<b>Appendix A. Error Messages and Recovery . . . . .</b>	<b>151</b>
<b>Appendix B. Migrating to CMVC Version 2.3 . . . . .</b>	<b>179</b>
Databases Supported by Versions of CMVC . . . . .	179
Migration Utilities . . . . .	180
Using the dbConvert.v1r1m1 Utility . . . . .	181
Using the dbConvert.v2r1 Utility . . . . .	181
Pre-Migration Tasks . . . . .	181
Migration Tasks . . . . .	184
<b>Appendix C. Converting Existing CMVC Server from ORACLE6 to ORACLE7 . . . . .</b>	<b>195</b>
Prerequisites . . . . .	195
Steps for Migration . . . . .	195
Exporting from ORACLE6 . . . . .	195
Importing to ORACLE7 . . . . .	195
Run the Database Conversion Routine . . . . .	197
<b>Appendix D. Migrating to CMVC Server/6000 V2.3.0 for DB2/6000 . . . . .</b>	<b>199</b>
Prerequisites . . . . .	199
Steps for Migration . . . . .	199
Steps Required to Perform the Database Migration . . . . .	200
Estimating the additional disk space . . . . .	200
Installing the DB2/6000 database . . . . .	203
Stopping the cmvcd and notifyd daemons . . . . .	203
Backing up the database . . . . .	203
Installing the CMVC Server/6000 V2.3.0 for DB2/6000 . . . . .	203
Running the conversion . . . . .	203
Update Releases Table . . . . .	206
Starting the cmvcd and notifyd daemons . . . . .	206
Verifying the data of your family . . . . .	206
Reclaiming disk space . . . . .	207
<b>Appendix E. Authority Groups Worksheet . . . . .</b>	<b>209</b>
<b>Appendix F. Interest Groups Worksheet . . . . .</b>	<b>215</b>
<b>Appendix G. Configurable Processes Worksheets . . . . .</b>	<b>219</b>
<b>Appendix H. User Exit Parameters . . . . .</b>	<b>221</b>
Parameters Passed to User Exit Programs . . . . .	221
User Exit Parameter Definitions . . . . .	235
<b>Glossary . . . . .</b>	<b>241</b>
<b>Index . . . . .</b>	<b>247</b>



---

## Notices

**Warning:** You must not issue operating system or version control system commands against the files in the version control directories. Any tampering with these files can result in discrepancies between the information stored in the relational database on the CMVC server and the information in the version control directories.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY, USA 10594.

IBM may change this publication, the product described herein, or both. These changes will be incorporated in new editions of the publication.

---

## Trademarks

The following terms, denoted by an asterisk (\*) on their first occurrence in this publication, are trademarks of the IBM Corporation in the United States or other countries:

AIX	AIX/6000	Common User Access
CUA	DATABASE 2	DB2
DB2/6000	IBM	IBMLink
Operating System/2	OS/2	PROFS
PowerServer	RISC System/6000	

The following terms, denoted by a double asterisk (\*\*) on their first occurrence in this publication, are trademarks of other companies:

HP	Hewlett-Packard Company
HP-UX	Hewlett-Packard Company
INFORMIX	Informix Software, Inc.
NCS	Apollo Computer, Inc.
NetLS	Apollo Computer, Inc.
Network File System	Sun Microsystems Inc.
Network License System	Apollo Computer, Inc.
NFS	Sun Microsystems Inc.
ORACLE	Oracle Corporation
ORACLE7	Oracle Corporation
OSF/Motif	Open Software Foundation, Inc.
PVCS Version Manager	INTERSOLV, Inc.

SoftBench	Hewlett-Packard Company
Solaris	Sun Microsystems, Inc.
SPARCserver	Sun Microsystems, Incorporated
SPARCstation	Sun Microsystems, Incorporated
SQL*Loader	Oracle Corporation
SQL*Plus	Oracle Corporation
Sun	Sun Microsystems Inc.
SunOS	Sun Microsystems Inc.
SYBASE	Sybase, Inc.
SYBASE Open Client DB-Library/C	Sybase, Inc.
SYBASE SQL Server	Sybase, Inc.
UNIX	UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.
Windows	Microsoft Corporation

---

## About This Book

This book is part of the library that supports the IBM\* Configuration Management Version Control (CMVC) licensed programs for CMVC *clients* and *servers*.

It describes how to install, configure, and maintain the CMVC server. For information about installing, configuring, and maintaining the CMVC clients, read the following books

- *IBM CMVC UNIX Client Installation and Configuration*
- *IBM CMVC Client/2 Getting Started*.
- *IBM CMVC Getting Started Using the DOS/Windows\*\* Client*.

---

## Who Should Read This Book

This book describes all administrative tasks required to customize, maintain, and support the CMVC server within your development environment. The administrative tasks should be the responsibility of one or two persons who are familiar with *database* concepts and system administration. Users doing any of the following tasks should read this book:

- Hardware configuration planning
- Installation of the relational database
- Installation of the CMVC server
- Preconfiguration planning and configuration of a CMVC *family*
- Maintenance and backup activities.

---

## What You Should Know

The administrator must also know how to use CMVC. See the books *IBM CMVC User's Guide* and the *IBM CMVC Commands Reference* for a detailed description of how to use CMVC. Before continuing with this book, read *IBM CMVC Concepts* for a description of the concepts and processes involved in using CMVC.

You should be familiar with system administration procedures for your operating system. You also must be familiar with one of the following relational database management systems:

- IBM DATABASE 2\* AIX/6000\* (DB2/6000\*)
- ORACLE\*\* Relational Database Management System
- INFORMIX\*\*-Online relational database
- SYBASE SQL Server\*\* relational database.

Review the documentation for the relational database at your site before starting installation.

---

## Organization of This Book

This book is designed to lead the administrator through each of the steps that are required to install, configure, and maintain a CMVC server.

The information in this book is divided into the following parts:

- Part 1, “Installing Your CMVC Products,” outlines your CMVC server hardware and software requirements. It also describes how to install the CMVC server and the *Network License System\*\* (NetLS\*\*)* software.
- Part 2, “Configuring Your CMVC Environment,” describes the post-installation planning needed for each CMVC family. It also shows how to configure your version control environment and your CMVC server, and shows how to establish *authority* groups, interest groups, and the configuration defaults for your family.
- Part 3, “Working with CMVC,” outlines the activities that you can perform after your initial setup. These activities include starting your CMVC server, migrating *files* from SCCS, managing the audit log, using the mail facility, monitoring *daemon* activity, aging *defects*, backing up and recovering data, and archiving and restoring *releases* and *levels*.

---

## Highlighting Style

The following highlighting style is used in this book:

- |               |  |
|---------------|--|
| <b>Bold</b>   | Commands, flags, files, directories, field names, and other items predefined by CMVC are in bold. Valid abbreviations for commands also are in bold.   |
| <i>Italic</i> | Arguments or options whose names or values must be supplied by you are in italics. Italics are also for emphasis, for the first occurrence in text of items that are in the glossary, and for the titles of manuals. |
| Monotype      | Examples of specific data values, examples of displayed text, messages, menu items that you select to initiate an action, or information that you should type are in monotype.                                       |

---

## Related Publications

The following books contain additional information about CMVC and are shipped with the server.

- *IBM CMVC Concepts*, SC09-1633, provides the basis for your understanding of CMVC. It describes in detail the concepts and processes involved in using CMVC.
- *IBM CMVC User's Guide*, SC09-1634, describes all CMVC *actions* as implemented in the *graphical user interface (GUI)* on the AIX, Sun-OS, Solaris, and HP-UX platforms.
- *IBM CMVC User's Reference*, SC09-1597, contains the reference lists, tables, and *state* diagrams for CMVC. It also describes how the message-integrated CMVC uses the *Broadcast Message Server (BMS)* to fully integrate with other integrated development environment tools.

- *IBM CMVC Commands Reference*, SC09-1635, describes all CMVC commands, their syntax and use, as implemented in the command-line interface.
- *IBM CMVC UNIX Client Installation and Configuration*, SC09-1596, contains detailed information needed to install and configure the CMVC clients.
- *NetLS Quick Start Guide*, SC09-1661, provides the information needed to set up the Network License System (NetLS) software to work with CMVC.
- *Managing Software Products with the Network License System*, SC09-1660, provides the information needed to manage the use of the NetLS software with CMVC.
- *Managing NCS\*\* Software*, SC09-1834, provides additional information for managing the NetLS software.

*IBM CMVC Client/2 Getting Started*, SC09-1599, is shipped with the OS/2 workstation client for CMVC and can be ordered separately. It contains detailed information about installing and configuring the OS/2 workstation client for CMVC.

*IBM CMVC Getting Started Using the DOS/Windows Client*, SC09-3000, is shipped with the DOS/Windows workstation client for CMVC and can be ordered separately. It contains detailed information about installing and configuring the DOS/Windows workstation client for CMVC.

When necessary, refer to your operating system or relational database documentation while installing or using CMVC.





---

## Chapter 1. Network Overview

This chapter introduces IBM Configuration Management Version Control (CMVC) and briefly describes the system configuration, the user interfaces, and the three main *user* roles within a CMVC environment.

---

### System Configuration

CMVC consists of the following products:

- CMVC/6000
- CMVC for HP\*\* systems
- CMVC for Sun\*\* systems
- CMVC for Solaris\*\* systems

Each product comprises a server and clients. Designed for use in a networked environment, each of the CMVC products is based on a client-server model, as shown in Figure 1.

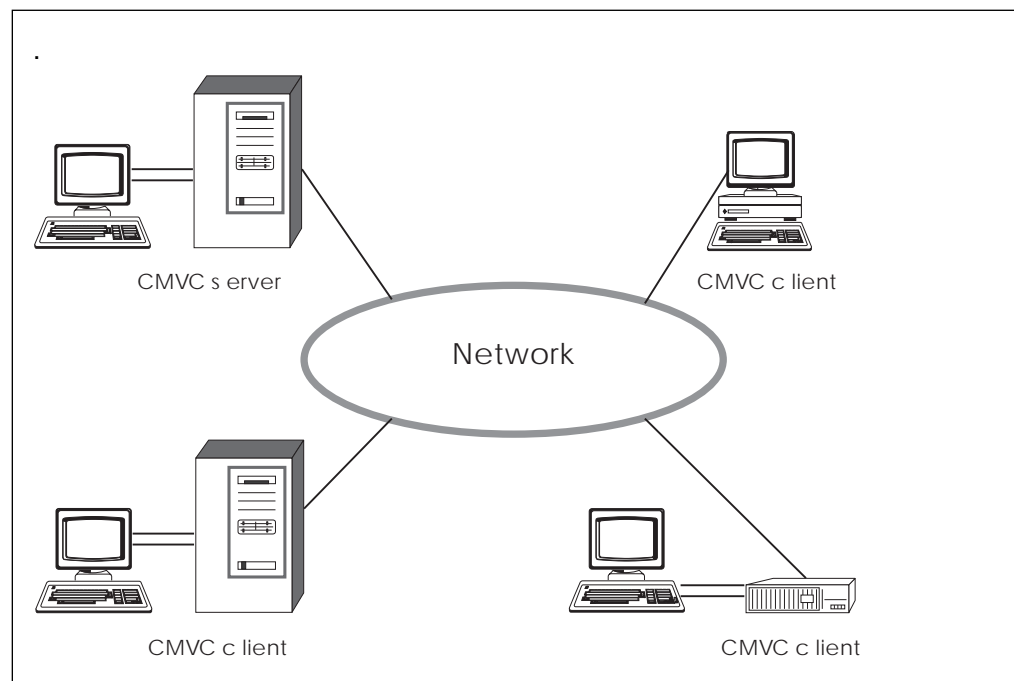


Figure 1. Example of a Client-Server Network of CMVC

A CMVC server is a workstation that runs the CMVC server software that controls all data within the CMVC environment. Files are stored in a file system in combination with a version control system on the CMVC server. All other development data is stored in a relational database on the CMVC server. The CMVC server handles this database as a local database. A CMVC client is a workstation that runs the CMVC client software to access the information and files stored on the CMVC server. With this client-server architecture, users can access project files and data without needing to know where the networked resources physically reside.

## User Interfaces

CMVC supports two graphical user interfaces (GUIs) based on the OSF/Motif\*\* window manager, one GUI for the OS/2 and DOS/Windows clients that is based on IBM's Common User Access\* (CUA\*) 1991 guidelines, and a command-line interface.

The two OSF/Motif GUIs are:

- A non-message-integrated GUI for operation in your development environment.
- A message-integrated GUI for operation with the following environments:
  - The IBM AIX\* Software Development Environment (SDE) WorkBench/6000 product
  - The HP SoftBench\*\* and HP SoftBench for Sun and Solaris products

For more information about the OSF/Motif GUIs, see the book *CMVC User's Guide*. For more information about the OS/2 GUI, see the book *IBM CMVC Client/2 Getting Started*. For more information about the DOS/Windows GUI, see the book *Getting Started Using the DOS/Windows Client*.

- A command-line interface is provided for use from an operating system shell. For more information about this interface, see the book *IBM CMVC Commands Reference*.

**Note:** When using the DOS/Windows client, you cannot issue CMVC commands from a DOS prompt. However, you can issue CMVC commands from within the GUI.

## Roles in CMVC

The roles of users within the CMVC environment can be divided into three main categories:

- *System administrator*
- *Family administrator*
- *End user.*

For the purposes of all documentation within the CMVC library, the roles are defined as follows.

### System Administrator

The system administrator is responsible for:

- Installing, maintaining, and backing up the CMVC server
- Installing, maintaining, and backing up the relational database used by CMVC
- Planning, maintaining, and configuring all client and server hardware.

The system administrator has root access to the CMVC server and database administration access to the relational database.

### Family Administrator

The family administrator is responsible for:

- Planning and configuring one or more CMVC families
- Managing user access to one or more CMVC families
- Maintaining one or more CMVC families.

The family administrator has root access to the CMVC server and database administration access to the relational database.

**End User**

The end user uses CMVC within one or more families. This user has access to one or more CMVC clients.



---

## Part 1. Installing Your CMVC Products

This part of the book outlines the CMVC server hardware and software requirements. It also describes how to install the CMVC server and the NetLS software.



---

## Chapter 2. Hardware and Software Requirements

CMVC consists of the following products:

- CMVC/6000
- CMVC for HP systems
- CMVC for Sun systems
- CMVC for Solaris systems

Each product comprises a server and clients. This chapter lists the hardware and software requirements for the CMVC servers. Refer to the following books for information on the hardware and software requirements for the CMVC clients:

- *IBM CMVC Client Installation and Configuration*
- *IBM CMVC Client/2 Getting Started*
- *IBM CMVC Getting Started Using the DOS/Windows Client.*

---

### Requirements for the CMVC Servers

CMVC supports these servers: the CMVC/6000 server, the CMVC for Solaris systems server, the CMVC for Sun systems server, and the CMVC for HP systems server.

The minimum amount of internal memory required to run a CMVC server is 16 megabytes (MB), not including the memory required to run the database products and the operating system. However, performance may be affected by total system memory and the amount of fixed-disk storage available. See "Preconfiguration Planning" on page 31 for recommended disk space requirements relative to the size of your CMVC family.

The following are the minimum hard disk storage required to install the CMVC server for the various platforms:

<b>Server Platform</b>	<b>Minimum Hard Disk Requirements</b>
RISC System/6000	12 MB
Sun systems	18 MB
HP systems	16 MB
Solaris systems	30 MB

After you install the CMVC server code, you must register and obtain the NetLS password before you can install the NetLS server and use CMVC. The NetLS software is an integral part of CMVC and is for use only with CMVC. The NetLS password is unique and is based on the machine ID where the NetLS server or servers are installed, the product and version number, and the number of concurrent user licenses purchased.

You can obtain the NetLS passwords by contacting the IBM Registration Center. Before contacting the IBM Registration Center, read the document *Read This First* that comes with your copy of the CMVC product. Then complete the relevant sections of the Product Registration Information Form that is included in the document.

For more information about the NetLS product, see the book *Managing Software Programs with the Network Licensing Software*, SC09-1660.

## CMVC/6000

The CMVC/6000 server code requires the following hardware:

- Any IBM RISC System/6000\* PowerServer\* workstation Model 320 or higher
- Any terminal supported by the RISC System/6000 workstation
- Any token-ring or Ethernet Local Area Network (LAN) adapter card supported by the RISC System/6000 workstation that supports *Transmission Control Protocol/Internet Protocol (TCP/IP)*
- Keyboard
- Printer (optional).

The CMVC/6000 server code requires the following software:

- Any supported CMVC Version 2.3 client (corequisite product)
- AIX Version 3.2 for RISC System/6000 operating system which includes TCP/IP and *Network File System\*\* (NFS\*\*)* software or a later release of Version 3
- One of the following version control systems:
  - Source Code Control System (SCCS), which comes with the AIX Base Application Development Toolkit
  - INTERSOLV PVCS Version Manager\*\* Version 5.1 (only required if using the PVCS Version Manager software instead of SCCS).
- One of the following relational databases operating on AIX Version 3.2 or a later release of Version 3:
  - IBM Database 2 AIX/6000 (DB2/6000) (Program No. 5765-172)  
**Note:** The DB2/6000 database requires AIX version 3.2.4 or 3.2.3 with a program temporary fix (PTF).
  - INFORMIX relational database
    - INFORMIX-OnLine (Runtime) Version 5.00 or a later release of Version 5 and the corresponding release of
    - INFORMIX-SQL (Development) software.
  - ORACLE relational database
    - Oracle RDBMS (runtime) Version 7 and the corresponding release of
    - ORACLE (Transaction Processing Options) TPO (Runtime)
    - SQL\*Plus\*\* (Runtime).
  - SYBASE\*\* relational database
    - SYBASE SQL Server V4.9.2, or a later release of Version 4, or SYBASE SQL Server System 10 and the corresponding release of
    - SYBASE Open Client DB-Library/C\*\*.

**Note:** If you are using a DB2/6000 database, the recommended requirement is to get a license for the database server with at least 4 connections. One of the connections is for the `notifyd` daemon while at least one



token is required to run the cmvcd daemon. You can use the other two connections to run various CMVC server tools including: age, chfield, cmvcarchive, cmvcrestore, mkdb, chintr, chcfg, chcomproc, chrelproc, vcPath, and resetAge. The connections are needed for the duration of the running of each tool. The actual number of connections required is dependent on the number of cmvcd daemons you intend to run.

The minimum user license required by the database vendors is a 2-8 user license for the ORACLE and SYBASE databases, and a 2-16 user license for the INFORMIX database. However, the number of user licenses required depends on the number of CMVC/6000 server daemons started by the CMVC administrator. This number varies depending on the number of CMVC clients simultaneously requesting service from the CMVC/6000 server.

## CMVC for Sun Systems

The CMVC for Sun systems server code requires the following hardware:

- A SPARCserver\*\* 10 workstation or any binary compatible SPARCserver workstation
- Any display supported by the SPARCserver workstation
- Any token-ring or Ethernet LAN adapter card supported by the SPARCserver workstation that supports TCP/IP
- Keyboard
- Printer (optional).

The CMVC for Sun systems server code requires the following software:

- Any supported CMVC Version 2.3 client (corequisite product)
- SunOS\*\* Version 4.1.3
- A Korn shell, required to run the SCCS Migration and Import tools, and the sample scripts
- One of the following relational databases operating on the SunOS operating system:
  - ORACLE relational database
    - ORACLE RDMBS (Runtime) Version 7 and the corresponding release of
    - ORACLE (Transaction Processing Options) TPO (Runtime)
    - SQL\*Plus (Runtime).
  - INFORMIX relational database
    - INFORMIX-OnLine (Runtime) Version 5.00 or a later release of Version 5 and the corresponding release of
    - INFORMIX-SQL (Development) software.
  - SYBASE relational database
    - SYBASE SQL Server V4.9.2, or a later release of Version 4, or SYBASE SQL Server System 10 and the corresponding release of
    - SYBASE Open Client DB-Library/C.

**Note:** The minimum user license required by the database vendors is a 2-8 user license for the ORACLE database and a 2-16 user license for the INFORMIX database. However, the number of user licenses required depends on the number of CMVC server daemons started by the CMVC administrator. This number varies depending on the number of CMVC clients simultaneously requesting service from the CMVC server.

For users of the CMVC for Sun systems server code in conjunction with SCCS, SCCS comes with the SunOS operating system.

**Note:** CMVC for Sun systems does not support INTERSOLV's PVCS Version Manager software.

## CMVC for HP Systems

The CMVC for HP systems server code requires the following hardware:

- Any HP 9000 Series 700 or 800 workstation
- Any display supported by the Series 700 or 800 workstation
- Any token-ring or Ethernet LAN adapter card supported by the Series 700 or 800 workstation that supports TCP/IP
- Keyboard
- Printer (optional).

The CMVC for HP systems server code requires the following software:

- Any supported CMVC Version 2.3 client (corequisite product)
- HP-UX\*\* Version 9.0, or later, which includes TCP/IP and NFS software
- One of the following relational databases operating on the HP-UX operating system:
  - ORACLE relational database
    - ORACLE RDBMS (Runtime) Version 7 and the corresponding release of
    - ORACLE (Transaction Processing Options) TPO (Runtime)
    - SQL\*Plus (Runtime) software.
  - INFORMIX relational database
    - INFORMIX-OnLine (Runtime) Version 5.00 or a later release of Version 5 and the corresponding release of
    - INFORMIX-SQL (Development) software.

**Note:** The minimum user license required by the database vendors is a 2-8 user license for the ORACLE database and a 2-16 user license for the INFORMIX database. However, the number of user licenses required depends on the number of CMVC server daemons started by the CMVC administrator. This number varies depending on the number of CMVC clients simultaneously requesting service from the CMVC server.

For users of the CMVC for HP systems server code with SCCS, SCCS comes with the HP-UX operating system.

**Note:** CMVC for HP systems does not support INTERSOLV's PVCS Version Manager software.

## CMVC for Solaris Systems

The CMVC for Solaris systems server code requires the following hardware:

- A SPARCserver 10 workstation or any binary compatible SPARCserver workstation
- Any display supported by the above SPARCserver workstation
- Any Token Ring or Ethernet LAN adapter card supported by the above SPARCserver workstation that supports TCP/IP
- Keyboard
- Printer (optional).

The CMVC for Solaris systems server code requires the following software:

- Any supported CMVC Version 2.3 client (corequisite product)
- Solaris Version 2.3 operating system
- One of the following relational databases operating on the Solaris operating system:
  - ORACLE relational database
    - ORACLE RDMBS (Runtime) Version 7 and the corresponding release of
    - ORACLE (Transaction Processing Options) TPO (Runtime)
    - SQL\*Plus (Runtime).
  - SYBASE relational database
    - SYBASE SQL Server V4.9.2, or a later release of Version 4, or SYBASE SQL Server System 10 and the corresponding release of
    - SYBASE Open Client DB-Library/C.

**Note:** The minimum user license required by the database vendor is a 2-8 user license for the ORACLE and SYBASE databases. However, the number of user licenses required depends on the number of CMVC server daemons started by the CMVC administrator. This number varies depending on the number of CMVC clients simultaneously requesting service from the CMVC server.

For users of the CMVC for Solaris systems code in conjunction with SCCS, SCCS is available with the Solaris operating system.

**Note:** CMVC for Solaris systems does not support INTERSOLV's PVCS Version Manager software.



---

## Chapter 3. Installing the CMVC Servers

You can install the CMVC server code from tape or from CD-ROM. This chapter describes the packaging for the CMVC server code, the system requirements for its installation, and the procedures for installing it.

**Note:** If you are migrating from CMVC Version 1.1, the installation process overwrites your existing CMVC server code. Be prepared to restore the original Version 1.1 server code, either from a *backup copy* of your server or from the original product tape.

After installing the Version 2, Release 3 server code, refer to Appendix B, "Migrating to CMVC Version 2.3" on page 179 for instructions on converting the relational database to the new format.

Before installing the CMVC servers, read the book *IBM CMVC Concepts* for a thorough understanding of components, releases, and files in the CMVC environment.

---

### CMVC Packaging

The IBM CMVC servers are packaged on tapes as well as a CD-ROM.

Each tape includes all the files for one server and one client:

- IBM CMVC/6000 server code, IBM CMVC/6000 client code, and NetLS Runtime Kit for RISC System/6000 workstations
- IBM CMVC for Sun systems server code, IBM CMVC for Sun systems client code, and NetLS Runtime Kit for Sun SPARCstation\*\* workstations
- IBM CMVC for HP systems server code, IBM CMVC for HP systems client code, and NetLS Runtime Kit for HP 9000 Series 700 workstations.
- IBM CMVC for Solaris systems server code, IBM CMVC for Solaris systems client code, and NetLS Runtime Kit for Sun SPARCstation workstations

The CD-ROM includes all the files for:

- IBM CMVC/6000 server code
- IBM CMVC for Sun systems server code
- IBM CMVC for HP systems server code
- IBM CMVC for Solaris systems server code
- IBM CMVC/6000 client code
- IBM CMVC for Sun systems client code
- IBM CMVC for HP systems client code
- IBM CMVC for Solaris systems client code
- NetLS Runtime Kits for these platforms.

You must know the *installp* image names in the CMVC server package only if you are installing the CMVC/6000 server code. The *installp* image names for the CMVC/6000 server code are listed in the following section.

## CMVC/6000 Packaging

The CMVC/6000 server consists of the following installp images:

Installp Image	Description of Contents
<b>cmvcsrvmlanguage.msg</b>	The CMVC/6000 server message catalog, where <i>language</i> , for example, is equal to En_US for US English.
<b>cmvcsrvdb2.obj</b>	The CMVC/6000 server software with support for an DB2/6000 relational database.
<b>cmvcsrvora7.obj</b>	The CMVC/6000 server software with support for an ORACLE7** relational database.
<b>cmvcsrvinfx.obj</b>	The CMVC/6000 server software with support for an INFORMIX relational database.
<b>cmvcsrvsyb.obj</b>	The CMVC/6000 server software with support for a SYBASE relational database.
<b>NetLS.Ark</b>	The NetLS Runtime Kit for RISC System/6000 workstations.

The package may also contain update files.

---

## Prerequisite Tasks for Installing the CMVC Server Code

For all CMVC servers, you must do the following prerequisite tasks:

1. Log in as root.
2. Install the appropriate operating system for your server code:

Server Code	What to Install
CMVC/6000 server	AIX Base Operating System (BOS) Runtime. (It is part of the IBM AIX Version 3 Release 2 for RISC System/6000 licensed program.)
CMVC for Sun systems server	SunOS Version 4.1.3
CMVC for HP systems server	HP-UX Version 9.0
CMVC for Solaris systems server	Solaris 2.3

3. Install TCP/IP.
4. Install the NFS software.

**Note:** If you are installing the CMVC/6000 server code for use with a DB2/6000 relational database, you need to run the database's **db2ln** command as well. Refer to the installation guide of the DB2/6000 database for more information.

## CMVC for Sun Systems

For the CMVC for Sun systems server code, install a Korn shell if you plan to use the SCCS migration or import scripts, or any of the sample scripts shipped with the CMVC server code. The CMVC for Sun systems server code does not require a Korn shell. The SCCS migration and import scripts and all sample scripts shipped with the CMVC server code, however, must be run under a Korn shell. If you do not have a Korn shell, you cannot use these scripts.

These scripts were tested with the Unix System Laboratories Inc. Korn Shell Version 88 Release f. No testing has been done with the Korn shells supplied by other software vendors.

---

## Installing the CMVC Server Code from Tape

This section describes how to install the CMVC server code from tape. To install your CMVC server code from CD-ROM, see “Installing the CMVC Server Code from CD-ROM” on page 16.

Each tape contains the code for a server, a client, and a NetLS Runtime Kit. For instructions on installing the CMVC clients, refer to the book *IBM CMVC UNIX Client Installation and Configuration*.

### CMVC/6000

To install the CMVC/6000 server code from tape:

1. Log in as root.
2. Insert the tape containing the CMVC/6000 server installp images into the tape drive.
3. Type the following on your command line:

```
smit startup
```

This command starts the System Management Interface Tool (SMIT), which presents a menu-driven environment for the installation process. For information on SMIT, refer to your AIX operating system documentation.

4. Follow the directions and answer the prompts in the SMIT installation menus to select the *one* installp image that is relevant for your database. Only select one image because the first selected image is overwritten by the next selected image.

Installp Image	Description of Contents
cmvcsrvdb2.obj	CMVC/6000 server for DB2/6000
cmvcsrvora7.obj	CMVC/6000 server for ORACLE7
cmvcsrvinfx.obj	CMVC/6000 server for INFORMIX
cmvcsrvsyb.obj	CMVC/6000 server for SYBASE

5. Follow the directions on the SMIT installation menus to install the **cmvcsrvmlanguage.msg** installp image, such as `cmvcsrvmlanguage_US.msg`.
6. Follow the directions on the SMIT installation menus to install the **NetLS.Ark** installp image, if you want to install the NetLS software on the CMVC server. For information on completing the NetLS software installation, refer to Chapter 4, “The NetLS Software.”

## CMVC for Sun, Solaris, and HP Systems

To install the CMVC for Sun systems server code, the Solaris server code, or the CMVC for HP systems server code from tape:

1. Log in as root.
2. Insert the tape containing the CMVC files into your tape drive.
3. Create a directory for the CMVC server code files, for example, `/usr/lpp/cmvc`.
4. Determine the device name for your tape drive.

Figure 2 shows common device names for the Sun, HP, and Solaris platforms. Verify the device name with your system administrator before proceeding.

Tape Mode	Sun Device Name	HP Device Name	Solaris Device Name
Rewind	<code>/dev/rst0</code>	<code>/dev/rmt0m</code>	<code>/dev/rmt0m</code>
Non-rewind	<code>/dev/nrst0</code>	<code>/dev/rmt0mn</code>	<code>/dev/rmt0mn</code>

Figure 2. Common Device Names for Tape Drives

5. Assuming that `TAPEDEV` is the name of the device used to access your tape drive in a nonrewinding mode, enter the following commands for HP-UX:

```
cd /usr/lpp/cmvc
mt -t TAPEDEV rewind
dd if=TAPEDEV skip=0 > cmvcinstall
chmod u+x cmvcinstall
./cmvcinstall
```

Enter the following commands for SunOS or Solaris:

```
cd /usr/lpp/cmvc
mt -f TAPEDEV rewind
dd if=TAPEDEV skip=0 > cmvcinstall
chmod u+x cmvcinstall
./cmvcinstall
```

The **cmvcinstall** script prompts you for the name of the device used to access your tape drive in nonrewind mode and in rewind mode. You must supply the correct values to run the installation script successfully.

6. To install either the CMVC server or the NetLS Runtime Kit, follow the instructions and answer the prompts presented by the **cmvcinstall** script.

**Note:** The NetLS software supplied with CMVC for HP systems can only be installed on HP Series 700 and Series 800 workstations. You must run the CMVC-supplied NetLS software instead of the HP-supplied NetLS software.

---

## Installing the CMVC Server Code from CD-ROM

This section describes how to install the CMVC server code from CD-ROM. To install your CMVC server code from tape, see “Installing the CMVC Server Code from Tape” on page 15.

The CD-ROM contains the software for the UNIX CMVC servers and CMVC clients. For instructions on installing the UNIX CMVC clients, refer to the book *IBM CMVC UNIX Client Installation and Configuration*.



The terms and conditions for using the CMVC servers are in the file **cpyright.inf** on the CD-ROM. You can display or print **cpyright.inf**, which is in ASCII format, from your workstation.

If you are installing the CMVC for HP systems server code, all file names are displayed in uppercase.

## CMVC/6000

To install the CMVC/6000 server code from CD-ROM:

1. Log in as root.
2. Insert the CD-ROM in the CD-ROM drive.
3. Type the following on the AIX command line:

```
smit
```

This command starts the System Management Interface Tool (SMIT) which presents a menu-driven environment for system administration tasks. For more information on SMIT, see your AIX operating system documentation.

4. Use SMIT to mount the CD-ROM as a file system. Mount the file system into a directory such as /cdrom.
5. Run the standard software installation procedure from SMIT. Specify the source directory that you created in step 4.
6. Follow the directions and answer the prompts in the SMIT installation menus to select the *one* installp image that is relevant for your database. Only select one image because the first selected image is overwritten by the next selected image.

Install Image	Description of Contents
cmvcsrvdb2.obj	CMVC/6000 server for DB2/6000
cmvcsrvora7.obj	CMVC/6000 server for ORACLE7
cmvcsrvinfx.obj	CMVC/6000 server for INFORMIX
cmvcsrvsyb.obj	CMVC/6000 server for SYBASE

7. Follow the directions on the SMIT installation menus to install the **cmvcsrvmlanguage.msg** installp image.
8. Follow the directions on the SMIT installation menus to install the **NetLS.Ark** installp image, if you want to install the NetLS software on the CMVC server.

**Note:** For information on completing the NetLS software installation, refer to Chapter 4, "The NetLS Software."

## CMVC for Sun Systems

To install the CMVC for Sun systems server code from CD-ROM:

1. Insert the CD-ROM into the CD-ROM drive.
2. Log in as root and type the following commands:

```
mkdir /cdrom
mount -r -t hsfs /dev/sr0 /cdrom
cd /cdrom
./cmvcinst
```

3. Follow the instructions in the installation script to install the appropriate CMVC server or the NetLS Runtime Kit, or both.
4. After the installation is completed, type the following commands:  

```
umount /cdrom  
eject /dev/sr0
```

## CMVC for Solaris Systems

To install the CMVC for Solaris systems server code from CD-ROM:

1. Insert the CD-ROM into the CD-ROM drive. The Solaris volume management automatically mounts the CD-ROM as `/cdrom/cmvc_230`.
2. Log in as root and type the following commands:  

```
cd /cdrom/cmvc_230  
./cmvcinst
```
3. Follow the instructions in the installation script to install the appropriate CMVC server or the NetLS Runtime Kit, or both.
4. After the installation is completed, type the following command:  

```
eject cdrom
```

**Note:** If volume management is not running, contact your system administrator for instructions on mounting the CD-ROM.

## CMVC for HP Systems

To install the CMVC for HP systems server code from CD-ROM:

1. Insert the CD-ROM into the CD-ROM drive.
2. Log in as root and use System Administration Manager (SAM) to mount the CD-ROM as a file system under a directory, such as `/cdrom`.
3. Type the following commands:  

```
cd /cdrom  
./CMVCINST
```
4. Follow the instructions in the installation script to install the appropriate CMVC server or the NetLS Runtime Kit, or both.  
  
**Note:** The NetLS software supplied with CMVC for HP systems can only be installed on HP Series 700 workstations. You must run the CMVC-supplied NetLS software instead of the HP-supplied NetLS software.
5. After the installation is completed, type the following command:  

```
umount /cdrom
```

---

## Chapter 4. The NetLS Software

This chapter gives a brief overview of the Network License System (NetLS) software and provides guidelines for installing and configuring it.

CMVC uses the Network License System (NetLS) software to license usage. The number of concurrent CMVC users that can access CMVC is determined by the number of concurrent licenses, or tokens, purchased with CMVC. The NetLS software enforces the usage of these tokens. This type of licensing has several benefits:

- The CMVC server does not need any license control. The number of CMVC server instances is controlled by the number of database licenses.
- The CMVC UNIX clients require a NetLS token to operate. The CMVC client software can be run from any machine on the network, provided that the client has access to a NetLS license server.

For more detailed information about the NetLS software, refer to the following NetLS documentation that is shipped with CMVC:

- *NetLS Quick Start Guide*
- *Managing Software Products with the Network License System*
- *Managing NCS Software.*

You can also refer to the **README.NetLS** file in the **usr/ipp/cmvc/install** directory for additional information about NetLS.

---

### Installing the NetLS Software on the Server

The NetLS Runtime Kit (ARK) contains the NetLS server daemon, **netlsd**, and administration software. The NetLS software must be installed on a network node before a user can issue any CMVC commands or start the CMVC GUI.

Because each CMVC command consults the NetLS license server, it is recommended that you have:

- More than one instance of the NetLS license server serving the entire network
- A separate NetLS license server for each distinct network.

Do the following to install the NetLS software for use with CMVC.

1. Determine the computers to use as NetLS license servers.
2. If you have not already installed the NetLS code from the distribution media, for example, CD-ROM or tape, refer to Chapter 3, "Installing the CMVC Servers" on page 13 for instructions.
3. Start the NetLS system by running the **/usr/lib/netls/conf/netls\_config** shell script or by following the manual instructions in the *NetLS Quick Start Guide*. For **netlsd** options, refer to *Managing Software Products with the Network License System*. Because CMVC uses the concurrent license scheme, refer to the concurrent scheme sections in the NetLS documentation.

4. Obtain the target ID of each computer that will act as a NetLS license server by running the **ls\_targetid** program. By default, the **ls\_targetid** program is in the **/usr/lib/netls/bin** directory after installation.
5. If you are running more than one NetLS license server, you must determine the number of tokens you want to have available on each license server.
6. Contact IBM to receive your NetLS passwords. A NetLS password is unique and is based on the planar ID of each computer that will act as the NetLS license server, the product and version number, and the number of concurrent licenses purchased.  
**Note:** You must provide IBM with the information requested on the IBM Product Registration Information Form, which is included in the CMVC package. Contact information for IBM is provided on the IBM Product Registration Information Form.
7. Run the license administration program, **ls\_admin**, to install the NetLS licenses. By default, the **ls\_admin** program is in the **/usr/lib/netls/bin** directory after installation.

**Warning:**

- To shut down a NetLS server, you must stop the following daemons in the order listed:
  - NetLS (netlsd)
  - Global Location Broker (glbd)
  - Local Location Broker (llbd)
- We recommend that you do **not** run the nrglbd daemon from NCS with CMVC.

---

## Understanding How CMVC Implements NetLS Licensing

To help you better utilize CMVC, the following lists some ways in which CMVC implements NetLS licensing:

- By default, the duration of a NetLS token is a minimum of 15 minutes. However, you can increase this time.
- Each time a CMVC request is issued by a unique userid/groupid/hostid combination, a new token is acquired.
- When a userid/groupid/hostid that already has a token issues a request, that same token is kept but its time-out period is reset to the specified time-out period.
- When a user exits the CMVC GUI or a line command completes, the token is not released until its time-out period expires.
- A token must be available to issue any CMVC command.
- Because CMVC GUI actions frequently initiate CMVC commands, the GUI generally requires a continuous token while it is being used.
- When CMVC tokens are not available, a request issued by a new userid/groupid/hostid is queued until one becomes available.
- By default, the CMVC client cannot acquire tokens outside its local network. If you want to use a NetLS license server that is outside the client's local network, refer to the *Managing NCS Software* book for more information.

---

## **Part 2. Configuring Your CMVC Environment**

This part of the book describes the post-installation planning needed for each CMVC family. It also shows how to configure your version control environment and your CMVC server and shows how to establish authority groups, interest groups, and the configuration defaults for your family.



---

## Chapter 5. Configuring Your Version Control System

CMVC supports two different version control systems: SCCS and PVCS Version Manager. Decide which system you intend to use, before you configure the CMVC server.

CMVC for Sun systems, CMVC for Solaris systems, and CMVC for HP systems support SCCS only.

---

### Overview

The chosen version control system must be installed and initialized before using CMVC. When this has been accomplished and once the CMVC server environment has been initialized with the appropriate environment variables, the version control system will become transparent to users of CMVC.

After the version control system is established and the CMVC environment is made available to users, the version control directories and files on the *familyname* account should not be altered in any way.

**Warning:** You must not issue operating system or version control system commands against the files in the version control directories. Any tampering with these files can result in discrepancies between the information stored in the relational database on the CMVC server and the information in the version control directories.

Refer to your PVCS documentation for additional information regarding PVCS Version Manager. CMVC controls the directories in which PVCS Version Manager files will be created, therefore there is no need to create special directories as suggested in the PVCS Version Manager documentation.

---

### Your Version Control Environment

The version control system is transparent to users of CMVC once the environment has been configured. However, the following areas are handled differently by SCCS and PVCS Version Manager.

### File Versioning

The PVCS Version Manager and SCCS use different version numbering methods. CMVC will work with either method.

Usually PVCS file revisions begin with 1.0, whereas SCCS file versions begin with 1.1. For consistency, CMVC will request PVCS file revisions to begin at 1.1. When project complexity increases and the file branching occurs, CMVC does not intervene with either PVCS Version Manager or SCCS; it allows the version control mechanisms to branch as designed. Therefore, if a branch is made between revision 1.1 and 1.2 of a file when using PVCS Version Manager, the resulting version number will be 1.1.1.0, whereas the same branch using SCCS would result in a version number of 1.1.1.1.

## Binary Files

When storing binary files within a CMVC environment that uses PVCS Version Manager as its underlying version control mechanism, CMVC handles the run-time modification of the configuration parameters that control keyword expansion and carriage-return-line-feed (cr/lf) translation. When storing binary files within CMVC using SCCS, CMVC supports delta versioning but not keyword expansion.

## Version Labels

The PVCS Version Manager enables the user to assign a version label to the revisions. CMVC contains a similar function but uses a different mechanism to accomplish the same result. For this reason, any version label assigned to a PVCS Version Manager file is not used by CMVC. SCCS assigns no version labels.

## PVCS and SCCS Commands

Do not use operating system or version control system commands which modify the contents of or remove the files in the version control directories of your *familyname* account. Using these commands results in discrepancies between the information stored in the relational database and the files stored in these directories. Use only CMVC commands to maintain the files in the version control directories.

---

## Customizing Your Environment to Use the PVCS Version Manager

The following steps should be performed to configure PVCS Version Manager and prepare your environment for use of CMVC.

### Installing PVCS Version Manager

Install PVCS Version Manager in setuid mode, with primary group **system** (the same as each CMVC family account). After the PVCS Version Manager executables are installed, make a directory, titled **sem**, relative to the directory where the executables are stored. That is, if the PVCS Version Manager executables are stored in **/usr/pvcs**, enter the following command:

```
mkdir /usr/pvcs/sem
```

This directory will be used to store all the semaphore files (logfile, journal) that are created and used by PVCS Version Manager. Log in as root, modify the directory permissions to 775, and set the group of the directory to **system** so that the CMVC families that operate within the system group can write to the **/usr/pvcs/sem** directory.

### Resetting the PVCS Version Manager Executables

Reset the PVCS Version Manager executables to use only the default configuration parameters supplied with the PVCS Version Manager. To do this, log in to the user ID which owns the PVCS Version Manager executables and enter the following:

```
vconfig -u /usr/pvcs/*
```

If you installed the executables in a directory other than **/usr/pvcs**, replace **/usr/pvcs** with the name of that directory.



## Configuring the PVCS Version Manager Executables

Refer to your PVCS documentation to gain an understanding of the various PVCS Version Manager configuration options and procedures. Both a master configuration file and a local configuration file were installed with CMVC. These files are in the `/usr/lpp/cmvc/install` directory.

### The Master Configuration File

The master configuration file is called `cmvc_master.cfg`. You must use the master configuration file in order to use PVCS Version Manager as the version control system for CMVC. The file contains the PVCS Version Manager configuration parameter settings that are compatible with the operation of CMVC. You must not alter the values in the master configuration file, except to edit the line `SEMAPHOREDIR=/usr/pvcs/sem` if the path is not correct for your system.

Figure 3 on page 26 is a copy of the master configuration file, `cmvc_master.cfg`. If, for any reason, your version of the `cmvc_master.cfg` file is corrupted use Figure 3 to recreate a master configuration file.

## Using vconfig to Configure the PVCS Executables

You must configure the PVCS Version Manager executables to use the master configuration file for operation of CMVC with the PVCS Version Manager software. To accomplish this, log in to the user ID that owns the PVCS Version Manager executables and issue the following command:

```
vconfig -sn -c/usr/lpp/cmvc/install/cmvc_master.cfg /usr/pvcs/*
```

This will configure all PVCS Version Manager executables in the `/usr/pvcs` directory to treat the file `/usr/lpp/cmvc/install/cmvc_master.cfg` as the master configuration file.

Do not delete the master configuration file or alter it in any way once the `vconfig` command has been issued. The `vconfig` command embeds only the name of a master configuration file within the PVCS Version Manager executables, not the contents. Therefore, the master configuration file itself must always be available for each invocation of any executable so configured.

## Registering Users to the PVCS License Administration Database

The CMVC family name and all CMVC users must be registered as licensed users of PVCS. This generally happens automatically; but if you need to register users manually, you use the PVCS `ladmin` command. For example, enter the following command:

```
ladmin -a<familyName> PVCS serial_number
```

Refer to your PVCS documentation for information on the `ladmin` command.

Also, you must register the uid being specified in the `-uid` attribute flag of the `Release -extract` or `Level -extract` command as the licensed user of PVCS. This uid will either be registered automatically to the PVCS administration database or it must be registered manually depending on how PVCS was configured on your CMVC server. If you must manually register the uid to the PVCS administration database, you must obtain the user name represented by that uid on the CMVC server and add that user name to the PVCS administration database.

---

```

# PVCS Master Configuration File for use with CMVC
#
# -----
# CMVC handles access control for files. There
# is no need for the Enhanced Access Control Database or Access
# List.
#
DISALLOW ACCESSDB ACCESSLIST
# -----
# CMVC handles branching of files. PVCS version
# labels are ignored by CMVC.
#
DISALLOW BASEVERSION BRANCHVERSION DEFAULTVERSION
# -----
# Use the default settings: CASE, CHECKLOCK, COMMENTPREFIX=" "
#
#         DELETEWORK, NOEXCLUSIVELOCK,
#         NOFORCEUNLOCK, NOIGNOREPATH,
#         LOGSUFFIX=??v__, MESSAGESUFFIX=??@__,
#         LOGWORK, NEWLINE="\r\n",NOSIGNON,
#         VERBOSE, VCSDIR=cwd
# Disallow the resetting of these default values.
#
DISALLOW NOCASE NOCHECKLOCK COMMENTPREFIX NODELETEWORK
DISALLOW EXCLUSIVELOCK FORCEUNLOCK IGNOREPATH LOGSUFFIX
DISALLOW MESSAGESUFFIX NOLOGWORK NEWLINE OWNER REFERENCEDIR
DISALLOW SIGNON QUIET NOWRITEPROTECT VCSDIR
# -----
# To permit PVCS logfiles to have both multiple locks per
# revision and multiple locks per user, specify MULTILOCK with
# no parameters and disallow the NOMULTILOCK configuration
# parameter.
#
MULTILOCK
DISALLOW NOMULTILOCK
# -----
# To conserve space, delete the message file after the file is
# checked back into PVCS.
#
DELETEMESSAGEFILE
DISALLOW NODELETEMESSAGEFILE
# -----
# PVCS validates logins by HOST, VCSID and UNKNOWN.
# Disallow the resetting of these to maintain security.
#
LOGIN HOST,VCSID,UNKNOWN
DISALLOW LOGIN
# -----
# Indicate the directory to be used for SEMAPHORE files.
# Disallow the resetting of this directory. Note that this
# directory must be created if it does not already exist. The
# CMVC family must have write access to this directory.
#
SEMAPHOREDIR=/usr/pvcs/sem
DISALLOW SEMAPHOREDIR
# -----
ENDMASTER

```

---

Figure 3. Master Configuration File, *cmvc\_master.cfg* for CMVC/6000

Figure 4 illustrates an example of which user name must be registered to the PVCS administration database:

---

Uid	User Name on the CMVC Server To Be Registered to the PVCS Administration Database	User Name on the Workstation Corresponding to the -uid Attribute Flag
222	Srv222	Clt222

---

Figure 4. Registering a User Name to the PVCS Administration Database

In this example, you must register Srv222, not Clt222, to the PVCS administration database.

---

## Optional Customization of Your Environment

You can customize your PVCS environment for each CMVC family by creating individual local configuration files containing PVCS configuration parameters.

### PVCS Configuration Parameters

For a description of all PVCS Version Manager parameters, refer to your PVCS documentation. The following are optional configuration parameters that can be set within the local configuration file:

#### Compression

You can specify whether deltas and workfiles are compressed when stored in PVCS Version Manager. There are three parameters that control compression:

- **[NO]COMPRESS**
- **[NO]COMPRESSDELTA**
- **[NO]COMPRESSWORKIMAGE**

The advantage of compression is that space is conserved. The disadvantage is that it may take longer to *check out* or *check in* a file because additional processing will be performed. The default is **NOCOMPRESS**.

#### PVCS Keywords

You can specify whether keywords embedded in text files are expanded when the files are checked out or extracted from the PVCS Version Manager. The parameter that controls keyword expansion is **[NO]EXPANDKEYWORDS**. The default is **EXPANDKEYWORDS**. Regardless of the setting of this parameter, the CMVC user can control keyword expansion on an individual file basis from the CMVC client. For more information on PVCS Version Manager keywords, refer to your PVCS documentation. A list of supported PVCS Version Manager keywords is included in the *IBM CMVC User's Reference*.

#### Semaphores

You can set the following PVCS Version Manager configuration parameters that are related to PVCS Version Manager semaphores:

- **SEMAPHOREDELAY [=] *number***
- **SEMAPHORERETRY [=] *number***

The default semaphore delay is 1 second and the default semaphore retry is 3. For more information about PVCS Version Manager semaphores, refer to your PVCS documentation.

### Translate

You can set the PVCS Version Manager configuration parameter so that the end-of-line sequence within text files is translated from line-feed to carriage-return plus line-feed. The parameter that controls translation is **[NO]TRANSLATE**.

The default is **TRANSLATE**.

### Working Directory

You can set the PVCS Version Manager configuration parameter that controls where PVCS Version Manager will create temporary files. The parameter that controls the working directory is **WORKDIR [=] *directory***.

The default is the *current working directory*. The PVCS executables must have access to write to the directory that is specified.

### Journal

You can set the PVCS Version Manager configuration parameter that controls whether PVCS Version Manager makes journal entries for every commit action taken by a PVCS Version Manager command that modifies a logfile. The parameter that controls journal entries is **JOURNAL** and can be set in one of two ways:

- **JOURNAL [= < *journaldir* >]**
- **JOURNAL [= < *journalfile* >]**

If this parameter is not set then no journal entries will be made. The PVCS executables must have access to write to the directory or file that is specified.

## Creating a Local Configuration File

Creating a local configuration file involves the home directory of the account you will create in Chapter 6, “Creating a CMVC Family” on page 31. Return to this section after you have configured your CMVC family.

If you, as the family administrator, choose to set optional configuration parameters for PVCS Version Manager, then these parameters will be set for all users within your CMVC family. Therefore, you must decide which settings will be appropriate for all users of each CMVC family.

To assign values to these parameters, a local configuration file must be created and placed in the *familyname* account home directory on the CMVC server. If you have more than one CMVC family, the local configuration file can be different for each family, if so desired.

A local configuration file, **cmvc\_local.cfg**, exists in the **/usr/lpp/cmvc/install** directory. This is a sample of what a local configuration file can contain and does not necessarily have to be established for your CMVC family. If you wish to use a local configuration file, copy it to the *familyname* account's home directory and modify it to suit the needs of the CMVC family. This is a text file that can be created or modified by a text editor and can be renamed as appropriate.

Figure 5 shows the local configuration file shipped by IBM.

---

```
#
# PVCS Local Configuration File for use with CMVC
#
COMPRESS
SEMAPHOREDELAY=60
SEMAPHORERETRY=3
```

---

*Figure 5. Local Configuration File, cmvc\_local.cfg for CMVC/6000*

When a local configuration file is created, the VCSCFG environment variable must be set so that PVCS Version Manager reads the contents of the local configuration file when PVCS Version Manager executables are invoked. The command:

```
VCSCFG=$HOME/cmvc_local.cfg
export VCSCFG
```

should be added to the family's environment to set the VCSCFG environment variable. Note that *\$HOME* should be replaced with the name of the directory on the CMVC server where the local configuration file resides. The PVCS executables must have access to read the local configuration file.

## Modifying Your Local Configuration Files

The configuration parameters that control compression, keyword expansion and translation are PVCS Version Manager attribute parameters. The value of these parameters is set in the logfile when a file is created. If you modify the local configuration file and alter these parameters after they have been in use for some time, the new settings will only be in effect for newly created logfiles. If you wish to modify the behavior of existing logfiles so that they match the new settings in your local configuration file, then the PVCS Version Manager **vcs** command must be invoked against the existing logfiles in the CMVC family's version control directories. For more information on the **vcs** command, refer to your PVCS documentation.



---

## Chapter 6. Creating a CMVC Family

A single installation of the CMVC server can support multiple CMVC families. You must create each CMVC family from the operating system shell on the workstation where the CMVC server software is installed.

This chapter explains what to do to prepare for configuring your environment. It also describes in detail the configuration activities you must perform when creating each CMVC family.

---

### Preconfiguration Planning

All CMVC objects in a family are related implicitly to one another and cannot relate to CMVC objects in another family. Each family has its own set of users, data, and its own organizational structure.

Within your CMVC environment, you can create one or more families. The projects you group within each CMVC family must be planned carefully. Consider the following issues:

- Group all development projects that share source data within the same family. Each family has its own database tables and its own version control (**vc**) tree; information cannot be shared between families.
- Consider where each development project is heading in future releases. Will a project depend upon other projects? If so, group these projects within the same family because they must share information.
- Designate a server that can handle your future needs. As a family grows, so do the database tables and the **vc** file system.

To help you estimate the size of the file systems that you will create, Figure 6 provides sizes of three sample families for CMVC/6000.

Item name	Algorithm	Family A	Family B	Family C
Number of files	x	30000	10000	5000
Number of defects	0.20x	6000	2000	1000
Size of \$HOME	25000x	750MB	250MB	125MB
Size of SQL tables and indexes	5000x	150MB	50MB	25MB

x is the number of files.

Figure 6. File System Size Estimates of Sample Families for CMVC/6000

**Note:** The estimates in Figure 6 do not include the space required for the relational database nor the CMVC executable files.

## Prerequisite Tasks or Conditions

The CMVC server code and the supported DB2/6000, ORACLE6, ORACLE7, INFORMIX, or SYBASE relational database must be installed. The relational database you are using must be running before you continue with the configuration of the CMVC server software. For information on installing CMVC, see Chapter 3, "Installing the CMVC Servers" on page 13.

---

## Configuring the CMVC Server

The family administrator usually configures the CMVC server for each family. As the family administrator you must have root access to the CMVC server. Configuring the CMVC server involves both the configuration as root and the configuration as the *familyname* user.

Figure 7 summarizes the configuration activities you must perform.

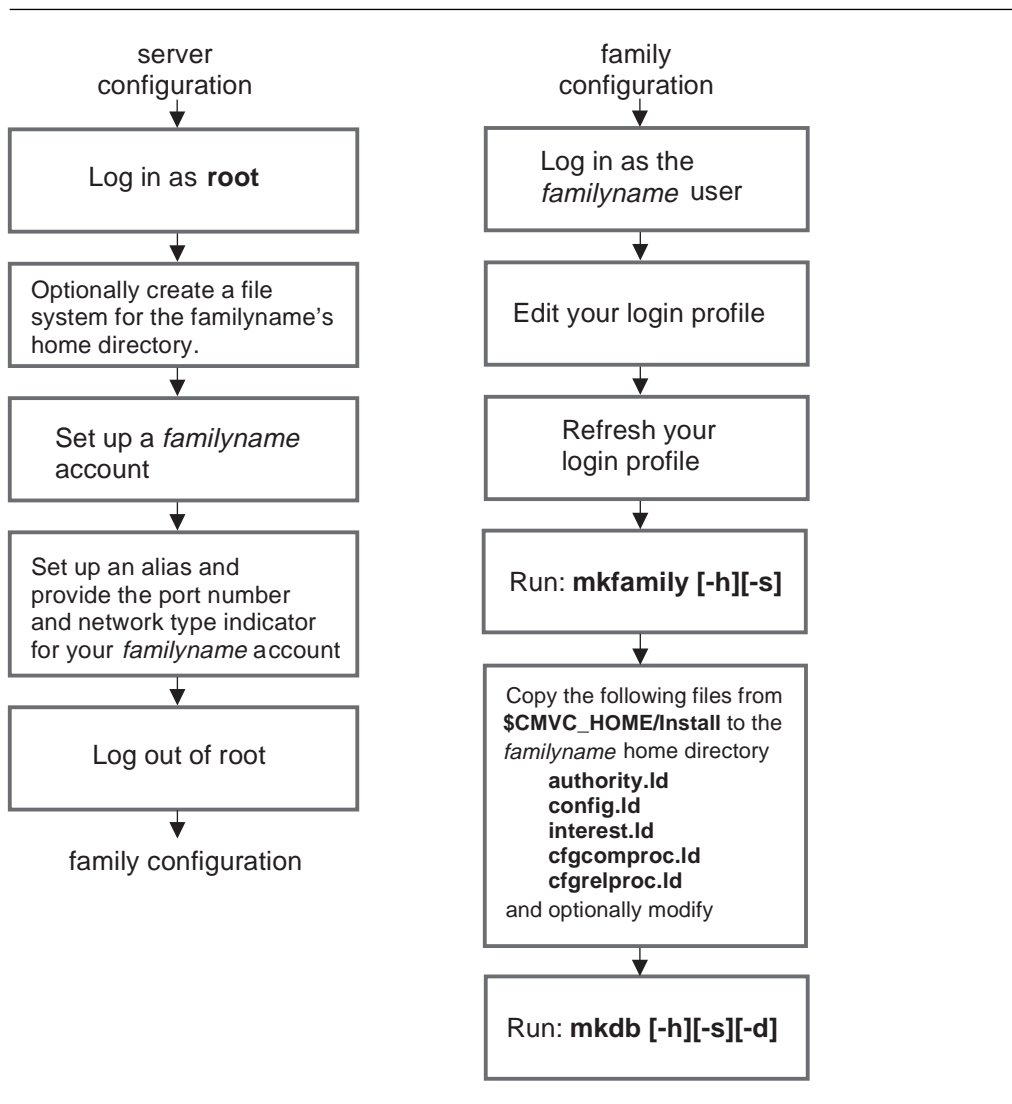


Figure 7. Configuration Flowchart for the CMVC Server



## Configuration as Root

Log in as root and perform the following steps:

1. Read the file `/usr/lpp/cmvc/install/README.first` before proceeding.
2. Optionally create and mount a file system for the home directory of the family account that you create in step 3.
3. Choose a name for your CMVC family, and create an account for that name with the primary group of ID 0 to prevent other groups from accessing files in the CMVC family. For example, the primary group of ID 0 on AIX systems is `system`; the group of ID 0 in SunOS and Solaris systems is `root`.

If you are using a DB2/6000 server:

- Add the DB2 group ID as a secondary group for the CMVC family account. Set the group of the family user ID to that of the owner of the DB2/6000 database instance that the CMVC family will use. For example, if the group ID of the DB2/6000 database instance owner `db2` is `sysadm`, the group ID of the CMVC family should be `sysadm`.
  - Ensure that the CMVC family name does not contain uppercase characters.
  - The length of the password must be less than or equal to 18 characters.
4. Verify that the *home directory* for your CMVC family has the following properties:

**name**      The home directory path

**mode**      755

**owner**     The *familyname* created in step 3

**group**     ID 0 (and if DB2, the group of the owner of the DB2/6000 database instance)

5. Add the *familyname* as an alias of the CMVC server workstation in your TCP/IP naming environment. Add the name through the network information system, a name server, or by changing the entry in your `/etc/hosts` file.

For example, the family called `cmvcfam1` is defined as an alias to the machine `cmvcserv`, which is the designated CMVC server workstation. The family is defined as follows:

```
9.25.16.12 cmvcserv cmvcfam1
```

**Note:** This example is for `/etc/hosts` only.

6. Add an entry to the end of the `/etc/services` file containing your *familyname*, the TCP/IP port number for your family, and the network type indicator `tcp`. For example, the following line establishes the port number for the family `cmvcfam1`:

```
cmvcfam1 1221/tcp
```

The number 1221 is arbitrary, but it must be unique and it must match the entry in the `/etc/services` file on each client workstation.

7. Log out of **root**.

## Configuration as the Familyname User

Now that you have created an account for your family, perform the following steps:

1. Log on to the CMVC family account using the family user ID.
2. For security reasons, change the permissions of your *login profile* to 700.
3. Add the mandatory environment variables to your login profile, including those for your database, as described in the section “Mandatory Environment Variables in Your Login Profile” on page 35.
4. Add the optional environment variables for your database, as described in the section “Optional Environment Variables in Your Login Profile” on page 39.
5. If you are using PVCS, you can create a local configuration file for optional configuration parameters. See “Creating a Local Configuration File” on page 28 for more information about the optional parameters for a PVCS environment.
6. Refresh your login profile to pick up the new environment variables.
7. Run the **mkfamily** command, as described in “Using the mkfamily Command” on page 41.
8. Copy the following files from **/usr/lpp/cmvc/install** into the *familyname* account’s home directory:
  - **authority.ld**
  - **config.ld**
  - **interest.ld**
  - **cfgcomproc.ld**
  - **cfgrelproc.ld**
9. Optionally configure additional fields for the defect, feature, file, and user objects. See Chapter 7, “Configurable Fields” on page 47 for more information.
10. Optionally configure your components and releases. See Chapter 8, “Configuring Components and Releases” on page 59 for more information.
11. Optionally configure your authority groups. See Chapter 9, “Configuring Authority Groups” on page 67 for more information.
12. Optionally configure your interest groups. See Chapter 10, “Configuring Interest Groups” on page 73 for more information.
13. Optionally modify the configuration database table. See Chapter 11, “Modifying Your Configuration Table” on page 77 for more information.
14. Optionally define your user exists. See Chapter 12, “Providing User Exits” on page 87 for more information.
15. Run the **mkdb** command as described in “Using the mkdb Command” on page 43.
16. Start the CMVC server daemons for each CMVC family. See Chapter 13, “CMVC Server Daemons” on page 95 for more information.

## Mandatory Environment Variables in Your Login Profile

Add the following mandatory environment variables to your login profile:

Environment Variable	Description
<b>PATH</b>	<p>Add <code>/usr/lpp/cmvc/install</code>, <code>/usr/lpp/cmvc/bin</code>, <code>/usr/lpp/cmvc/samples</code> and <code>\$HOME/bin</code>, where <code>\$HOME</code> is the home directory of the CMVC family's account.</p> <p>If you are using PVCS, add the directory name that contains the PVCS executable files to access the PVCS executable files from the CMVC family directories.</p>
<b>CMVC_HOME</b>	<p>Add the <code>/usr/lpp/cmvc</code> directory.</p> <p><b>Example:</b> <code>export CMVC_HOME=/usr/lpp/cmvc</code></p>
<b>CMVC_SUPERUSER</b>	<p>Add the CMVC user ID of the first CMVC user to be created.</p> <p><b>Example:</b> <code>export CMVC_SUPERUSER=johndoe</code></p>
<b>CLIENT_HOSTNAME</b>	<p>Add the client host name of the first CMVC user to be created.</p> <p><b>Example:</b> <code>export CLIENT_HOSTNAME=cmvcc1t1</code></p>
<b>CLIENT_LOGIN</b>	<p>Add the client login of the first CMVC user to be created.</p> <p><b>Example:</b> <code>export CLIENT_LOGIN=jdoe</code></p>
<b>CMVC_VCBIN</b>	<p>Add the full path to either the SCCS or the PVCS executable files, depending on the <i>version control</i> mechanism you have selected to use.</p> <p>If you are using SCCS and have installed the executable files in the <code>/usr/bin</code> directory:</p> <p><b>Example:</b> <code>export CMVC_VCBIN=/usr/bin</code></p> <p>For Solaris: <code>export CMVC_VCBIN=usr/ccs/bin</code></p> <p>If you are using PVCS and have installed the executable files in the <code>/usr/pvcs</code> directory:</p> <p><b>Example:</b> <code>export CMVC_VCBIN=/usr/pvcs</code></p>
<b>CMVC_VCTYPE</b>	<p>Indicate the version control mechanism you have selected. Use lowercase letters when defining the type.</p> <p><b>Examples:</b></p> <pre>export CMVC_VCTYPE=pvcs export CMVC_VCTYPE=sccs</pre>
<b>CMVC_VCLOGIN</b>	<p>If you are using PVCS, indicate the name of the user on the CMVC server who owns the PVCS executable files.</p> <p><b>Example:</b> <code>export CMVC_VCLOGIN=pvcs</code></p>

**Note:** The environment variables `CMVC_SUPERUSER`, `CLIENT_HOSTNAME`, and `CLIENT_LOGIN` initiate the first CMVC user ID for this family. These variables are used only when the **mkdb** command is run to initialize the database.

The first CMVC user for each family is automatically a CMVC superuser and can therefore create user IDs for any other users.

The following sections discuss the various mandatory environment variables that you can add to your login profile that pertain to your relational database.

### For a DB2/6000 Server

If you are using a DB2/6000 server, add the following environment variables to your login profile:

Environment Variable	Description
<b>DB2INSTANCE</b>	The DB2 instance name that the CMVC family uses. <b>Example:</b> export DB2INSTANCE=db2
<b>DB2_HOME</b>	The home directory of the DB2 instance that the CMVC family uses. <b>Example:</b> export DB2_HOME=/u/db2
<b>DB2_PASS</b>	The password of the UNIX CMVC family user. <b>Note:</b> This is not the password of the DB2 administration user. The length of the family's AIX password cannot be longer than 18 characters. <b>Example:</b> export DB2_PASS=cmvcdev1
<b>DB2_DBPATH</b>	The path on which to create the CMVC database. If you do not specify a path, the database is created in the default path, that is, the HOME directory of the owner of DB2/6000 instance that the family is using. The pathname can have a maximum of 215 characters. For more information, refer to the <i>DB2/6000 Command Reference</i> book. <b>Note:</b> Ensure that the DB2/6000 instance owner has write permission to DB2_DBPATH. The CMVC database requires at least 15 megabytes of hard disk space in the file system. <b>Example:</b> export DB2_DBPATH=/u/db2/dbspace/cmvc23
<b>PATH</b>	Include the following search paths: \$CMVC_HOME/bin \$CMVC_HOME/install \$CMVC_HOME/samples \$DB2_HOME/sql11ib/bin \$DB2_HOME/sql11ib/adm

### For an ORACLE Server

If you are using an ORACLE relational database, add the following environment variables to your login profile.

Environment Variable	Description
<b>ORACLE_HOME</b>	The ORACLE home directory that the CMVC family uses. <b>Example:</b> export ORACLE_HOME=/usr/oracle

Environment Variable	Description
<b>ORACLE_DBA</b>	<p>The user ID and password of the ORACLE DBA.</p> <p><b>Note:</b> The ORACLE_DBA environment variable is needed only for the <b>mkdb</b>, <b>chfield</b>, and <b>rmdb</b> commands. You can remove this environment variable from your login profile after these commands run successfully.</p> <p><b>Example:</b> <code>export ORACLE_DBA=system/manager</code></p>
<b>ORACLE_PASS</b>	<p>The password of the UNIX CMVC family user.</p> <p><b>Note:</b> This is not the password of the ORACLE administration user.</p> <p><b>Note:</b> The ORACLE_PASS environment variable is needed only for the <b>mkdb</b>, <b>chfield</b>, and <b>rmdb</b> commands. You can remove this environment variable from your login profile after these commands run successfully.</p> <p>No password has yet been established; choose a password for your family at this point. This password will be used when you run <b>mkdb</b> in step 15 on page 34.</p> <p><b>Example:</b> <code>export ORACLE_PASS=cmvcdev1</code></p>
<b>ORACLE_SID</b>	<p>The ORACLE SID.</p> <p><b>Example:</b> <code>export ORACLE_SID=SID</code></p>

### For an INFORMIX Server

If you are using an INFORMIX relational database, add the following environment variable to your login profile:

Environment Variable	Description
<b>INFORMIXDIR</b>	<p>Name of the INFORMIX home directory.</p> <p><b>Example:</b> <code>export INFORMIXDIR=/usr/informix</code></p>

See “Mandatory Environment Variables in Your Login Profile” on page 35 for more information on the other mandatory environment variables for the login profile.

### For a SYBASE Server

If you are using a SYBASE relational database, add the following environment variables to your login profile:

Environment Variable	Description
<b>SYBASE</b>	<p>Name of the SYBASE home directory.</p> <p><b>Example:</b> <code>export SYBASE=/usr/sybase</code></p>

Environment Variable	Description
<b>SYBASE_PASS</b>	<p>The password of the UNIX CMVC family user.</p> <p><b>Note:</b> This is not the password of the SYBASE administration user.</p> <p>No password has yet been established; choose a password for your CMVC family at this point. This password is used when you run <b>mkdb</b> in step 15 on page 34.</p> <p><b>Example:</b> <code>export SYBASE_PASS=cmvcdev1</code></p>
<b>SYBASE_SA_PASS</b>	<p>Password of the SYBASE System Administrator (SA) login.</p> <p><b>Note:</b> The SYBASE_SA_PASS environment variable is needed only for the <b>mkdb</b>, <b>chfield</b>, and <b>rmdb</b> commands. You can remove this environment variable from your login profile after these commands run successfully.</p> <p><b>Example:</b> <code>export SYBASE_SA_PASS=sapasswd</code></p>

If you have multiple SYBASE SQL servers, by default, the server with the name of SYBASE is used. Optionally, you can set the SYBASE server name in the DSQUERY environment variable to the SYBASE server containing the CMVC relational database, for example:

```
export DSQUERY=SYBASE
```

**Reminder:** Return to step 4 on page 34 to continue configuring the CMVC server.

## Optional Environment Variables in Your Login Profile

CMVC creates its own tables and indexes in the database. You can use the optional environment variables described below to have these tables and indexes created in the previously created tablespaces (the ORACLE database), dbspaces (the INFORMIX database), or database devices (the SYBASE database) rather than the system default ones.

### For CMVC

For CMVC, the following environment variable is optional:

Environment Variable	Description
<b>CMVC_BINARY_THRESHOLD</b>	Used for binary delta versioning in SCCS only. This variable specifies a threshold of the maximum number of different contiguous bytes between two binary files. When the threshold is reached, the entire binary file is stored instead of being versioned.  In case of performance degradation, you can set this environment variable to a lower value. If you do not specify this variable, the default value of 15000 bytes is used.  <b>Example:</b> <code>export CMVC_BINARY_THRESHOLD=15000</code>

### For DB2/6000 Server

If you are using the DB2/6000 relational database, you can add in your profile the **db2profile** script (for Bourne and Korn shells) or the **db2cshrc** script (for C shell) to setup the environment for DB2.

### For ORACLE Server

If you are using the ORACLE relational database, by default, these tables and indexes are created in the default tablespace, for example, SYSTEM. Optionally, you can have these tables and indexes stored in previously created tablespaces by setting the following environment variables:

Environment Variable	Description
<b>ORACLE_TBLSP</b>	Add this for the ORACLE tables.  <b>Example:</b> <code>export ORACLE_TBLSP=mytblsp</code>
<b>ORACLE_NDXSP</b>	Add this for the ORACLE indexes.  <b>Example:</b> <code>export ORACLE_NDXSP=myndxsp</code>

### For INFORMIX Server

If you are using the INFORMIX relational database, by default, these tables and indexes are created in a database stored in the default *dbspace*, for example, rootdbs. Optionally, you can have the database stored in a previously created *dbspace* by setting the following environment variable:

Environment Variable	Description
<b>INFORMIX_DBSP</b>	Add this for the INFORMIX dbspace.  <b>Example:</b> <code>export INFORMIX_DBSP=mydbsp</code>

## For SYBASE Server

If you are using the SYBASE relational database, by default, the CMVC family database and log are created in the SYBASE default device, for example, master. Optionally, you can have the database or log created in previously created devices by setting the following environment variables:

Environment Variable	Description
<b>SYBASE_DBDEV</b>	Add this for the SYBASE database device. <b>Example:</b> <pre>export SYBASE_DBDEV="devName = n, devName = n ..."</pre>
<b>SYBASE_LOGDEV</b>	Add this for the SYBASE log device. <b>Example:</b> <pre>export SYBASE_LOGDEV="devName = n, devName = n ..."</pre> <p>Where:</p> <p><i>devName</i> is a SYBASE device name <i>n</i> is the device size in megabytes.</p>

**Note:** The SYBASE\_DBDEV (database device) must have a minimum of 5 megabytes of disk space. If **SYBASE\_LOGDEV** (log device) is not set, the SYBASE log is created in the SYBASE database device you defined, in **SYBASE\_DBDEV**, or in master by default.

**Reminder:** Return to step 5 on page 34 to continue configuring the CMVC server.



## Using the `mkfamily` Command

The `mkfamily` shell script creates the mandatory subdirectories listed in Figure 8 as local file systems, relative to the home directory of your CMVC family account. If the `$HOME/bin` directory does not exist, run the `mkfamily` command to create it.

If you want to create the subdirectories as separate file systems, manually create the subdirectories and set their owner to your CMVC family name and their group to ID 0 or `SYSADM` if needed. Set their file mode to 750, except for the `vc` directory, which must have a file mode of 755.

Mandatory Subdirectory	Description
<code>vc</code>	Contains the versioned files and deltas (also referred to as the version control (vc) file system).
<code>audit</code>	Contains the audit log of CMVC transactions.
<code>queue</code>	Contains the instruction files for notification delivery.
<code>queue/messages</code>	Contains the message files for notification delivery.
<code>maps</code>	Contains the level maps.
<code>config</code>	Contains the <i>user exit</i> definition file. A sample of the file is in <code>/usr/lpp/cmvc/install/userExits</code> . Copy the <code>userExits</code> file to the <code>config</code> subdirectory if you created <code>config</code> manually.

Figure 8. CMVC Mandatory Subdirectories

The `mkfamily` shell script creates a local file system structure for the *familyname* account created in step 3 of “Configuration as Root” on page 33. The syntax for the `mkfamily` command is shown in Figure 9.

### Syntax:

---

```
mkfamily [-h] [-s] [familyname]
```

---

Where:

<code>-h</code>	Indicates help is needed
<code>-s</code>	Runs the command in silent (no prompt) mode
<code>familyname</code>	Names the CMVC family

---

Figure 9. Syntax for the `mkfamily` Command

**Note:** If this shell script fails, make sure that you either choose another *familyname* or run the `rmfamily` command before running the `mkfamily` command again. See “Using the `rmfamily` Command” on page 45 for a description of the `rmfamily` command.

If you are using PVCS, do the following:

1. Log in to the root account on the CMVC server.
2. Issue the following command:

```
chown -R <pvcsOwner>.system /<familyHomeDir>/vc
```

Where *pvcsOwner* is the name of the user on the CMVC server who owns the PVCS executable files (for example, *pvcs*), and *familyHomeDir* is the home directory of your family.

3. Log out of root.

**Reminder:** Return to step 8 on page 34 to continue configuring the CMVC server.

## Using the **mkdb** Command

The **mkdb** shell script creates and initializes the CMVC family database tables, *views*, and indexes. For information on configuring the CMVC database tables, see Chapter 7, “Configurable Fields.” The syntax for the **mkdb** command is shown in Figure 10.

### Syntax:

---

**mkdb** [-h] [-s] [-d] [familyname]

---

Where:

<b>-h</b>	Indicates help is needed.
<b>-s</b>	Runs the command in silent (no prompt) mode.
<b>-d</b>	Loads the default configurable fields shipped by IBM for the Defect and Feature tables. Existing configurable field properties are cleared. If the <b>-d</b> flag is omitted, the existing configurable fields are installed.
<b>familyname</b>	Names the CMVC family.

---

Figure 10. Syntax Statement for the **mkdb** Command

**Note:** If the **mkdb** shell script is not successful, you might have to run the **rmdb** script before running the **mkdb** script again. See “Using the **rmdb** Command” on page 46 for a description of the **rmdb** command.

Depending on the database you use, the following files are created when you run the **mkdb** command:

- **ORACLE**

Files are created in the family account’s home directory when the **mkdb** command uses the SQL\*Loader\*\* utility to load the Authority, Config, Interest, and Configurable Process tables.

Each of the following files contains all records that the SQL\*Loader utility attempted to insert but could not:

- **bad.authority**
- **bad.config**
- **bad.interest**
- **bad.cfgcomproc**
- **bad.cfgrelproc**

Each of the following files contains a detailed summary of the SQL\*Loader execution:

- **authority.log**
- **config.log**
- **interest.log**
- **cfgcomproc.log**
- **cfgrelproc.log**

For more information about the contents of these files, refer to your ORACLE documentation.

- **SYBASE**

Files are created in the family account's home directory when the **mkdb** command uses the **bcp** command to load the Authority, Config, Interest, and Configurable Process tables. Each of the following files is SYBASE's **bcp** error file:

- **authority.log**
- **config.log**
- **interest.log**
- **cfgcomproc.log**
- **cfgrelproc.log**

For more information about the contents of these files, refer to your SYBASE documentation.

- **INFORMIX**

The **mkdb** script displays an error log when errors occur.

- **DB2**

The **mkdb** script displays an error log when errors occur.

**Reminder:** Return to step 9 on page 34 to continue configuring the CMVC server.

## Using the **rmfamily** Command

If you are using PVCS and you want to use the **rmfamily** script to remove the directories that have been created for your CMVC family, do the following:

1. Log in to the root account on the CMVC server.
2. Issue the following command:

```
chown -R <familyname>.system /<familyHomeDir>/vc
```

Where *familyname* is the name of your CMVC family and *familyHomeDir* is the home directory of your family.

3. Log out of root.

The **rmfamily** shell script removes the file system structure for the specified *familyname* account. Run this command while you are logged in to the *familyname* account. The syntax for the **rmfamily** command is shown in Figure 11.

### Syntax:

---

```
rmfamily [-h] [-s] [familyname]
```

---

Where:

<b>-h</b>	Indicates help is needed
<b>-s</b>	Runs the command in silent (no prompt) mode
<b>familyname</b>	Names the CMVC family

---

Figure 11. Syntax Statement for the **rmfamily** Command

Use the **rmfamily** command in the following cases:

- When the **mkfamily** command fails after it creates part of the file system structure for the newly created family. After you correct the problem that caused the **mkfamily** command to fail, run the **rmfamily** command to remove the partially created file system structure, and run the **mkfamily** command again.
- You no longer need the data in the file system structure for an existing family. After you back up the files, run the **rmfamily** command to remove the file system structure created by the **mkfamily** command.

**Warning:** The contents of your family is destroyed when you run the **rmfamily** command.

## Using the `rmdb` Command

The `rmdb` shell script removes the CMVC database tables, views, and indexes for the specified *familyname* account. Run this command while you are logged in to the *familyname* account. The syntax for the `rmdb` command is shown in Figure 12.

---

```
rmdb [-h] [-s] [-f] [familyname]
```

---

Where:

<b>-h</b>	Indicates help is needed
<b>-s</b>	Runs the command in silent (no prompt) mode
<b>-f</b>	Forces removal of the database if another <i>familyname</i> with the same letter sequence was found
<b>familyname</b>	Names the CMVC family

---

Figure 12. Syntax for the `rmdb` Command

Use the `rmdb` command in the following cases:

- When the `mkdb` command fails after it creates part of the CMVC family database tables, views, and indexes. After you correct the problem that caused the `mkdb` command to fail, run the `rmdb` command to remove the partially created database tables, views, and indexes, and run the `mkdb` command again.

**Note:** The `rmdb` command might produce error messages because it attempts to remove some of the tables, views, and indexes that might not have been created when the `mkdb` command failed.

- You no longer need the data in the CMVC family database tables, views, and indexes. After you back up the database tables, views, and indexes, you can run the `rmdb` command to remove them.

**Note:** The contents of your family will be destroyed when you run the `rmdb` command.

---

## Chapter 7. Configurable Fields

With CMVC, family administrators can configure existing fields in the Defect and Feature tables, or create additional fields for the Defect, Feature, File, and User tables. The customized fields can be used as command attributes, and can be accessed from the GUI as well as displayed in reports. In this way, CMVC users can store information that is customized for their development environment and terminology. For more information on customizing the attributes of CMVC objects, refer to Chapter 11, “Modifying Your Configuration Table” on page 77.

**Warning:** We strongly recommend that you back up the CMVC family account and the corresponding database before attempting to do any changes to the configurable fields.

---

### Overview

You can install the default configurable fields shipped by IBM by running the **mkdb** command with the **-d** flag. For more information on the **mkdb** command, see “Using the mkdb Command” on page 43. The shipped defaults for the Defect and Feature tables contain configurable fields. The CMVC administrator can create additional configurable fields in these tables, and the File and User tables, or modify the existing fields by running the configurable fields installation program, **chfield**. For instructions on running the **chfield** program, see the section “Creating and Modifying Configurable Fields” on page 51.

After configurable fields are created by the system administrator, they are displayed like any predefined field. They can be accessed from the command line or through the GUI. All configurable fields can be used as *search* criteria in the **Report** command.

### Properties of Configurable Fields

The CMVC family administrator can specify the following properties for the configurable fields:

<b>Active</b>	This flag indicates whether the field is active. A value of <b>yes</b> means the field is active, and <b>no</b> means it is inactive.
<b>DB Column Name</b>	This is the column name as defined in the database table.
<b>CMD Attribute</b>	This name is used as the command attribute on the command line.
<b>Field Label</b>	This name appears in the GUI dialog boxes for the <b>Defect</b> and <b>Feature -open</b> and <b>-modify</b> actions, the <b>User -create</b> and <b>-modify</b> actions, and the <b>File -modify</b> action.
<b>Title</b>	This name appears in the header title for the object window. If this name is not provided, the configurable field is not displayed in the object window.
<b>Create</b>	This flag shows whether the field is one of the attributes within the <b>User -create</b> , <b>Defect -open</b> , or <b>Feature -open</b> actions. A value of <b>yes</b> means the field is included as an attribute, and <b>no</b> means it is not included.

<b>Required</b>	This flag indicates whether the field is required for the <b>-create</b> or <b>-open</b> actions. A value of yes means the field is mandatory, and no means it is not mandatory.
<b>Type</b>	This name appears in the type column of the database configuration file <b>config.ld</b> . Default values can be defined in the <b>config.ld</b> file.

## Using Configurable Fields

The following conditions apply to the use of configurable fields:

- All CMVC daemons must be stopped before configurable fields are created or modified.
- Configurable fields for defect, feature, and user objects are effective only on **-create**, **-open**, and **-modify** actions, with the exception of the **File** command, for which they are effective only for the **-modify** action.
- No more than 20 active configurable fields are allowed per object.
- The content of any configurable field cannot exceed 85 characters in length and cannot contain spaces.

**Note:** If you are using the DB2/6000 database server, there is a DB2/6000 row length limitation of 4,005 bytes that limits the number of configurable fields which can be added to certain CMVC objects. The File object can have up to 17 configurable fields, while the Defect and Feature objects can have up to a combined total of 19 configurable fields.

If you are using a SYBASE database, it is recommended that you configure up to a maximum of 10 new fields for any of the CMVC objects supporting configurable fields. Setting up more than 10 configurable fields for each CMVC object may result in a SYBASE error (416).

- No more than 50 fields are allowed in the reports containing the configurable fields.
- The data type of all configurable fields is character.
- Configurable fields are not used by any CMVC process logic. They can only be used for reports.
- The CMVC usage statements apply only to the default settings shipped by IBM. The usage statements do not reflect the updated command usage after the properties of the configurable fields are altered.

---

## Default Configurable Fields Shipped by IBM

Fields in the Feature and Defect tables can be modified. As shipped, they have the default values shown in Figure 13 on page 49, and Figure 16 on page 50.

**Note:** There are no IBM shipped configurable fields for the File table or User table.



## Feature Table

As shipped, the configurable fields in the Feature table have the properties shown in Figure 13.

Active	DB Column Name	CMD Attribute	Field Label	Title	Create	Required	Type
yes	priority	priority	Priority:	Priority	no	no	priority
yes	target	target	Target:	Target	no	no	

Figure 13. Defaults Shipped by IBM for Feature Table Configurable Fields

## Stanza Report Format for the Feature Table

The stanza format shipped by IBM for the Feature table is shown in Figure 14.

```

prefix          %s
name            %s
reference       %s
abstract       %s
duplicate      %s
component      %s

state          %-25.25s  priority    %-20.20s
target        %-25.25s  age         %ld

addDate       %-25.25s  assignDate  %-20.20s
lastUpdate   %-25.25s  responseDate %-20.20s
endDate       %-25.25s

ownerLogin    %-25.25s  originLogin %-20.20s
ownerName     %-25.25s  originName  %-20.20s
ownerArea     %-25.25s  originArea  %-20.20s
  
```

Figure 14. Stanza Report Format Shipped by IBM for the Feature Table

**Note:** To understand the format specification, see “Updating Reports” on page 55 for an example.

## Table Report Format for the Feature Table

The table format shipped by IBM for the Feature table is shown in Figure 15.

```

%-4.4s %-15.15s %-15.15s %-8.8s %-8.8s %-8.8s %-3.3ld %-4.4s %-8.8s %-50.50s
pref name          compName      state    originLo ownerLog age prio target abstract
-----
  
```

Figure 15. Table Report Format Shipped by IBM for the Feature Table

## Defect Table

As shipped, the configurable fields in the Defect table have the properties shown in Figure 16.

Active	DB Column Name	CMD Attribute	Field Label	Title	Create	Required	Type
yes	symptom	symptom	Symptom:		yes	yes	symptom
yes	phaseFound	phaseFound	Phase found:		yes	yes	phase
yes	phaseInject	phaseInject	Phase injected:		no	no	phase
yes	priority	priority	Priority:	Priority	no	no	priority
yes	target	target	Target:	Target	no	no	

Figure 16. Defaults Shipped by IBM for Defect Table Configurable Fields

## Stanza Report Format for the Defect Table

The stanza format shipped by IBM for the Defect table is shown in Figure 17.

```

prefix          %s
name            %s
reference       %s
abstract        %s
duplicate       %s

state           %-25.25s  priority      %-20.20s
severity        %-25.25s  target        %-20.20s
age             %ld

compName        %-25.25s  answer        %-20.20s
release         %-25.25s  symptom       %-20.20s
envName         %-25.25s  phaseFound    %-20.20s
level           %-25.25s  phaseInject   %-20.20s

addDate         %-25.25s  assignDate    %-20.20s
lastUpdate      %-25.25s  responseDate  %-20.20s
endDate         %-25.25s

ownerLogin      %-25.25s  originLogin   %-20.20s
ownerName       %-25.25s  originName    %-20.20s
ownerArea       %-25.25s  originArea    %-20.20s

```

Figure 17. Stanza Report Format Shipped by IBM for the Defect Table

**Note:** To understand the format specification, see “Updating Reports” on page 55 for an example.

## Table Report Format for the Defect Table

The table format shipped by IBM for the Defect table is shown in Figure 18.

<pre>%-4.4s %-15.15s %-15.15s %-8.8s %-8.8s %-8.8s %-3.3s %-3.3ld %-4.4s %-55.55s pref name          compName      state      originLo  ownerLog  sev  age  prio  abstract -----</pre>
--

Figure 18. Table Report Format Shipped by IBM for the Defect Table

## Displaying Configurable Field Properties

The properties of active the configurable fields can be displayed with the following commands:

- **Defect** `-configInfo -family familyName [-raw]`
- **Feature** `-configInfo -family familyName [-raw]`
- **File** `-configInfo -family familyName [-raw]`
- **User** `-configInfo -family familyName [-raw]`

If the `-raw` flag is selected, the information is organized in a fixed ASCII table format as follows:

```
Field Label|Title|Attribute|DB Column Name|Create|Required|Type
```

**Note:** The properties of both the Prefix and Severity non-configurable fields will also be displayed for Defects, whereas the Prefix field will be displayed for Features.

---

## Creating and Modifying Configurable Fields

The family administrator uses the configurable field installation program **chfield** to create or modify the properties of the configurable fields. Use the **chfield** program to modify or create configurable fields for an existing family, to create configurable fields for a new family, or to update reports.

You must stop all CMVC daemons before running **chfield**.

Figure 19 on page 52 displays the syntax for the **chfield** command.

---

**chfield -object Name [-source Name -s]**

---

Where:

<b>object</b>	Indicates the object to configure. The <i>Name</i> parameter must be one of Defect, Feature, File, or User.
<b>source</b>	Indicates where to find the system generated file for the configurable fields. If the <b>-source</b> flag is used, the program reconfigures CMVC according to the information in the specified file. The <i>Name</i> parameter must be either default, an existing family name, or the full path of an existing family name. <b>Note:</b> If only an existing family name is used, then the directory path of the current CMVC family is used.
<b>s</b>	This flag can only be used if the <b>-source</b> flag is selected. The <b>-s</b> flag turns on silent mode. The program is run without a request for user confirmation.

---

Figure 19. Syntax Statement for the *chfield* Command

System files residing at the family home directory are generated or updated after the configurable fields have been installed into CMVC. The system-generated files are:

- For the Defect object:
  - **configField/DefectConfigTbl**
  - **configField/DefectConfigFormat**
- For the Feature object:
  - **configField/FeatureConfigTbl**
  - **configField/FeatureConfigFormat**
- For the File object:
  - **configField/FileConfigTbl**
  - **configField/FileConfigFormat**
- For the User object:
  - **configField/UserConfigTbl**
  - **configField/UserConfigFormat**

**Note:** These system-generated files are maintained by the **chfield** program. Do not modify them manually.

## Running the **chfield** Program with the **-source** Option

You can use this option to import the configurable fields from an existing family (called the *source family*) to your current CMVC family (called the *target family*). The install program uses the information in the system-generated file that corresponds to the object specified by the **-object** flag for the indicated family. The report format is also imported. If default is specified, the corresponding shipped defaults are imported.

If the source family is on a different machine than the target family, copy the system-generated files from the source family for the specified object to the target family. Then run the **chfield** program, as follows:

```
chfield -object <objectName> -source <TargetFamilyName>
```

The program displays information about the configurable fields to be imported. Figure 20 on page 53 shows the display layout.

Active	DB Column Name	CMD Attribute	Field Label	Title	Create	Required	Type
yes	target	target	Target:	Target	no	no	
yes	phaseFound	phaseFound	Phase found:		yes	yes	phase
no	phaseInject	phaseInject	Phase inject:		no	no	phase

You have selected to import the above configurable fields, press [Y] to continue:

Figure 20. The `chfield` Display When Importing Configurable Field Information

If a configuration table is already in the target family, a warning message is displayed. You are prompted to confirm that you want to overwrite the existing settings.

## Running the `chfield` Program with the `-object` Flag Only

If you run `chfield` with only the `-object` flag, the current configurable field information is displayed. You are prompted to select one of the following options:

1. Create a configurable field
2. Modify a configurable field
3. Update report details
4. Display help
5. Quit

### Creating a New Field

If you select the option to create a new field, you are prompted to type the information needed to create the field.

Figure 21 on page 54 shows the display on which you type the configurable field information.

```

Active DB Column Name CMD Attribute Field Label Title Create Required Type
yes target target Target: Target no no
yes phaseFound phaseFound Phase found: yes yes phase
no phaseInject phaseInject Phase inject: no no phase

Select one of the following actions:
1. Create a configurable field
2. Update a configurable field
3. Update report details
4. Display HELP
Q. Exit
Please enter command number [1,2,3,4,Q] : 1

You have selected to create a new field.
Enter DB Column Name : developer
Enter CMD Attribute : developer
Enter Field Label : Developer:
Enter Title :
Is it an attribute for "create/open" action [Y/N] : yes
Is it a required field for "create/open" action [Y/N] : no
Enter the field type :

```

Figure 21. The chfield Display When Configurable Field Information Is Typed

After you type the information, the new configurable fields and the existing configurable fields are displayed and you are prompted to confirm that you want to create the new field. The database tables are then altered.

**Note:** If you are using INFORMIX-Online, creating a new configurable field may be time-consuming. Free disk space in the INFORMIX logs is also needed. You require free space that is twice the size of the table that is related to the object for which you are creating a new configurable field. Figure 22 shows the CMVC object names and their related table names.

---

CMVC Object	CMVC Database Table
<b>Defect</b>	Defects
<b>Feature</b>	Defects
<b>File</b>	Files
<b>User</b>	Users

---

Figure 22. CMVC Objects and Their Related Tables

### Updating an Existing Field

If you choose to update an existing configurable field, you are prompted to enter the updated information.

After you type the information, the updated configurable fields and the unmodified configurable fields are displayed.

## Updating Reports

You can customize the report stanza and table formats. Select the Update stanza and table reports option of the **chfield** program.

You edit report formats using the system editor **vi**, which is started by the **chfield** program when you select the option to update reports.

**Stanza format** The stanza format consists of a format section and a column section (see Figure 24 on page 57).

**Table format** The table format consists of a format section, column section, and header section (see Figure 25 on page 57).

The format section specifies the layout of the report using the string format as used in the C programming language. For example, a format specification of `%-25.25s` indicates the following:

- `%` Start of format specification.
- `-` The output is left-justified. If you do not include this character, the output is right-justified.
- `25` The minimum number of characters (bytes) of output.
- `.25` The maximum number of characters (bytes) printed for all or part of the output field, or minimum number of digits printed for integer values.
- `s` Type of data:
  - `s` for strings
  - `ld` for integers

The column section describes the column name of each of the labels specified in the format section. The columns must appear in the same order in the format and column sections.

If no reports exist for the object corresponding to the **-object** flag, a basic report screen with no configurable fields is displayed.

The report editing screen is divided into 5 sections. The sections are separated by colons (:). Figure 23 on page 56 shows a sample report format for the Defect table after configurable fields have been added. The changes are noted in bold font.

The sections are:

- Stanza format, such as `ViewStanzaFMT`
- Stanza columns, such as `ViewStanzaCOL`
- Table format, such as `ViewTableFMT`
- Table columns, such as `ViewTableCOL`
- Table header, such as `ViewTableHDR`

```

# ViewStanzaFMT
prefix      %s
name        %s
reference    %s
abstract     %s
duplicate    %s

state       %-25.25s  priority    %-20.20s
severity    %-25.25s  target      %-20.20s
age         %ld

compName    %-25.25s  answer      %-20.20s
release     %-25.25s  symptom     %-20.20s
envName     %-25.25s  phaseFound  %-20.20s
level       %-25.25s  phaseInject %-20.20s

addDate     %-25.25s  assignDate  %-20.20s
lastUpdate  %-25.25s  responseDate %-20.20s
endDate     %-25.25s

developer  %-25.25s

ownerLogin  %-25.25s  originLogin %-20.20s
ownerName   %-25.25s  originName  %-20.20s
ownerArea   %-25.25s  originArea  %-20.20s

:
# ViewStanzaCOL
prefix,name,reference,abstract,duplicate,state,priority,severity,target,age,compName,
answer,releaseName,symptom,envName,phaseFound,levelName,phaseInject,addDate,
assignDate,lastUpdate,responseDate,endDate,developer,ownerLogin,originLo gin,ownerName,
originName,ownerArea,originArea
:
# ViewTableFMT
%-4.4s %-15.15s %-15.15s %-8.8s %-9.9s %-8.8s %-8.8s %-3.3s %-3.3ld %-4.4s %-55.55s
:
# ViewTableCOL
prefix,name,compName,state,developer,originLogin,ownerLogin,severity,age,priority,abstract
:
# ViewTableHDR
pref name      compName      state      developer  originLo  ownerLog  sev  age  prio  abstract
-----
:

```

Figure 23. Sample Report Format after Adding Configurable Fields

In Figure 23, the Defect format shipped by IBM has been modified as follows:

- Added a new label in the stanza format section (ViewStanzaFMT), **developer** and the format specification %-25.25s (after endDate)
- Added the column name in the stanza column section (ViewStanzaCOL), **developer** (after endDate)
- Altered the table format section (ViewTableFMT) and put %-9.9s in the corresponding position for the **developer** entry
- Added the column name in the table column section (ViewTableCOL), **developer** (after state)
- Added a new label in the table header section (ViewTableHDR), **developer** (after state) and added the corresponding dashes in the following line

The stanza report format shown in Figure 23 displays the report shown in Figure 24 on page 57. To display the report, enter the command:

```
Defect -view 3168
```



prefix	ptr		
name	3168		
reference	testcase_099		
abstract	Compilation error occurred.		
duplicate			
state	open	priority	
severity	2	target	level_020
age	9		
compName	demoComponent	answer	
release	demoRelease	symptom	compile_failed
envName		phaseFound	prototyping
level	level_019	phaseInject	
addDate	93/04/01 11:32:47	assignDate	93/04/04 18:45:41
lastUpdate	93/04/13 11:54:15	responseDate	93/04/03 11:29:59
endDate			
developer	johnDoe		
ownerLogin	annHar	originLogin	martin
ownerName	Ann Harrison	originName	Martin Karland
ownerArea	Development	originArea	Testing

Figure 24. Sample Stanza Report Displayed after Adding Configurable Fields

The table report format shown in Figure 23 on page 56 displays the report shown in Figure 25. To display the report, enter the command:

Report -view DefectView -where "name='3168'" -table

pref	name	compName	state	developer	originLo	ownerLog	sev	age	prio	abstract
ptr	3168	demoComponent	open	johnDoe	martin	annHar	2	009		Compilation error.

Figure 25. Sample Table Report Displayed after Adding Configurable Fields



---

## Chapter 8. Configuring Components and Releases

For a complete understanding of the CMVC objects discussed in this chapter read *IBM CMVC Concepts*.

Once your family has been created, the initial management and structure needs to be planned. Within each family, data is organized into groups called *components*. Components are the focal point for information retrieval, access control, problem reporting, and data organization. All files that make up a single version of a product are also grouped into categories called releases. Releases are defined separately from components to ease the maintenance of multiple versions of a product. You can define the processes that your users can select for each component or release to tailor CMVC to your requirements.

Planning the component structure, organizing the initial release groupings, and defining the processes to be used are an important part of family administration.

---

### Planning Your Component Structure

Each CMVC family has its own component structure.

Components are the main management entity within your family. The component structure provides hierarchical access control for all CMVC objects.

Careful planning of a component hierarchy that will work best for your organization is the key to the effective use of CMVC. You should plan an initial structure that reflects the composition of your development organization. You can modify this structure as your organization grows or as your needs change. An important part of planning this structure is determining the users who will own each component. Again, this can reflect the composition of your development organization.

---

### Planning Your Release Groupings

Each CMVC family contains unique releases.

Releases are the main project organizational entity within your family. Plan the initial component structure and then plan your release structure by organizing your development files into product-related groups.

Careful planning of the release groupings that will work best for your organization is the key to the effective use of CMVC. You should plan initial releases that reflect the projects under development. You can modify these releases or create new releases as your organization grows or as your projects change. An important part of planning your CMVC environment is determining the users who will own each release and the component that will manage each release. Again, this can reflect the composition of your development organization.

---

## Planning Your Processes

The family administrator chooses the CMVC subprocesses that can be used by the family by assigning a *process* name to a group of one or more CMVC subprocesses. Project or release leaders can choose from the process names when creating or modifying components or releases. The group of CMVC subprocesses appears to users as a single process. CMVC is shipped with preconfigured processes; however, administrators can create new processes, or modify or delete the defaults. You can configure processes by editing the **cfgcomproc.ld** file and the **cfgrelproc.ld** file. See “Configuring Processes” on page 63 for details on how to configure processes for use by your organization.

When configuring processes, you must tailor the configurable processes shipped by IBM to your organization. The configurable processes shipped with CMVC cover a wide range of situations and all processes may not be appropriate for a particular organization.

You can tailor CMVC to suit different development cycles, methodologies, or organizational structures while maintaining control of the processes being used on a family or corporate basis. By selecting the processes that are required for components and releases, you can optimize CMVC for the stage of your development cycle or the size of your organization.

The extra flexibility provided by configurable processes requires some forethought by family administrators, component *owners*, and release leaders. Administrators must notify users of the processes that are appropriate for each stage in their projects.

## Configuring Component Processes

Configure processes for components by selecting the CMVC subprocesses that users can apply when creating or modifying components. Users do not choose the individual subprocesses. Instead, they select from the processes that you have configured for them.

A *feature* is a CMVC object used to monitor proposed design changes and record information about the design change implementation. A *defect* is a CMVC object used to monitor reported problems and record information about the problem resolution. Each feature or defect is opened for a component for evaluation, management, and resolution. The subprocesses that can be applied for a component are design, size, review (grouped as one subprocess called DSR), and verify.

### Examples

During the early stages of the development cycle or during a prototyping stage, users can select processes that do not include the DSR subprocess while still using features and defects. In this situation, users select the prototype default process. This avoids imposing administrative overhead while the program is still in a state of flux.

In later stages of development, users can select processes which include the DSR subprocess to enforce more control over development resources. These include the test and preship default processes.

You should review the default processes shipped by IBM to ensure that they provide enough choices for your users. If the default processes are not adequate or appropriate, you can create new processes or modify the existing ones.

## Configuring Release Processes

A release is a logical organization of files that are related to a particular version of a product. Each file must be managed by at least one component and grouped in at least one release. Configure processes for releases by selecting the CMVC subprocesses that users can apply when creating or modifying releases. Users do not choose the subprocesses. Instead, they select from the processes that you have configured for them.

Change control involves controlling all changes made to files. CMVC uses tracks to link changes in files to features or defects. A *track* is the CMVC object that links changes to files in a release to a particular feature or defect.

When you create a release, your most critical decision is whether to use a process that includes the *track subprocess*, for example, the `track_level` or `track_full` processes. When a process that includes the track subprocess is selected for a release, information about changes to files under CMVC control is linked to a defect or feature that is in the working state.

The *change control* requirements for each release within a family may differ. Tracks provide a way to manage the stages involved in the development of a release. The file-change process can be refined by selecting a process which includes any of the following subprocesses, all of which have the track subprocess as a prerequisite:

- Approval subprocess
- Fix subprocess
- Level subprocess
- Test subprocess.

Each of these subprocesses adds a higher level of control over changes and provides more information for managing change control. If you have configured the processes appropriately, users may select processes containing these subprocesses at any point during the development of a release. If you are defining new processes containing these subprocesses, you must also include the track subprocess. The highest level of control is exercised by requiring users to choose the `track_full` process when creating or modifying a release.

Tracks can be used to monitor file changes to a release, even if all four optional subprocesses are not used. For example, if you use the default processes shipped by IBM, users may select the `track_only` default process. This allows file changes to be tracked against a particular defect or feature, without imposing administrative overhead that may not be required for a particular project or stage of development.

### Examples

The ability to select processes allows different configurations of CMVC to be used during different stages of the development cycle. The amount of administrative control required can be varied to match the availability of resources. The following examples assume that you have installed the default processes shipped by IBM.

One possible scenario is as follows:

- At the beginning of a project, users can select a process such as prototype that does not include the track subprocess. There is no need to link features and defects to specific files because the files will be in a state of flux.
- During later stages of development, users can select a process such as track\_only, which includes the track subprocess but not the approval, test, level and fix subprocesses. This allows features and defects to be linked to the required changes in files without forcing a lot of administrative overhead.
- After the code becomes stable or the project is in beta test, users can select a process such as track\_level, which includes the track, fix, and level subprocesses. This allows a level of a release to be recreated at any point in the future.
- For the final stages of a project, users may select a process such as track\_full, which includes the track, approval, level, test and fix subprocesses. This allows more control to be exercised against development resources, which may be limited at the final stage of a project, or when you wish to enforce a freeze on code changes.

Configurable processes also allow you to configure CMVC for the size of an organization. A tighter level of control may be required to manage changes made by a large development team. The same control may not be necessary for a small development team. Figure 26 shows an example of how CMVC might be configured for different sizes of organizations.

Organization Size	Process Shipped by IBM	CMVC Release Subprocesses			
		Track	Approval	Level	Fix Test
Small	track_only	x			
Medium	track_level	x		x	x
Large	track_full	x	x	x	x x

Figure 26. Configurable Processes by Size of Organization

Figure 26 is based on the following organizational sizes.

- Small** Represents an organization where the originator and owner of features and defects is usually the same person and there is no independent test team. The level subprocess is not required. You can select a process that includes the level subprocess when you wish to capture a baseline release.
- Medium** Represents an organization where the originator and owner of features and defects are not necessarily the same person and there is no independent test team.
- Large** Represents an organization with concurrent releases doing multiplatform development, and having a test department with separate build and change teams and a committee that approves changes to releases.

---

## Configuring Processes

The family administrator must define names for the processes that will be used by the family members. CMVC users cannot select CMVC subprocesses for releases or components; they must use the processes that have been configured by the family administrator.

Processes are defined by mapping a combination of CMVC subprocesses to a name designated as a configurable process. As family administrator, you can name your own processes or use the defaults shipped by IBM. If you choose to name your own processes, the names you choose must be between 1 and 15 characters long, with no tabs, blanks or vertical separators.

Blank worksheets for recording the configurable processes for your family can be found in Appendix G, “Configurable Processes Worksheets” on page 219.

For a table showing the default processes shipped by IBM, refer to the *IBM CMVC User's Reference*.

## The `cfgcomproc.ld` and `cfgrelproc.ld` Files

Information about configurable processes for components is stored in the `cfgcomproc.ld` file. Information about configurable processes for releases is stored in the `cfgrelproc.ld` file. Both files are in the `/usr/lpp/cmvc/install` directory and are copied into *familyname* account's home directory during CMVC server configuration. If you plan to change the default processes shipped by IBM, you should edit these files before running the `mkdb` command. The `mkdb` command creates the configurable process tables, based on the settings in the `cfgcomproc.ld` and `cfgrelproc.ld` files. If you need to modify the configurable process tables after running the `mkdb` command, you must modify the `.ld` files and then run the `chcomproc` or `chrelproc` commands.

### Editing the `cfgcomproc.ld` and `cfgrelproc.ld` Files

To edit the `cfgcomproc.ld` or `cfgrelproc.ld` file:

1. Log on to the family account on the CMVC server.
2. Change directory to the family's home directory.
3. Make the proposed changes by editing the `cfgcomproc.ld` or `cfgrelproc.ld` files.

Each line in the file consists of two fields: a configurable process name and a CMVC subprocess in the following format:

```
configurable process name|CMVC subprocess name
```

The CMVC subprocess name is one of: `none`, `verifyDefect`, `verifyFeature`, `dsrDefect`, or `dsrFeature` for the `cfgcomproc.ld` file; or `none`, `track`, `approval`, `fix`, `level`, or `test` for the `chrelproc.ld` file. The configurable process name can be up to 15 characters long and cannot have tabs or blanks.

**Note:** The CMVC subprocess field cannot be left blank. However, CMVC is shipped with a configurable process named `prototype` for which all CMVC subprocesses are turned off by specifying `none` in the subprocess field.

Only CMVC subprocesses listed in the **cfgcomproc.ld** or **cfgrelproc.ld** files are used for a configurable process. CMVC subprocesses not listed are not used for that configurable process.

### The Root Process

An initial process called root is assigned to the *root component* when it is created by **mkdb**. The root process consists of the *dsrFeature*, *verifyFeature*, and *verifyDefect* subprocesses. The root process does not appear initially in the **cfgcomproc.ld** file.

If the root process is not in the **cfgcomproc.ld** file, the **Report -view cfgcomproc** command will not display this process, even though it is used by the root component. You can associate any process with the root component, as long as the process is defined in the **cfgcomproc.ld** file.

You can change the subprocesses associated with the root process by adding it to the **cfgcomproc.ld** file and modifying the included subprocesses.

### Executing the **chcomproc** and **chrelproc** Shell Scripts

After the **mkdb** command completes, the configurable process tables contain values for the configurable processes. The **chcomproc** shell script allows the family administrator to reload the contents of the configurable component process table after running the **mkdb** command. The equivalent shell script for releases is **chrelproc**.

These shell scripts replace the contents of the appropriate configurable process table with the contents of the **cfgcomproc.ld** or **cfgrelproc.ld** files. If you have to create or modify the configurable processes after a family has been created, you must first edit the **cfgcomproc.ld** or **cfgrelproc.ld** file and then run the appropriate shell script. The shell scripts are found in the **/usr/lpp/cmvc/install** directory on the CMVC server.

**Note:** We strongly recommend that you stop the CMVC daemons before issuing these shell scripts.

To execute one of the shell scripts:

1. Log in to the *familyname* account on the CMVC server.
2. Run the **chcomproc** or **chrelproc** script.

The syntax for the **chcomproc** and **chrelproc** shell scripts is shown in Figure 27.

---

**chcomproc [-h] [-s] or chrelproc [-h] [-s]**

---

Where:

<b>-h</b>	Indicates help is needed
<b>-s</b>	Runs the command in silent (no prompt) mode

---

Figure 27. Syntax for the *chcomproc* and *chrelproc* Commands



## Verification

Duplicate entries in the **cfgcomproc.ld** or **cfgrelproc.ld** file or an entry with a CMVC subprocess that is not valid will cause the shell script to fail or only partially load the configurable process database table.

If you are using a DB2/6000 or an INFORMIX database, the scripts display an error log when errors occur.

If you are using an ORACLE relational database, check the following files:

- **bad.cfgcomproc** for components and **bad.cfgrelproc** for releases  
These files contain all records that SQL\*Loader attempted to insert but could not.
- **cfgcomproc.log** for components and **cfgrelproc.log** for releases  
These files contain a detailed summary of the SQL\*Loader execution.

If you are using a SYBASE relational database, the following files are created when the **chcomproc** or **chrelproc** command uses the **bcp** command to load the configurable process table:

- **cfgcomproc.log** for components
  - **cfgrelproc.log** for releases
- These files are the SYBASE **bcp** error file.

To verify that the shell script successfully modified the configurable process tables, you can create a report using the GUI or the CMVC **Report** command from the command-line interface. You can do this only from a CMVC client workstation.

If the configurable process tables did not load correctly, make the necessary changes to the **cfgcomproc.ld** or **cfgcomproc.ld** files and run the shell script again.

## Conditions Applying to Configurable Processes

The following conditions apply to configurable processes:

- To avoid confusion for end users, do not modify configurable processes after they are created. If changes (additions or deletions of subprocesses) must be made to an existing configurable process, a new configurable process should be created.
- Do not delete a configurable process that is being used by any component or release in your family.
- Changes to configurable processes do not take effect until:
  - A component or release is modified using the configurable process name.
  - A component or release is created using the configurable process name.
- If a configurable process that includes the approval subprocess is chosen for a release, and if an *approver list* is not created for the release, an error message is displayed prompting the user to create an approver list entry consisting of at least one name or to include the approver in the command for creating the release.
- If a configurable process that includes the test subprocess is chosen for a release, and an *environment list* has not been created for that release, an error

message is displayed prompting the user to create an environment list entry consisting of at least one name or to include the environment in the command for creating the release.

## Migration from CMVC Version 1

The **-binding** flag of the **Release** command has been removed from CMVC Version 2 Release 1. Users migrating from CMVC Version 1 should use the **cfgrelproc.ld** file shipped by IBM. Previous processes have been mapped to the **no\_track**, **track\_level**, **track\_test**, **track\_approve** and **track\_full** configurable processes. These processes should not be deleted before a migration from Version 1 has been performed.

Users migrating from CMVC Version 1 should use the **cfgcomproc.ld** file shipped by IBM. Previous processes have been mapped to the default configurable process. This process should not be deleted before a migration from Version 1 has been performed.

For instructions on migrating from CMVC Version 1, refer to Appendix B, "Migrating to CMVC Version 2.3" on page 179.

---

## Chapter 9. Configuring Authority Groups

There are over one hundred actions that can be performed with CMVC commands. Grouping these actions for access control is done through the authority groups within each family.

This chapter describes the different types of authority and the groups that are used to control and configure the CMVC actions.

---

### Controlling Access Authority

Access to perform any action within CMVC is based on authority.

Within CMVC there are four types of authority:

- *Base authority*
- *Implicit authority*
- *Explicit authority*
- *Restricted authority.*

### Base Authority

All users in the family have the authority to perform the following actions as soon as a user ID is created for them: DefectOpen, DefectComment, FeatureOpen, FeatureComment, FileResolve, Report, and UserView.

### Implicit Authority

The implicit access to perform certain actions against a CMVC object is automatically granted to the user who owns the object.

For a detailed list of all CMVC actions and who can implicitly perform them, refer to the *IBM CMVC User's Reference*.

### Explicit Authority

Users who are not CMVC superusers and who do not own the specified object need to be granted the authority to perform most actions. Explicit authority to each development object is managed through the *access list* of the component that manages the specified object. The owner of each component creates entries in the access list designating users with specific explicit authority at that component.

### Restricted Authority

A superuser, a component owner, or a user with AccessRestrict authority can restrict the access of a user or an authority group for a specific component. The user can restrict this authority group access for all inherited users or for specific inherited users. Restricted authority is not inherited and does not affect the user's implicit authority or superuser authority.

## The CMVC Superuser Privilege

A user with CMVC *superuser privilege* can perform any action without being granted any explicit authority. Restricted authority does not affect superuser authority. Since a CMVC superuser is the only user who can add or delete a user, this privilege should only be granted to a user who will perform administrative tasks. Only a CMVC superuser can grant superuser privilege to another user. Minimize the number of users who have this privilege.

## Grouping Actions into Authority Groups

Authority groups are used to group CMVC actions for access control purposes. Only the family administrator should create or modify authority groups. Any number of authority groups can be configured and any number of actions can be contained within one group. Each group represents the actions a particular type of user would use. For example, in Figure 28 three authority groups are created: developer, releaselead, and manager.

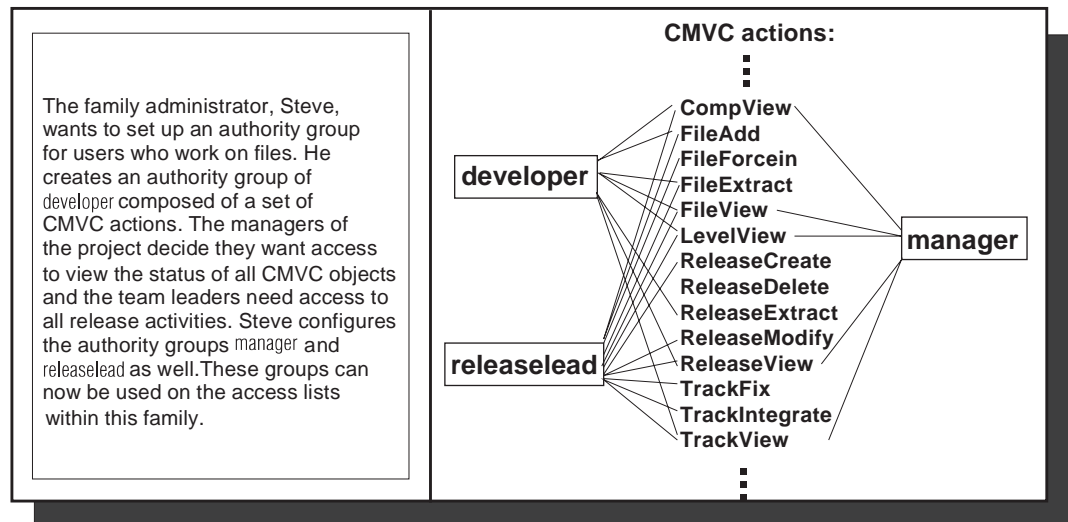


Figure 28. Grouping CMVC Actions into Authority Groups

Once these groups are configured they can be used to control the type of actions performed on objects by individual users.

Access lists pair user IDs with authority groups to grant users the authority to perform the actions contained in the specified authority group. There is one access list for each component. The authority granted on an access list pertains to all objects managed by that component and any descendant components, unless the authority has been restricted for those components.

As family administrator, you can configure your authority groups to reflect the development roles within your organization. For more information on granting access, refer to the *IBM CMVC Concepts* manual.

## Restricting Authority

A user with the proper authority can restrict access to a component for an authority group, or for specific users in that group, by using the **Access -restrict** command. This makes it easier to control the use of confidential information in CMVC, by allowing owners of components to control which users inherit access authority from the parent components. CMVC notifies users whose authority is restricted or who are subscribers of the AccessRestrict action for that component. If all inherited users are restricted at that component, then only those users who have subscribed to the AccessRestrict action are notified.

Restricted authority can be removed by using the **Access -delete** command.

---

## Configuring CMVC Actions into Authority Groups

Authority groups require careful planning and should be configured to the needs of each family. Once the authority groups for your family are configured, careful management of access is needed at each component to ensure the security of your files, and other objects managed by the components.

Authority groups are defined by mapping any configuration of CMVC actions to a name designated as an authority group name. As a family administrator, you can designate your own authority groups for your family or use the defaults shipped by IBM. If you choose to designate your own, the names you choose for the groups must be between 1 and 15 characters in length with no blanks, tabs, or vertical separators. A blank worksheet for recording your own authority group configurations can be found in Appendix E, "Authority Groups Worksheet" on page 209.

The CMVC action names to which you can map these groups and the access authority groups shipped by IBM are listed in the *IBM CMVC User's Reference*.

To create your own authority groups, edit the file **authority.Id** found in the *familyname* account's home directory prior to running the **mkdb** command during configuration of the CMVC environment.

## Editing the authority.Id File

To edit the **authority.Id** file:

1. Log on to the *familyname* account on the CMVC server.
2. Change directory to the *familyname*'s home directory.
3. Make the proposed changes by editing the **authority.Id** file.

When adding new access authority groups or when adding more low-level actions for an existing access authority group, follow the existing format of the file:

```
AuthorityGroup|CMVCActionName
```

The **authority.Id** file can only have these two fields. The **AuthorityGroup** field cannot have tabs or blanks, and it must be between 1 and 15 characters in length. The **CMVCActionName** field cannot have tabs or blanks and will only accept one of the CMVC actions specified in the table of access authority groups shipped by IBM, in the *IBM CMVC User's Reference*. Certain CMVC actions cannot be included within an authority group. The *IBM CMVC User's Reference* describes these actions and describes how to perform them.

## Using the `chauth` Script to Reload the Authority Table

Once the `mkdb` command has been performed, the Authority table will contain values for authority groups. The `chauth` shell script allows the family administrator to reload the contents of the Authority table without rerunning the `mkdb` command.

The `chauth` shell script replaces the contents of the Authority table with the values found in the `authority.ld` file which you have already copied into the `familyname` account's home directory. To modify the access authority groups after the CMVC server software is configured, you must first edit the `authority.ld` file and then run the `chauth` shell script.

The `authority.ld` file was copied to your family home directory during the CMVC server configuration. The `chauth` shell script is found in the `/usr/lpp/cmvc/install` directory on the CMVC server.

### CAUTION:

**Do not remove authority groups from the `authority.ld` file without checking if any users reference the group being considered for deletion.**

### Executing the `chauth` Shell Script

**Note:** We strongly recommend that you stop the CMVC daemons before issuing this shell script.

To execute the `chauth` shell script:

1. Log in to the `familyname` account on the CMVC server.
2. Run the `chauth` script.

The syntax for the `chauth` shell script is shown in Figure 29.

---

`chauth [-h] [-s]`

---

Where:

<code>-h</code>	Indicates help is needed
<code>-s</code>	Runs the command in silent (no prompt) mode

---

Figure 29. Syntax for the `chauth` Command

### Verification

Duplicate entries in the `authority.ld` file or an entry with a CMVC action that cannot be included in an authority group will cause the `chauth` shell script to fail or only partially load the Authority database table.

If you are using a DB2/6000 or an INFORMIX database, the shell script displays an error log when errors occur.

If you are using an ORACLE relational database, check the following files:

- **bad.authority**  
This file contains all records that SQL\*Loader attempted to insert but could not.
- **authority.log**  
This file contains a detailed summary of the SQL\*Loader execution.

If you are using a SYBASE relational database, the **authority.log** file is created when the **chauth** script uses the **bcp** command to load the Authority table. This file is the SYBASE **bcp** error file.

To verify that the shell script successfully modified the Authority table you can generate a report on the Authority table using the GUI or the CMVC **Report** command from the command-line interface. This can only be done from a CMVC client workstation.

If the Authority table did not load correctly, make the necessary changes to the **authority.ld** file and run **chauth** again.





---

## Chapter 10. Configuring Interest Groups

There are over one hundred actions that can be issued with CMVC commands. Interest groups provide configurable control of the users who are notified when certain actions are performed against certain objects.

This chapter describes the process of configuring interest groups for individual families.

---

### Controlling Notification of Actions

Users are not notified every time an action is performed within their family. With over one hundred actions possible, even a small family would generate too many notifications.

Therefore, when an action is performed users get notified in one of two ways:

- Automatic notification
- Explicit notification.

#### Automatic Notification

A user who owns a CMVC object is automatically notified of certain actions being performed against that object by other users in the family. Users also receive notification when certain actions are performed by other users that affect their CMVC user ID. For example, a user receives automatic notification when their CMVC user ID is added to an access list or when their access to an object is restricted.

#### Explicit Notification

Users who do not own a specified CMVC object need to be specified as subscribers to receive notification when actions are performed against that object. Users must subscribe to the AccessRestrict action for a component to ensure that they are notified if their access to that component is restricted.

For a detailed list of all CMVC actions and who is notified when they are performed, refer to the *IBM CMVC User's Reference*.

#### Subscribers

A subscriber is a user who is to be notified when a specific action is performed against a specific development object. Users do not subscribe to individual CMVC actions, instead they subscribe to interest groups which contain a set of CMVC actions. Subscribers to each development object are managed through the *notification list* of the component that manages that object.

---

## Grouping Actions into Interest Groups

Interest groups are used to group CMVC actions for notification purposes. Only the family administrator should create or modify interest groups. Each group should represent the actions a particular type of user would want to be notified about. The family administrator can configure and control the actions that each type of user receives notification about by using interest groups. For example, in Figure 30, three interest groups are created: low, high and builder.

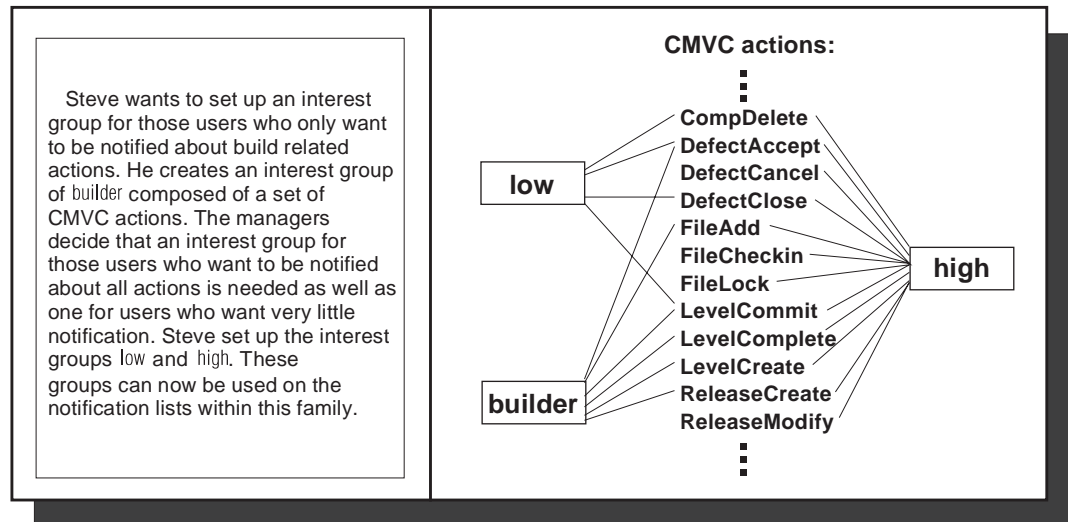


Figure 30. Grouping CMVC Actions into Interest Groups

The component owners can then use these notification interest groups to specify individual users as subscribers on notification lists. For more information on using notification interest groups, see *IBM CMVC Concepts*.

---

## Configuring Interest Groups

Interest groups take careful planning and should be configured to the needs of each user.

Interest groups are defined by mapping any configuration of CMVC actions to a name designated as an interest group name. As family administrator, you can designate your own interest groups or use the defaults shipped by IBM. A blank worksheet for recording the interest configuration for your family can be found in Appendix F, "Interest Groups Worksheet" on page 215.

The CMVC action names to which you can map these groups and the notification interest groups shipped by IBM are listed in the *IBM CMVC User's Reference*.

To create your own interest groups, edit the **interest.ld** file found in the *familyname* account's home directory prior to running the **mkdb** command during configuration of the CMVC environment.

## Editing the `interest.ld` File

To edit the `interest.ld` file:

1. Log in to the family account on the CMVC server.
2. Change directory to the family's home directory.
3. Make the proposed changes by editing the `interest.ld` file.

When adding new interest groups or when adding more low level actions for an existing notification interest group, follow the existing format of the file:

```
InterestGroup|CMVCActionName
```

The `interest.ld` file can only have these two fields. The `InterestGroup` field cannot have tabs or blanks and it must be between 1 and 15 characters in length. The `CMVCActionName` field cannot have tabs or blanks and only accepts one of the CMVC actions specified in the table of notification interest groups shipped by IBM. Certain CMVC actions cannot be included within an interest group. These actions and information regarding the type of notification they relate to are also described in the *IBM CMVC User's Reference*.

## Using the `chintr` Script to Reload the Interest Table

After the `mkdb` command completes, the interest table contains values for interest groups. The `chintr` shell script allows the family administrator to reload the contents of the Interest table without rerunning the `mkdb` command.

The `chintr` shell script replaces the contents of the Interest table with the values found in the `interest.ld` file in your *familyname* home directory. If there is a need to modify the interest groups once the CMVC server is configured, you must first edit the `interest.ld` file and then run the `chintr` shell script.

The `interest.ld` file was copied to the *familyname* account's home directory during the CMVC server configuration. The `chintr` shell script is found in the `/usr/lpp/cmvc/install` directory on the CMVC server.

### CAUTION:

**Do not remove interest groups from the `interest.ld` file without checking if any users reference the group being considered for deletion.**

### Executing the `chintr` Shell Script

**Note:** We strongly recommend that you stop the CMVC daemons before issuing this shell script.

To run the `chintr` shell script:

1. Log in to the *familyname* account on the CMVC server.
2. Run the `chintr` shell script.

The syntax for the `chintr` shell script is shown in Figure 31 on page 76.

---

**chintr [-h] [-s]**

---

Where:

<b>-h</b>	Indicates help is needed
<b>-s</b>	Runs the command in silent (no prompt) mode

---

*Figure 31. Syntax for the chintr Command*

## Verification

Duplicate entries in the **interest.ld** file or an entry with a CMVC action that cannot be included in an interest group will cause the **chintr** shell script to fail or only partially load the Interest database table.

If you are using a DB2/6000 or an INFORMIX database, the script displays an error log when errors occur.

If you are using an ORACLE relational database, check the following files:

- **bad.interest**

This file contains all records that SQL\*Loader attempted to insert but could not.

- **interest.log**

This file contains a detailed summary of the SQL\*Loader execution.

If you are using a SYBASE relational database, the **interest.log** file is created when the **chintr** command uses the **bcp** command to load the Interest table. This file is the SYBASE **bcp** error file.

To verify that the shell script successfully modified the Interest table you can generate a report on the Interest table by using the GUI or the CMVC **Report** command from the command-line interface. This can only be done from a CMVC client workstation.

If the Interest table did not load correctly, make the necessary changes to the **interest.ld** file and run **chintr** again.

---

## Chapter 11. Modifying Your Configuration Table

The configuration database table defines values for CMVC object attributes. These values should be configured to represent the information important to your development organization.

This chapter describes how to modify the configuration database table and the information that needs to be included in this table.

---

### The Configuration Database Table

The database table for configuration is called the Config table. The Config table is divided into six columns:

**type** The **type** field represents configuration type values for defect, feature, and level attributes. You can create new types and you can configure how many name entries each type has in the database table. There must be at least one entry per type. Type entries cannot have blank spaces or tabs. Types shipped by IBM are: priority, leveltype, severity, answerAccept, answerReturn, defectPrefix, featurePrefix, phase, and symptom.

**name** The **name** field represents the choices the user has within the configuration type. The defaults shipped by IBM may not be adequate for your CMVC family. You can create new types and add as many names as you want for each configuration type. Name entries cannot have spaces or tabs.

**Note:** The name of a type cannot be the prefix for another name of the same type. For example:

```
severity|123|yes|0|0|desc
```

123 is a valid type name. If it exists in your Config table, then the following would be considered INVALID type names:

```
severity|1|no|0|0|desc  
severity|12|no|0|0|desc
```

1 and 12 are prefixes of 123 and are thus invalid.

**dflt** The **dflt** field indicates the defined names that are used as the default when the user does not enter a value for the configuration type when performing defect, feature, or level related activities. The **dflt** must be either a yes or no and only one name for each configuration type may have its **dflt** field set to yes. The Config table shipped by IBM does not have any of the values set as defaults.

**value1** and **value2**

Must be set to 0. Reserved for future use.

**description** The **description** field provides a description of each name value. It must be less than or equal to 63 characters and can be set to null.

## Configuration Types

The entries in the configuration table are provided to help your organization structure your *problem tracking* information. How you choose to use each type is configurable for each family. The following descriptions are recommendations:

<b>priority</b>	An indication of the timing or scheduling requirements for resolving a defect or feature.
<b>leveltype</b>	The type of level the defect or feature resolution should be included in.
<b>severity</b>	An indication of the severity of a defect.
<b>answerAccept</b>	The answer to the defect or feature, used by the defect or feature owner when accepting to work on a defect or consider a feature.
<b>answerReturn</b>	The answer to the defect or feature, used by the defect or feature owner when returning a defect or feature to the originator.
<b>defectPrefix</b>	A prefix indicating the type of a defect. The prefix attribute of a defect or feature can be used to identify a problem as either a defect or a feature when looking at information regarding both. Use unique prefixes for defects.
<b>featurePrefix</b>	A prefix indicating the type of feature. The prefix attribute of a defect or feature can be used to identify a problem as either a defect or a feature when looking at information regarding both. Use unique prefixes for features.
<b>phase</b>	The development phase in progress when the defect was discovered or the development phase in progress when the defect was injected into the code.
<b>symptom</b>	An indication of the symptom of the problem.

**Note:** If the default configurable fields shipped by IBM are not installed, then the types *priority*, *phase*, and *symptom* are not used.

---

## Modifying the Config Table

To modify the Config table prior to running the **mkdb** command, edit the file **config.ld** found in the *familyname* account's home directory.

To modify the configuration table after the family has been configured see "Using the chcfg Script to Reload the Config Table" on page 84.

Figure 32 on page 79 shows the values shipped by IBM for the configuration table.

<b>type</b> characters:15	<b>name</b> characters:15	<b>dflt</b> characters:3	<b>value1</b>	<b>value2</b>	<b>description</b> characters:63
answerAccept	docs_defect	no	0	0	Documentation needs to be changed
answerAccept	program_defect	no	0	0	The problem was due to a program error
answerAccept	plans_change	no	0	0	Plans or schedules need to be changed
answerAccept	new_function	no	0	0	New function will be added
answerAccept	redesign	no	0	0	Current function needs to be redesigned
answerAccept	docs_change	no	0	0	Documentation needs to address new features
answerAccept	remove_support	no	0	0	Non-supported functions need to be removed
answerAccept	fix_testcase	no	0	0	Testcase needs to be fixed
answerAccept	remove_code	no	0	0	Obsolete code needs to be removed
answerAccept	comply_with	no	0	0	Coding practices and operation needs to comply with standards
answerReturn	fixed	no	0	0	The problem is already fixed
answerReturn	future	no	0	0	Future releases or versions will address the defect or feature
answerReturn	duplicate	no	0	0	This is a duplicate of an existing defect or feature
answerReturn	unrecreatable	no	0	0	The problem cannot be re-created
answerReturn	as_designed	no	0	0	The program works as designed

Figure 32 (Part 1 of 5). Defaults Shipped by IBM for the Config Table

<b>type</b> characters:15	<b>name</b> characters:15	<b>dflt</b> characters:3	<b>value1</b>	<b>value2</b>	<b>description</b> characters:63
answerReturn	hardware_error	no	0	0	The problem is caused by a hardware error
answerReturn	deviation	no	0	0	Code or documentation will deviate from the standards
answerReturn	info_needed	no	0	0	More information is required
answerReturn	limitation	no	0	0	This problem is a current limitation
answerReturn	suggestion	no	0	0	This problem is a suggestion, not an error
answerReturn	usage_error	no	0	0	The problem is caused by incorrect usage
defectPrefix	c	no	0	0	Defect reported by a customer
defectPrefix	d	no	0	0	Defect reported by internal users
featurePrefix	s	no	0	0	Suggestion made by customer
featurePrefix	f	no	0	0	Feature requested by internal users
leveltype	development	no	0	0	Development level
leveltype	production	no	0	0	Production level
leveltype	integration	no	0	0	Integration level
leveltype	prototype	no	0	0	Prototype level
leveltype	other	no	0	0	Other type of level
phase	development	no	0	0	Development Phase
phase	design	no	0	0	Design Phase
phase	planning	no	0	0	Planning Phase
phase	strategy	no	0	0	Strategic Planning Phase

Figure 32 (Part 2 of 5). Defaults Shipped by IBM for the Config Table



<b>type</b> characters:15	<b>name</b> characters:15	<b>dflt</b> characters:3	<b>value1</b>	<b>value2</b>	<b>description</b> characters:63
phase	prototyping	no	0	0	Prototyping Phase
phase	inspections	no	0	0	Inspection Phase
phase	maintenance	no	0	0	Maintenance Phase
phase	building	no	0	0	Building, Compiling or Module Integration Phase
phase	documenting	no	0	0	Documentation or Publication Phase
phase	functional_test	no	0	0	Functional Test
phase	regression_test	no	0	0	Regression Test
phase	install_test	no	0	0	Install Test
phase	config_test	no	0	0	Configuration Test
phase	integrate_test	no	0	0	Integrate Test
phase	quality_test	no	0	0	Quality Assurance Test
phase	usability_test	no	0	0	Usability Test
phase	ship_test	no	0	0	Ship Test
phase	beta_test	no	0	0	Beta test
phase	n/a	no	0	0	Not applicable to any particular phase
phase	unit_test	no	0	0	Unit Test
priority	mustfix	no	0	0	Defect or feature must be resolved in this release
priority	candidate	no	0	0	Defect or feature is a candidate if time permits
priority	deferred	no	0	0	Defect or feature deferred to next release
priority	easy	no	0	0	Defect or feature is easy to solve or implement

Figure 32 (Part 3 of 5). Defaults Shipped by IBM for the Config Table

<b>type</b> characters:15	<b>name</b> characters:15	<b>dflt</b> characters:3	<b>value1</b>	<b>value2</b>	<b>description</b> characters:63
priority	moderate	no	0	0	Defect or feature is moderately difficult to resolve
priority	difficult	no	0	0	Defect or feature is difficult to solve or implement
priority	n/a	no	0	0	Priority does not apply to this defect or feature
severity	1	no	0	0	Wrong results or failure; critical to program execution
severity	2	no	0	0	Wrong results; not critical to program execution
severity	3	no	0	0	Unexpected behavior
severity	4	no	0	0	Suggestion or enhancement request
symptom	incorrect_i/o	no	0	0	Incorrect or unexpected input or output
symptom	program_defect	no	0	0	Program defect
symptom	design_wrong	no	0	0	Original design is incorrect; redesign required
symptom	function_needed	no	0	0	Additional function is required
symptom	plans_incorrect	no	0	0	Plans need to be changed or enhanced
symptom	docs_incorrect	no	0	0	Documentation is incorrect
symptom	prog_suspended	no	0	0	Program suspended during normal operation

Figure 32 (Part 4 of 5). Defaults Shipped by IBM for the Config Table

<b>type</b> characters:15	<b>name</b> characters:15	<b>dflt</b> characters:3	<b>value1</b>	<b>value2</b>	<b>description</b> characters:63
symptom	core_dump	no	0	0	Core dump occurred during normal operation
symptom	lost_data	no	0	0	Data loss occurred during normal operation
symptom	usability	no	0	0	Program or application is not usable as is
symptom	test_failed	no	0	0	Test failed
symptom	build_failed	no	0	0	Build, compile or module integration failed
symptom	install_failed	no	0	0	Installation failed
symptom	obsolete_code	no	0	0	Remove obsolete code
symptom	intgr_problem	no	0	0	Integration problems with other applications
symptom	performance	no	0	0	Performance problems; code needs to be optimized
symptom	reliability	no	0	0	Reliability problems; code needs more work
symptom	non-standard	no	0	0	Coding practices or program execution is non-standard
symptom	not_to_spec	no	0	0	Program or application does not function as specified

Figure 32 (Part 5 of 5). Defaults Shipped by IBM for the Config Table

## Editing the config.ld File

To edit the **config.ld** file:

1. Log in to the *familyname* account on the CMVC server.
2. Change directory to the family's home directory.
3. Make the proposed changes.

When adding new values to this file, follow the existing format of the file. When using the:

- ORACLE relational database, the format is as follows:  
type|name|default|0|0|"description"
- INFORMIX relational database, the format is as follows:  
type|name|default|0|0|description|
- SYBASE relational database, the format is as follows:  
type|name|default|0|0|description
- DB2/6000 relational database, the format is as follows:  
type|name|default|0|0|"description"

## Using the `chcfg` Script to Reload the Config Table

The `chcfg` script replaces the contents of the Config table with the values found in the `config.ld` file. If there is a need to modify the Config table values after your family has been configured, you must first edit the `config.ld` file and then run the `chcfg` shell script. You must run `chcfg` if you are using configurable fields and create a new type.

The `config.ld` file is copied to the *familyname* account's home directory during the CMVC configuration. The `chcfg` shell script is found in the `/usr/lpp/cmvc/install` directory on the CMVC server.

Changing values in the Config table does not change values already in the CMVC database for existing records.

### Running the `chcfg` Shell Script

**Note:** We strongly recommend that you stop the CMVC daemons before issuing this shell script.

To run the `chcfg` shell script:

1. Log in to the *familyname* account on the CMVC server.
2. Run the `chcfg` script.

The syntax for the `chcfg` shell script is shown in Figure 33.

---

```
chcfg [-h] [-s]
```

---

Where:

<b>-h</b>	Indicates help is needed
<b>-s</b>	Runs the command in silent (no prompt) mode

---

Figure 33. Syntax for the `chcfg` Command

### Verification

Duplicate entries in the `config.ld` file or entries with field widths wider than the allowable values cause the `chcfg` shell script to fail or only partially load the Config table.

If you are using a DB2/6000 or an INFORMIX relational database, the script displays an error log when errors occur.

If you are using an ORACLE relational database, check the following files:

- **bad.config**  
This file contains all records that SQL\*Loader attempted to insert but could not.
- **config.log**  
This file contains a detailed summary of the SQL\*Loader execution.

If you are using a SYBASE relational database, check the `config.log` file. This is the SYBASE `bcp` error file.

To verify that the shell script successfully modified the Config table you can generate a report on the Config table using the GUI or the CMVC **Report** command from the command-line interface. This can only be done from a CMVC client workstation.

If the Config table did not load correctly, make the necessary changes to the **config.ld** file and run **chcfg** again.



---

## Chapter 12. Providing User Exits

This chapter describes user exits, how they can be used and how they can be implemented for each CMVC family. User exits are not necessary for the operation of CMVC; they are optional and can be configured for each CMVC family.

With user exits you can specify additional actions that you want performed before completing or proceeding with a specific CMVC action. A user exit enables the CMVC server to call a user-defined program during the processing of CMVC actions. Therefore, you can use CMVC as a trigger to start non-CMVC processing. You can also use user exits to restrict certain CMVC actions based on external considerations. For example, you might have a user exit scan C source files to ensure that the source code conforms to the standards defined by your development process.

CMVC provides user exits for most CMVC actions. See Appendix H, “User Exit Parameters” on page 221 for a list of the CMVC actions that support user exits.

---

### Configuring User Exits

Do the following to configure a user exit:

1. Stop the CMVC daemons.
2. Edit the **userExits** file that is in the **\$HOME/config** directory of the CMVC family’s account and add an entry for each user exit. See “Editing the userExits File” for more details.

The **userExits** file has no defined actions until you add entries for the user exits that your organization will use. The entries you add specify the programs or shell scripts that you want started for specific CMVC actions.

If you do not have a **\$HOME/bin** directory, use the **mkfamily** command to create it for you. Include the **\$HOME/bin** directory in the **\$PATH** environment variable for the CMVC family account so that the CMVC server can find the user exit programs.

For more information about editing the **userExits** file, refer to “Using the mkfamily Command” on page 41.

3. Create a program or shell script file in the **\$HOME/bin** directory for each entry in the **userExits** file.
4. Start the CMVC daemons. Your user exits are now available.

### Editing the userExits File

Before you can implement user exits, you must edit the **userExits** file. For each user exit, add an entry using the following format:

```
CMVCaction ExitID UProgram UParameters #Comments
```

Separate each field in the entry by one or more blank spaces. A line that begins with a # sign is treated as a comment. You can have blank lines in the file.

A description of each field in the entry follows:

- CMVCaction** The name of the CMVC action that causes the user exit to start.
- ExitID** An exit ID that identifies when the user exit program is started during the course of the CMVC action. Valid exit IDs are 0, 1, 2, and 3, where:
- 0** Indicates that the user exit program starts at the beginning of the CMVC action, before any initialization or access checking takes place.
  - 1** Indicates that the user exit program starts after all CMVC checks are made and CMVC is ready to process the command.
  - 2** Indicates that the user exit program starts after the CMVC action is completed. At this point, the action has been submitted to CMVC and all database or library updates have been committed.
  - 3** Indicates that the user exit program starts if a user exit with an exit ID of 0 or 1 is not successful, or the CMVC action is not successful. This exit ID allows the user exit program to clean up what the other user exit programs started.
- UEprogram** The name of the user exit program. It must exist in the **\$HOME/bin** directory of the CMVC family.
- UEparameters** A variable length list of character string parameters provided to the user exit program.
- #Comments** A comment about the user exit program. This field is optional.

---

## Writing User Exit Programs

Follow these guidelines when you write user exit programs:

- Limit the length of time that the user exit program runs.
- Avoid using commands that update the database. Instead, use the **Report** command or commands that have the **-view** action.  
**Note:** Using commands that update the database can cause deadlock.
- Run at least two **cmvcd** daemons when using commands from within a user exit.
- User exit programs do not permit user interaction (for example, from a user exit shell script, you cannot prompt a user with a read command).
- User exit programs must reside in the **\$HOME/bin** directory.
- Use the following command to set the executable bit for each user exit:  

```
chmod u+x filename
```
- Define only one user exit program in the **userExits** file for each CMVC CMVCaction and exitID combination. If you create more than one entry, CMVC uses the last entry.

User exit programs have the following behavior:

- Each user exit shell script receives a unique list of parameters, defined as follows:



**UEprogram** Name of the calling user exit (parameter \$0).  
**UEparameters** User-defined parameters in the **userExits** file, if any.  
**Action parameters** The parameters passed to the user exit program from the CMVC command.

- When a user does not enter a value in an optional GUI field or command line, no value is passed to the user exit for that positional parameter. This is also true for UEparameters.

For example, a user exit exists for FileCheckIn for exit ID 1. If a user does not provide information for the **Remarks** parameter, then the **Common release names** parameter value follows the **Force flag** parameter value. The **Remarks** field is null.

- Positional parameters that pass true or false values, such as **Break common link** (force flag), return the following:

**True** 1  
**False** 0

- A nonzero return code from a user exit shell script for exit ID 0 or 1 terminates the CMVC command. Nonzero return codes have no effect on exit ID 2 or 3.
- The **userExits** file is only read when the CMVC daemons are started. After a user exit is enabled, you can change it without restarting the CMVC daemons.

The table in Appendix H, “User Exit Parameters” on page 221 lists the parameters that are passed by each CMVC action and exit ID. The table also contains descriptions of the parameters.

Figure 34 on page 90 is an example of the parameters passed to the user exit program defined for the **FileAdd** action.

Exit ID	CMVC Action Parameters
0	<ul style="list-style-type: none"> <li>• file path name</li> <li>• temp file on server</li> <li>• release name</li> <li>• component name</li> <li>• file type (0 if not binary, a if binary)</li> <li>• remarks</li> <li>• fileMode</li> <li>• effective CMVC user ID</li> <li>• real UNIX** login</li> <li>• verbose flag</li> </ul>
1	<ul style="list-style-type: none"> <li>• file path name</li> <li>• temp file on server</li> <li>• release name</li> <li>• component name</li> <li>• file type (0 if not binary, a if binary)</li> <li>• remarks</li> <li>• fileMode</li> <li>• effective CMVC user ID</li> <li>• verbose flag</li> </ul>
2	<ul style="list-style-type: none"> <li>• file path name</li> <li>• temp file on server</li> <li>• release name</li> <li>• component name</li> <li>• file type (0 if not binary, a if binary)</li> <li>• remarks</li> <li>• fileMode</li> <li>• effective CMVC user ID</li> <li>• verbose flag</li> </ul>
3	The parameters passed to exit ID 3 are the same as those passed to exit ID 0 without the <i>real Unix login</i> parameter, but with one additional parameter as the last parameter. This parameter indicates that the last user exit ID that ran successfully, (for example, exit ID 0 or 1).

Figure 34. User Exit Parameters for the FileAdd Action

You might want to have a user exit that runs when a user creates a file in CMVC. You would include the following line in the **userExits** file to invoke a user exit program called **showActionParms** each time a user creates a CMVC file.

```
FileAdd 1 showActionParms "1991, 1992" # checks for 1991 & 1992
```

This program runs when CMVC recognizes that the user is requesting a valid **FileAdd** action, but before the **FileAdd** action actually starts. If the content of the **showActionParms** user exit shell script in **\$HOME/bin** is:

```
#!/bin/ksh
#Show name of user exit and action parameters
echo $0 $*
exit 0
```

and the command issued by the CMVC client is:

```
File -create fileX.c -component codeA -release ToolAv1 -family
TestFam -defect 100
```

the output displayed in the client window is:

```
/home/TestFam/bin/showActionParms "1991, 1992" fileX.c  
/tmp/cmvcAAGuacwxbb  
ToolAv1 codeA 0 Initial Version 0444 Matt 0
```

In the above example:

- 1991, 1992 are the UEParameters in the **userExits** file.
- The output beginning with fileX.c and going to the end of the statement is defined in Figure 34 on page 90 in the section for exit ID 1.
- The temporary file created on the server is /tmp/cmvcAAGuacwxbb.
- Because no remark was specified, CMVC provided Initial Version.
- The user issuing the command is Matt.



---

## Part 3. Working with CMVC

This part of the book outlines the activities that you can perform after your initial setup. These activities include starting your CMVC server, migrating the files from SCCS, managing the audit log, using the mail facility, monitoring the daemon activity, aging defects, backing up and recovering data, and archiving and restoring releases and levels.



---

## Chapter 13. CMVC Server Daemons

This chapter describes how to start and stop the CMVC server daemons for each CMVC family. The CMVC server software must be installed and configured before you can start a CMVC server daemon.

---

### Starting the CMVC Server `cmvcd` Daemons

The `cmvcd` daemons run on the CMVC server and handle CMVC client requests for CMVC actions. Each daemon handles one request at a time. Determine the number of daemons you want to start based on the load you expect the CMVC server to handle. For information on monitoring CMVC server performance and CMVC daemon activity, refer to Chapter 14, "Ongoing Maintenance" on page 101.

The `cmvcd` daemons can be run in maintenance mode to provide users with read-only access to CMVC during backups or archiving.

Each `cmvcd` daemon requires a database license. CMVC will support a maximum of 255 `cmvcd` daemons.

**Note:** Remember that you must reserve one database license for the `notifyd` daemon.

### Prerequisite Tasks

Before starting the `cmvcd` daemons you must do the following:

1. Ensure that the relational database is installed and running.
2. Ensure that the CMVC server code is installed and configured.
3. Ensure that the NFS client daemons are running.
4. Log in to the *familyname* user account.

### Starting `cmvcd`

Issue the following command:

```
/usr/lpp/cmvc/bin/cmvcd <familyname> n
```

where *familyname* is the name of your family and *n* is the number of `cmvcd` daemons you are starting for this family. If you do not specify the number of `cmvcd` daemons, the default is 1. The option of 0 daemons is available for use during problem determination.

**Note:** Twice the number of daemons specified are activated. However, only the number of daemons you specify actively process requests and only the number of daemons you specify are database users.

To start more daemons than the number with which you started `cmvcd`, stop all the `cmvcd` daemons and restart `cmvcd` specifying the new number of daemons.

To verify that the **cmvcd** daemons are running properly, issue one of the following commands, depending on which operating system you are using:

- For the AIX, Solaris, and HP-UX operating systems,

```
ps -u <familyName> | grep cmvcd
```

- For the SunOS operating system,

```
ps -ax | grep cmvcd
```

The **cmvcd** daemons and the CMVC archive and restore programs are mutually exclusive. You cannot run the CMVC archive and restore programs while any **cmvcd** daemons are running and you cannot start the **cmvcd** daemons while the archive and restore programs are running.

## Starting cmvcd in Maintenance Mode

Start the **cmvcd** daemons in maintenance mode to provide users with read-only access to CMVC during backups or archiving. The regular **cmvcd** daemons must be stopped to run the **cmvcd** daemons in maintenance mode. See “Stopping the CMVC Server Daemons” on page 98 for information on stopping the **cmvcd** daemons.

Issue the following command:

```
/usr/lpp/cmvc/bin/cmvcd -m <familyname> n
```

where *familyname* is the name of your family and *n* is the number of maintenance mode **cmvcd** daemons you are starting for this family.

**Note:** Twice the number of daemons specified are activated. However, only the number you specify actively process requests and only the number you specify are database users.

You can start the **cmvcd** daemons in maintenance mode while the archive and restore programs are running.

You can issue the following CMVC commands while **cmvcd** is running in maintenance mode:

```
Comp -view
Defect -config
Defect -view
Feature -config
Feature -view
File -config
File -extract
File -view
Level -view
Release -view
Report
Track -view
User -config
User -view
```



---

## Starting the CMVC Server notifyd Daemon

The **notifyd** daemon runs on the CMVC server and handles the routing of notification messages to CMVC users based on the mailing address in their CMVC user ID. There is one **notifyd** daemon for each CMVC family. The **notifyd** daemon also requires a database license. You must satisfy the following conditions before starting **notifyd**.

1. The relational database must be installed and running.
2. The CMVC server must be installed and configured.
3. The **sendmail** daemon must be running on the CMVC server.
4. Log in to the *familyname* user.
5. If you are using SYBASE SQL Server System 10, you must create a guest account within the database so that notifyd can create sysprocs in the database. For more information, refer to the SYBASE documentation.

To start **notifyd**, issue the following command:

```
/usr/lpp/cmvc/bin/notifyd
```

---

## Starting the CMVC Server Daemons on System Reboot

**Note:** You must ensure that you start the database before you invoke the CMVC daemons.

If you are using the AIX or HP-UX operating systems and you want to configure your system to start CMVC on system reboot, add the following commands to the bottom of the **/etc/inittab** file:

```
cmvc001:2:wait:/bin/su - <familyname> -c "/usr/lpp/cmvc/bin/cmvcd <familyname> n"  
cmvc002:2:wait:/bin/su - <familyname> -c "/usr/lpp/cmvc/bin/notifyd".
```

where *n* is the number of daemons. Enter each command on one line of the file.

If you are using the SunOS operating system and you want to configure your system to start CMVC on system reboot, add the following lines to the bottom of the **/etc/rc.local** file:

```
/usr/lpp/cmvc/bin/cmvcd <familyname> n &  
/usr/lpp/cmvc/bin/notifyd &
```

where *n* is the number of daemons.

If you are using the Solaris operating system, the sample initialization scripts used to implement init state changes for cmvcd have been supplied in the **/usr/lpp/cmvc/samples** directory. These files are named **S80cmvcd.rc2** and **cmvcd.init**. Modify these sample files to suit your individual site needs, and place them in the corresponding <0-3>.d directories.

---

## Stopping the CMVC Server Daemons

You must do the following before stopping any of the CMVC server daemons:

1. Log in to the *familyname* user account.
2. Determine the *process IDs* for the daemon you want to stop. Issue one of the following commands, depending on which operating system you are using:
  - For the AIX, Solaris, and HP-UX operating systems,  

```
ps -u <familyName> | grep cmvcd
```
  - For the SunOS operating system,  

```
ps -ax | grep cmvcd
```
3. For each **cmvcd** or **notifyd** daemon that you want to stop, issue the following command:

```
kill <PID>
```

where *PID* is the process ID of the daemon you are stopping.

Alternatively, you can run the **stopCMVC** sample script which is in the **/usr/lpp/cmvc/samples** directory on the CMVC server, by issuing the following command:

```
/usr/lpp/cmvc/samples/stopCMVC <familyname>
```

This command stops any **cmvcd** or **notifyd** daemons that are running for the CMVC family specified by *familyname*.

## Stopping the CMVC Server Daemons on System Shutdown

**Note:** If you are using ORACLE, you must manually shutdown the CMVC server daemons.

If you are using the AIX operating system and you want to stop **cmvcd** and **notifyd** when you shut down your system, add the following line to the top of the **/etc/shutdown** file:

```
/bin/su - <familyname> -c "/usr/lpp/cmvc/samples/stopCMVC <familyname>"
```

You must also stop the NetLS daemons before shutting down your system. To shut down the NetLS daemons, you can use the **kill** command, as described earlier in the previous section.

If you are using the AIX operating system and the NetLS software is installed on the CMVC server, you can add the following lines to the top of the **/etc/shutdown** file:

```
stopsrc -s netltd  
stopsrc -s glbd  
stopsrc -s llbd
```

If you are using the SunOS or HP-UX operating system, you can create a personal shutdown script which contains preshutdown procedures and the **shutdown** command. You can also use the **halt** and **reboot** commands to shut down the system. You should use the **kill** command to stop the NetLS daemons before system shutdown.

If you are using the Solaris operating system, the sample termination scripts used to implement init state changes for `cmvcd` have been supplied in the `/usr/lpp/cmvc/samples` directory. These files are named `K55cmvcd.rc0`, `K55cmvcd.rc1`, and `cmvcd.init`. Modify these sample files to suit your individual site needs, and place them in the corresponding `<0-3>.d` directories.

## Recycling the CMVC Server Daemons

The administrator may have to recycle a `cmvcd` daemon if a client request is too long. For instructions on how to recycle your `cmvcd` daemons, refer to “Determining Which Users Issue Time Consuming Reports” on page 108.



---

## Chapter 14. Ongoing Maintenance

This chapter describes the maintenance activities involved in administering a CMVC family.

---

### Audit Log

An audit log is maintained for each CMVC family. This audit log contains an entry for each CMVC transaction and must be carefully maintained. For a detailed description of this facility and how to manage it, see Chapter 15, "The CMVC Audit Log."

---

### Mail

All CMVC server **notifyd** daemons use the **sendmail** command to deliver notification messages from the CMVC server to CMVC client workstations. Notification messages are sent to the mail address specified in each user's CMVC user ID.

The **sendmail** command is part of the operating system and is installed with it. **Sendmail** must be properly configured on your network for CMVC client workstations to receive mail notification messages from the CMVC server. Refer to your operating system documentation for more information. The **sendmail** command can deliver messages to:

- Users in the local system.
- Users connected to the local system using the TCP/IP protocol.
- Users connected to the local system using the Basic Networking Utilities (BNU) command protocol.

To deliver mail to users on the local system, additional addressing and routing information is not required. The **sendmail** command routes local mail into a user's system mailbox. The location of the mailbox is as follows:

<b>Operating System</b>	<b>Directory</b>
AIX and SunOS	/usr/spool/mail
HP-UX	/usr/mail
Solaris	/var/mail

To deliver mail in a local area network, the network must be using the TCP/IP network protocol. To simplify the task of keeping the address information up-to-date, you may want to set up a name server for the network. A name server controls the addressing information for the network and connections to systems beyond the immediate network. To deliver mail to another system connected to the local system with a UUCP link, the link must be defined to the Basic Networking Utilities in the BNU configuration files. The Basic Networking Utilities system is the version of UUCP used by the operating system. The **sendmail** command transfers mail to the BNU commands for delivery across a UUCP link.

## Mail Addressing

A user's mail address is specified when a CMVC user ID is created. This mail address is subsequently used by the CMVC server **notifyd** daemon to send CMVC notifications to the user whenever actions for which the user has subscribed, or when actions against objects the user owns, have occurred. The subscription for notification can be either explicit (interest in actions can be set using the notification list of a component) or implicit (notification of certain actions is sent automatically to the user who is an owner or originator of the object).

## Addressing Using a Central Database of Names and Addresses

If the network to which your CMVC server is connected uses a central database of names, use the following format for the user's mail address:

`Login@SystemName`

## Addressing Using Domain Name Addressing

For networks that span large, unrelated local networks in widespread locations, a central database of names is not possible. In this instance, use the following format for the user's mail address:

`Login@SystemName.DomainName`

## Mail Queue—Processing Interval

The frequency with which CMVC client workstation users receive mail notification from the CMVC server **notifyd** daemon depends largely on the configuration of the **sendmail** daemon on the CMVC server. The interval at which the **sendmail** daemon processes the mail queue is determined by the value of the **-q** flag when the **sendmail** daemon starts. The default value for the queue-processing interval is usually 30 minutes and is contained in a shell script:

- **/etc/rc.tcpip** for the AIX operating system
- **/etc/rc.local** for the SunOS operating system
- **/etc/netbsdsrc** for the HP-UX operating system
- **/etc/rc2.d/588sendmail** for the Solaris operating system

This means that when running with the default **sendmail** configuration, client users will only receive mail notifications from the CMVC server once every 30 minutes. If this value does not suit your environment, or if you are unsure what the mail queue-processing interval is on your system, refer to your operating system documentation for more information.

## Restored Mail

Mail which cannot be delivered to a CMVC user will be returned to the mail box of the CMVC family login. As a family administrator, you should periodically examine the returned mail and take the appropriate action to correct the problems which have resulted in the mail being returned.

---

## Aging Defects and Features

There are two aging tools provided with CMVC. Both of these tools serve to update the age value of defects and features that are in states that indicate that work is still in progress for the resolution of defects or the implementation of features. If neither of the age tools are used the age value for each defect and feature will remain at zero.

### The Age Shell Script

The first of these tools is a customizable shell script that uses the database SQL interface to increment the age of defects and features. The supplied version of this script serves to increment the age value of defects and features that are in the open, working, design, size or review states. The states that you want to consider for the aging of defects and features are customizable.

The **age** shell script can be found in the `/user/lpp/cmvc/bin` directory on the CMVC server. You must be logged in as the *familyname* account to invoke the script. If you are using an ORACLE or SYBASE database, run the **age** shell script by entering the following:

```
age <familyname> dbpasswd
```

Where

Variable	Description
<i>familyname</i>	The name of the family for which you intend to age the defects and features
<i>dbpasswd</i>	The password that permits the family to login to the database.

If you are using a DB2/6000 or an INFORMIX database, run the **age** shell script by entering the following from your command line:

```
age <familyname>
```

Where

Variable	Description
<i>familyname</i>	The name of the family for which you intend to age the defects and features.

This script performs a simple increment function on every defect or feature that is in the appropriate state each time it is invoked. Use the **cron** utility to establish a daily invocation of the **age** script at a time when system activity is low.

If the value of the defect or feature ages loses its synchronization and becomes incorrect, you can use the **resetAge** program described in the next section to reset the age value to a correct value and then resume using the **age** shell script.

The **age** script requires a database license while it is running.

## The resetAge Program

The **resetAge** program calculates and resets the age of defects and features based on their state, the date they were opened, and the selected aging methodology.

The program is found in the **/usr/lpp/cmvc/bin** directory on the CMVC server and must be invoked while logged in as the *familyname* account. To invoke the program enter the following on the command line:

```
resetAge fullweek
```

or

```
resetAge workweek
```

If *fullweek* is provided as a parameter, all defects or features that are in the open, working, design, size and review states will be aged according to a 7-day schedule.

If *workweek* is provided as a parameter, all defects or features that are in the open, working, design, size and review states will be aged according to a 5-day work week schedule.

Before you invoke this program the **CMVC\_FAMILY** environment variable must be set to the name of the family to which you want to age the defects and features. For DB2/6000, ORACLE, or SYBASE users, the **DB2\_PASS**, **ORACLE\_PASS**, or **SYBASE\_PASS** environment variable must be set to the password which permits the family to connect to the database.

The **resetAge** script requires a database license while it is running.

---

## Monitoring the Performance of Your CMVC Server

The following aspects of system loading should be considered when monitoring the performance of your CMVC server. Refer to your operating system documentation for more information.

- I/O activity

Disk devices have a limit in terms of I/O throughput. When examining the performance of your system make sure that you have not reached the limits of the I/O bandwidth of your system.

- CPU usage

Determine whether your CMVC server is CPU-bound. If your CPU has little or no idle time, then you have reached the limits of your CPU.

- Paging

For maximum performance, there should be a relatively small amount of paging activity during normal operations of your CMVC server.



---

## Tuning Your CMVC Server

Careful tuning of your family database can dramatically increase the performance of your CMVC server. Refer to your database documentation for instructions on tuning your system and database.

Run only as many **cmvcd** daemons as you need to adequately support the number of clients you have. You can monitor the activity of your **cmvcd** daemons by using the CMVC Activity Monitor.

---

## Monitoring the CMVC Server Daemons

The CMVC Activity Monitor is a real-time program that permits the CMVC family administrator to monitor the activity of the CMVC server daemons.

The CMVC Activity Monitor makes use of the server's shared memory space. Each CMVC daemon as well as the monitor itself attaches to the same shared memory segment. Each time a request is serviced by a CMVC server daemon, the shared memory segment for that particular CMVC server daemon is updated with information regarding the user who has requested the work and the nature of the request.

The CMVC Activity Monitor can be used in a number of different ways:

- To determine the activity of the CMVC server
- To determine which users issue time consuming reports
- To determine the total number of requests serviced by the CMVC server and the number serviced by each CMVC server daemon since the CMVC server daemons were started
- To determine whether there is a problem with one or more of the CMVC server daemons
- To clean up shared memory when the CMVC server daemons terminate abruptly.

To invoke the CMVC Activity Monitor, you must be logged on to the *familyname* account on the CMVC server. The **monitor** program is found in the **/usr/lpp/cmvc/bin** directory and follows the syntax shown in Figure 35.

---

```
monitor refreshInterval [width]
```

---

Where:

refreshInterval	Indicates the time in seconds between successive screen refreshes.
width	Indicates the number of characters of status information you wish to display for each CMVC server daemon. The default is 132 characters. The maximum is also 132 characters.

---

Figure 35. Syntax Statement for the Activity Monitor

An example of the CMVC Activity Monitor screen with 4 CMVC server daemons running and 2 daemons running commands is shown in Figure 36 on page 106.

```
4 of 4 cmvc daemons running. Shared mem size 2248
Press any key to quit.
Total hits=12902

01,19403,03526,

02,18661,03126,08/30/91, 12:34:38, FileDelete,gray,gray,gray,torolab,ib.
filexyz,c,fvtTest

03,16615,02982,

04,18924,03268,08/30/91.12:34:37,Report,harrison,harrison,harrison,
torolab.ib,DefectView,compName in ('fvTest') and state not in ('clos
```

*Figure 36. Example Activity Monitor Screen*

The first line in Figure 36 is a general information line. It indicates that all four of the CMVC server daemons are running, and that the monitor and the CMVC server daemons are using 2448 bytes of shared memory. The shared memory information is displayed for reference only. Some users may be concerned with shared memory usage on the machine. This informs these users how much shared memory the CMVC server and monitor program use. The first line also informs users of the CMVC Activity Monitor how to exit the program. Any key can be pressed to exit from the program.

The second line indicates the total number of requests, or hits, that have been serviced by the CMVC server since it was last invoked.

The rest of the lines are status lines, one for each CMVC server daemon. The format of a status line is as shown in Figure 37 on page 107.

Column Number	Data Displayed	Description
1	daemon index	The index number of the CMVC server daemon in the shared memory segment.
2	daemon process id	The process ID of the CMVC server daemon.
3	number of requests serviced	The number of requests serviced by the daemon since the time it was invoked.
4	date	The date the request was issued. Format is mm/dd/yy.
5	time	The time the request was issued. Format is hh:mm:ss.
6	CMVC action name	The CMVC request which is being serviced.
7	CMVC user ID	The CMVC user ID which issued the request.
8	user's operating system login	The login ID of the user who issued the CMVC request.
9	hostname	The <i>host</i> from which the CMVC request was issued.
10	status info	Additional information about the CMVC request being serviced.

Figure 37. Format of the Status Line in the Activity Monitor

If only the first three columns are displayed for a daemon, that CMVC server daemon is not servicing a request.

The amount of data that will be displayed beyond the first three columns is dependent on the value you entered for the width parameter. If you did not specify a width parameter, then 132 characters in total are displayed for each CMVC server daemon entry.

The screen is refreshed based on the `refreshInterval` specified. If the `refreshInterval` is too long, you may never see any activity occurring because the CMVC server daemon would receive and process the request before the refresh interval expired. A refresh interval of 1 or 2 seconds is usually sufficient.

If one or more of the CMVC server daemons terminates gracefully, that is, a **kill -15** has been issued to stop a CMVC server daemon, the first line of the monitor screen is updated to reflect the number of daemons that are still active. The CMVC server daemon that terminated gracefully has `--` displayed in the first column instead of the index value. If a CMVC request was being processed by a CMVC server daemon when it was terminated, the status for that action remains on the screen. The monitor keeps running even if all of the daemons are gracefully terminated. If you leave the monitor running and then restart the CMVC server daemons, the monitor detects these new daemons as soon as they are fully initialized and active. The CMVC Activity Monitor runs until you press any key to exit from the program.

If one or more of the CMVC server daemons is terminated abruptly, that is, a **kill -9** has been issued to stop a CMVC server daemon, the first line of the monitor screen is updated to reflect the number of daemons that are still active. The CMVC server daemon that terminated abruptly has **>>** displayed in the first column instead of the index value. If a CMVC request was being processed by a CMVC server daemon when it was terminated, the status for that action remains on the screen. The monitor keeps running, even if all of the daemons are terminated abruptly. If you leave the monitor running and then restart the CMVC server daemons, the monitor detects these new daemons as soon as they are fully initialized and active. The CMVC Activity Monitor will run until you press any key to exit from the program.

## Monitoring the Activity of the CMVC Server

The CMVC Activity Monitor can be used to determine whether there are enough or too few CMVC server daemons running for a particular family. By running the monitor and watching the activity of each CMVC server daemon, you can determine whether the CMVC server daemons are constantly in use or if they are rarely being used.

If they are constantly in use, this indicates that there is a lot of activity from CMVC users, and there may not be enough CMVC server daemons available to service user requests. Each CMVC server daemon can process only one request at a time. If all of the CMVC server daemons are busy servicing requests, other requests are rejected. User's whose requests are rejected receive a message indicating that a connection cannot be established with the family. Usually a request can be completed very quickly, but in some cases a request can take several seconds to complete if the data being requested is lengthy or if certain actions such as LevelCommit are performed. If this situation is encountered, you may want to stop CMVC and restart it with more CMVC server daemons, provided that your database license agreement permits you to do so.

If the CMVC server daemons are rarely used, you may have started more daemons than are necessary. In this case, stop CMVC and restart it with fewer CMVC server daemons. This is particularly important if you are using the database for purposes other than just CMVC. Each CMVC server daemon you start becomes a user of the database. By starting unnecessary CMVC daemons you decrease the number of people who can use the database for other purposes.

A minimum of 2 CMVC server daemons per family should be running. The number of CMVC server daemons you have running per family depends on the number of users of the family and the peak load of requests from users of CMVC. Since this differs for each installation, the family administrator should periodically monitor the activity of the CMVC server daemons and increase or decrease the number of daemons as appropriate.

## Determining Which Users Issue Time Consuming Reports

The CMVC Activity Monitor can also be used to determine which users are issuing reports that are too time consuming for peak load hours. Since the monitor displays information regarding the user who has requested the work and the nature of the request, it is possible to determine which requests are taking a lot of time. The family administrator can decide whether or not they will permit the user to continue with the request or whether they would rather cancel the request and allow the user to run such a request during non-peak hours. By issuing a **kill -1**

**pid**, where *pid* is the process id of the particular CMVC server daemon servicing the request, the family administrator can recycle a CMVC server daemon. This effectively cancels the request and prepares the CMVC server daemon for subsequent work requests.

## Monitoring the Number of Requests Serviced

The CMVC Activity Monitor displays the total number of requests, or hits, serviced by CMVC since the CMVC server daemons were started. The total number of requests serviced is distributed among the various CMVC server daemons. The third value displayed on each status line is the number of requests serviced by that particular CMVC server daemon. This data allows a family administrator to determine the extent to which CMVC is being used.

## Monitoring Server Daemon Problems

The CMVC Activity Monitor can also be used as a tool to determine whether there is a problem with any of the CMVC server daemons. The daemons may appear to be running, but may not be processing requests. Alternatively, one or more daemons could be held up processing a request.

When a CMVC server daemon appears to be held up processing a request, there could be one or more reasons for this behavior:

- A request can take a long time to process. Actions such as LevelCommit, LevelCheck -long, LevelExtract, ReleaseExtract, and Report can be time consuming.
- A request may be held pending the release of a lock of a database table. Certain actions such as LevelCommit need to lock some of the database tables so that other users do not destroy the data integrity before the command completes. If a database table is locked and an update request for that database table is received, the request will be held until the database table is unlocked. An update request is any request that modifies the contents of the information in a database table. Reports that *query* the contents of a locked database table can still be completed.

## Cleaning Up Shared Memory

If the CMVC server daemons terminate abruptly, shared memory could be attached to defunct processes. If this is the case, running the CMVC Activity Monitor will remove the shared memory so that it can be reused. If this shared memory is left attached to defunct processes, subsequent invocation of the CMVC server daemons may result in an inability of the monitor and the CMVC server daemons to reattach to shared memory. This generates an error message on the server (either on the console or in the **syslog**), indicating that there was a problem obtaining shared memory. The CMVC server daemons will continue to initialize without the use of shared memory, and the monitor functions will not be supported.

### Recovery From a Shared Memory Problem

To recover from a shared memory problem, you first need to determine which shared memory segments are still attached, and then detach them:

1. Stop the CMVC server daemons if they are running.
2. Log in to the *familyname* account on the CMVC server.

3. Type `ipcs` and look for the entry that specifies *familyname* as the owner. You will use the ID number in this entry for a parameter in step 4.

**Note:** The `ipcs` program needs root authority to run on the SunOS operating system.

4. Type `ipcrm -m ID`, where *ID* is the ID of the shared memory used by your family. This will detach the shared memory segment.
5. Restart the CMVC server daemons as described on page 95.

---

## Version Control Path Finder Tool

Files that are defined in the CMVC development environment are stored in a family-based version control file repository on the CMVC server. Each family has its own version control file repository that consists of either SCCS or PVCS files depending on the version control mechanism that the family is using. The version control file repository is designed as a hierarchical directory structure as defined by the CMVC server. The root of this directory structure is the `vc` directory, which is located in the CMVC *familyname* account's home directory.

Due to the hierarchical nature of the directory structure, and the fact that the file names are defined by the CMVC server and do not represent the same names as those within the CMVC development environment, a path finder tool is available, so that family administrators can determine the location of specific CMVC files within the version control file repository. In most cases there is no need to know where the file is located or by which name the file is stored within the version control file repository, however, the tool is available in case the need arises.

You should not tamper with the files in the version control file repository.

The path finder tool, `vcPath`, is located in the `/usr/lpp/cmvc/bin` directory on the CMVC server. To use the tool, you must be logged on to the CMVC family account on the CMVC server. The `CMVC_FAMILY` environment variable must be set to the name of the CMVC family. When using the DB2/6000, ORACLE, or SYBASE database, the `DB2_PASS`, `ORACLE_PASS`, or `SYBASE_PASS` environment variable must be set to the password that enables the family to connect to the database.

The `vcPath` tool requires a database connection while it is running.

Figure 38 shows the proper format for use of the `vcPath` command.

---

`vcPath` filePathName releaseName

---

Where:

filePathName	The name of the file that exists in the CMVC development environment.
releaseName	The name of the release that the file is associated with in the CMVC development environment

---

Figure 38. Syntax Statement for the `vcPath` Command

If you specify a valid file name and release name combination, a message will be displayed informing you where the file can be located in the version control file

repository. For example, if file **main.c** is a text file associated with release projectA and you enter:

```
vcPath main.c projectA
```

the following message is returned:

The file, main.c, associated with release, projectA, maps to the file, s.04, in the directory, \$HOME/vc/0/0/0/0, of the CMVC family's account on the CMVC server.

If file **main.cat** is a binary file associated with release projectA and you enter:

```
vcPath main.cat projectA
```

the following message is returned if SCCS is being used as the version control mechanism:

The file, main.cat, associated with release, projectA, maps to the s.binary file in the directory, \$HOME/vc/0/0/0/0/b.05, of the CMVC family's account on the CMVC server.

Each delta to the binary file resides in this directory as an individual file which is identified by its version number.

If SCCS is used and the record length of a text file is more than 510 characters (file type is long), the **vcPath** command displays the above s.binary message.

The following message would be returned if PVCS is being used as the version control mechanism:

The file, main.cat, associated with release, projectA, maps to the file, s.05, in the directory, \$HOME/vc/0/0/0/0, of the CMVC family's account on the CMVC server.

If the file specified does not exist in the release specified, or if the release specified does not exist, then a message is displayed indicating that the file could not be found in the release.

---

## Managing Level Maps

A CMVC level is a group of changed files in a particular release. When a level is committed, CMVC defines a map for the level. This map contains a list of file names, their versions, and the changes that are being committed for the level. As development with CMVC progresses and the number of committed levels increases, the number of maps that are created also increases.

Levels are usually cumulative in nature and only rarely will a previous level for a release be of value. When obsolete levels exist, the family administrator can use the **cmvcarchive** program to archive the maps that define those obsolete levels, thereby freeing storage for the creation of new level maps.

Level maps are created relative to the **\$HOME/maps** directory in the *familyname* account on the CMVC server. Since maps are created for particular releases, the individual level maps are created in the directory of the same name as the release they were created for. The maps are created in the same name as the level they represent. Therefore, the format of a level map pathname is:

```
$HOME/maps/<releaseName>/<levelName>
```

## Level Map File

The level map is a readable text file consisting of line entries that identify the name of the file, the database version id, the database file id, and the type of change that was performed on the file. Do not alter the contents of the level maps. The level maps are used when a user requests that a level be extracted. If the maps are tampered with the results of the extraction cannot be guaranteed.

## Maintaining the Maps Directory

The number of level maps will increase as the number of committed CMVC levels increases. Some levels will become obsolete and never be required by CMVC users again.

The family administrator can identify the names of level maps that can be archived by querying the list of committed levels and determining the date they were committed and talking to the owners of those levels. The CMVC level will still be known within the CMVC environment regardless of whether or not the level map resides in the maps directory on the CMVC server.

**Note:** If it becomes necessary to free up disk space by archiving level maps or any other CMVC files, you should only use the CMVC **cmvcarchive** program.

When restoring level maps, maintain the same directory format as described on page 111 and create the necessary directories. These directories should have permissions of 755 and the level map files should have permissions of 644.

---

## Errors

Errors encountered by CMVC are recorded in the syslog facility when **syslog** is activated. If **syslog** is not activated, error messages are sent to the console of the CMVC server workstation. Since multiple families may be supported by a single CMVC server, the messages logged in **syslog** may pertain to multiple families.

It is important to activate the **syslog** daemon so that CMVC errors and database errors can be logged for subsequent problem resolution. To activate the **syslog** daemon, first edit the **/etc/syslog.conf** file and add the following line, depending on the operating system you are using:

Operating System	Line to be Added
AIX	*.warning /usr/spool/syslog <b>Note:</b> If you are using the DB2/6000 database, you may want to avoid the many warning messages by setting *.error /usr/spool/syslog
HP-UX	*.warning /usr/adm/syslog
SunOS	*.warning /var/log/syslog
Solaris	*.warning /var/adm/messages

Then start the **syslog** daemon by issuing the following command, depending on the operating system you are using:



<b>Operating System</b>	<b>File</b>
AIX	/etc/syslogd <b>Note:</b> Use <code>stopsrc -s syslogd</code> to stop <b>syslogd</b> and <code>startsrc -s syslogd</code> to start <b>syslogd</b> .
HP-UX	/etc/syslogd
SunOS	/usr/etc/syslogd
Solaris	/usr/sbin/syslogd

**Notes:**

1. Use the **touch** command to create the log file if it does not exist.
2. When creating the log file, set permissions according to the directions for the operating system; for example in AIX, the permissions for **/usr/spool/syslog** should be read-write for owner and group, and read for others, with owner **root** and group **system**.
3. Remember to stop and restart **syslogd** after modifying **syslog.conf** and creating the log file.
4. Refer to the **/etc/syslog.conf** file for the location of the **syslog** file. Monitor this file at regular intervals so that any required maintenance or problem resolution can be performed.



---

## Chapter 15. The CMVC Audit Log

An audit log is maintained for each CMVC family. This audit log contains an entry for each successful or unsuccessful CMVC transaction, and therefore provides a history of your CMVC family. It also includes entries for each unauthorized attempt to access the CMVC server, so it can be used to audit your system's security.

This chapter describes the CMVC audit log, the type of information it contains, and how to manage it.

---

### Managing the CMVC Audit Log

The audit log is in the `.audit/log` file relative to the home directory of the *familyname* account on the CMVC server. It contains an entry for every CMVC transaction performed since your family was created.

Because the audit log contains information about unsuccessful transactions, it can be used to track down the source of problems. The audit log also contains information which can be used to identify users who make unauthorized attempts to access the CMVC server. Administrators should review the audit log periodically to check for unsuccessful transactions or attempts to violate the server's security.

### Prerequisite Tasks or Conditions

Log in to the *familyname* account.

### Cleaning Up the Log File

Information is continually appended to the end of the log file. To keep the audit log from growing too large, use the `cmvclog.cleanu` shell script found in the `/usr/lpp/cmvc/samples` directory on the CMVC server. This shell script maintains 4 progressively older copies of the `log` file, named `log.0`, `log.1`, `log.2`, and `log.3`. Each time the shell script is run, it moves the contents of each log file as shown below:

1. `log.2` overwrites `log.3`
2. `log.1` overwrites `log.2`
3. `log.0` overwrites `log.1`
4. `log` overwrites `log.0`

These actions remove the contents of the `$HOME/audit/log` file, allowing logging to start over with a new file. You can change this shell script to create more or fewer old copies of the log file, or rename the files to whatever you want for archive purposes.

The `cmvclog.cleanu` can be invoked manually or run at specified intervals by the `cron` daemon:

1. Log in to root.
2. Change the directory to `crontabs` by typing:  

```
cd /usr/spool/cron/crontabs
```
3. Edit the file for your *familyname* account by adding a line in the following format:

```
min hr day mth wkday /usr/lpp/cmvc/samples/cmvclog.cleau > /dev/null
```

where *min hr day mth wkday* represent the time period in which you want the shell script to run. For example, the following line would represent 8:00 AM on the first day of every month:

```
0 8 1 * * /usr/lpp/cmvc/samples/cmvclog.cleau > /dev/null
```

If your *familyname* file does not exist then create one.

4. Save the file and exit from your editor.
5. Notify the **cron** daemon of the change made to the file by reloading the file while still in the **crontabs** directory, as follows:

```
crontab <familyname>
```

**Note:** If you did not stop **cmvcd** before changing the audit log, **cmvcd** might not be able to locate the new log. Logging will stop until **cmvcd** is stopped and started again.

---

## Format of the CMVC Audit Log

Information about transactions is recorded in the audit log. Figure 39 on page 117 shows an example of an audit log file. The information in the audit log file is in the following format:

- **For unauthorized transactions**
  - Process ID number of the CMVC daemon
  - CMVC User ID of user who requested the action
  - Login at hostname
  - UNAUTHORIZED
  - Date
  - Time
  - Error message.
- **For authorized transactions**
  - Process ID number of the CMVC Daemon
  - CMVC action
  - SUCCESS or FAILURE
  - Date
  - Time
  - CMVC User ID
  - Login at hostname
  - Additional information for successful transactions or error messages for unsuccessful transactions.

The additional information included at each entry depends on the CMVC action specified in the entry. Figure 39 on page 117 shows an example of an **audit.log** file. Figure 40 on page 117 lists the additional information for each CMVC action.

---

31381,ReleaseCreate,FAILURE,03/17/93,11:06:34,sccstest,gray,gray.torolab.ibm.com,  
0010-427 A release already exists or previously existed with the name relC1.  
Specify a new name for this release.  
31410,ReleaseCreate,SUCCESS,03/17/93,11:07:05,sccstest,gray,gray.torolab.ibm.com,relH1  
31417,FileAdd,FAILURE,03/17/93,11:07:32,sccstest,gray,gray.torolab.ibm.com,  
0010-258 The requested action requires that you specify one or more  
defects or features.  
0010-263 File FILEH1.bin associated with release relC1 cannot be created.  
31423,FileAdd,SUCCESS,03/17/93,11:08:18,sccstest,gray,gray.torolab.ibm.com,FILEH1.bin,relH1,1.1  
31450,FileCheckOut,SUCCESS,03/17/93,11:09:03,sccstest,gray,gray.torolab.ibm.com,FILEH1.bin,relH1,1.1  
31240,FileCheckIn,SUCCESS,03/17/93,11:09:56,sccstest,gray,gray.torolab.ibm.com,FILEH1.bin,relH1,1.2  
31425,FileUndo,SUCCESS,03/17/93,11:11:54,sccstest,gray,gray.torolab.ibm.com,FILEH1.bin,relH1,delta,1.3  
31436,ReleaseCreate,SUCCESS,03/17/93,11:32:50,sccstest,gray,gray.torolab.ibm.com,rel1  
31446,FileLink,FAILURE,03/17/93,11:33:17,sccstest,gray,gray.torolab.ibm.com,  
0010-052 File FILEH1.bin associated with release rel1 was not found.  
31449,FileLink,SUCCESS,03/17/93,11:33:41,sccstest,gray,gray.torolab.ibm.com,FILEH1.bin,relH1,rel1,1.2  
31249,FileCheckOut,SUCCESS,03/17/93,11:35:08,sccstest,gray,gray.torolab.ibm.com,FILEH1.bin,rel1,1.1.1.1  
31259,FileCheckIn,SUCCESS,03/17/93,11:35:18,sccstest,gray,gray.torolab.ibm.com,FILEH1.bin,rel1,1.1.1.2  
24942,Transaction from joe/gray@gray.torolab.ibm.com was UNAUTHORIZED,03/18/93,09:43:11,  
0010-100 User joe was not found.

---

*Figure 39. Sample of an Audit Log File*

<b>CMVC Action</b>	<b>Additional Information</b>
<b>AccessCreate</b>	CMVC user ID, component name, authority group name
<b>AccessDelete</b>	CMVC user ID, component name
<b>AccessRestrict</b>	CMVC user ID, component name, authority group name
<b>ApprovalAbstain</b>	Release name, defect/feature identifier, approver's name
<b>ApprovalAccept</b>	Release name, defect/feature identifier, approver's name
<b>ApprovalAssign</b>	Release name, defect/feature identifier, new approver's name
<b>ApprovalCreate</b>	Defect/feature identifier, release name, approver's name
<b>ApprovalDelete</b>	Defect/feature identifier, release name, approver's name
<b>ApprovalReject</b>	Release name, defect/feature identifier, approver's name
<b>ApproverCreate</b>	CMVC user ID, release name
<b>ApproverDelete</b>	CMVC user ID, release name
<b>CompCreate</b>	New component name
<b>CompDelete</b>	Component name
<b>CompLink</b>	Component name, new parent component name
<b>CompModify</b>	Component name
<b>CompRecreate</b>	Component name
<b>CompUnlink</b>	Component name, parent component name
<b>CompView</b>	Component name
<b>CoreqCreate</b>	Release name, first defect/feature identifier, second defect/feature identifier
<b>CoreqDelete</b>	Release name, defect/feature identifier
<b>DefectAccept</b>	Defect identifier
<b>DefectAssign</b>	Defect identifier
<b>DefectCancel</b>	Defect identifier

*Figure 40 (Part 1 of 4). Additional Information Provided in the Audit Log*

<b>CMVC Action</b>	<b>Additional Information</b>
<b>DefectClose</b>	**This action is not audited**
<b>DefectComment</b>	Defect identifier
<b>DefectDesign</b>	Defect identifier
<b>DefectModify</b>	Defect identifier
<b>DefectOpen</b>	Defect identifier
<b>DefectReOpen</b>	Defect identifier
<b>DefectReturn</b>	Defect identifier
<b>DefectReview</b>	Defect identifier
<b>DefectSize</b>	Defect identifier
<b>DefectVerify</b>	Defect identifier
<b>DefectView</b>	Defect identifier
<b>EnvCreate</b>	Tester's CMVC user ID, release name, environment name
<b>EnvDelete</b>	Environment name, release name
<b>EnvModify</b>	Tester's CMVC user ID, release name, environment name
<b>FeatureAccept</b>	Feature identifier
<b>FeatureAssign</b>	Feature identifier
<b>FeatureCancel</b>	Feature identifier
<b>FeatureClose</b>	**This action is not audited**
<b>FeatureComment</b>	Feature identifier
<b>FeatureDesign</b>	Feature identifier
<b>FeatureModify</b>	Feature identifier
<b>FeatureOpen</b>	Feature identifier
<b>FeatureReOpen</b>	Feature identifier
<b>FeatureReturn</b>	Feature identifier
<b>FeatureReview</b>	Feature identifier
<b>FeatureSize</b>	Feature identifier
<b>FeatureVerify</b>	Feature identifier
<b>FeatureView</b>	Feature identifier
<b>FileAdd</b>	Path name, release name, SID
<b>FileCheckIn</b>	Path name, release name, SID
<b>FileCheckOut</b>	Path name, release name, SID
<b>FileDelete</b>	Path name, release name
<b>FileDestroy</b>	Path name, release name, SID
<b>FileExtract</b>	Path name, release name, SID
<b>FileForceIn</b>	**Audited via FileCheckIn**
<b>FileForceOut</b>	**Audited via FileCheckOut**
<b>FileLink</b>	Path name, release name, new release name, SID
<b>FileLock</b>	Path name, release name, SID

Figure 40 (Part 2 of 4). Additional Information Provided in the Audit Log

<b>CMVC Action</b>	<b>Additional Information</b>
<b>FileLockForce</b>	**Audited via FileLock**
<b>FileModify</b>	Path name, release name
<b>FileRecreate</b>	Path name, release name
<b>FileRecreaForce</b>	**Audited via FileRecreate**
<b>FileRename</b>	Path name, new path name, release name
<b>FileRenameForce</b>	**Audited via FileRename**
<b>FileResolve</b>	Base name, release name
<b>FileUndo</b>	Path name, release name, undo type, SID
<b>FileUndoForce</b>	**Audited via FileUndo**
<b>FileUnlock</b>	Path name, release name
<b>FileView</b>	Path name, release name
<b>FixActive</b>	Defect/feature identifier, release name, component name
<b>FixAssign</b>	Defect/feature identifier, release name, component name
<b>FixComplete</b>	Defect/feature identifier, release name, component name
<b>FixCreate</b>	Defect/feature identifier, release name, component name
<b>FixDelete</b>	Defect/feature identifier, release name, component name
<b>HostCreate</b>	CMVC user ID, host name, user login on host
<b>HostDelete</b>	CMVC user ID, user login on host, host name
<b>LevelAssign</b>	Level name, release name, new level owner's CMVC user ID
<b>LevelCheck</b>	Level name, release name
<b>LevelCommit</b>	Level name, release name
<b>LevelComplete</b>	Level name, release name
<b>LevelCreate</b>	Level name, release name
<b>LevelDelete</b>	Level name, release name
<b>LevelExtract</b>	Level name, release name
<b>LevelModify</b>	Level name, release name
<b>LevelView</b>	Level name, release name
<b>MemberCreate</b>	Level name, defect/feature identifier, release name
<b>MemberDelete</b>	Level name, defect/feature identifier, release name
<b>NotifyCreate</b>	CMVC user ID, component name, interest group
<b>NotifyDelete</b>	CMVC user ID, component name,
<b>ReleaseCreate</b>	New release name
<b>ReleaseDelete</b>	Release name, new release name
<b>ReleaseExtract</b>	Release name, new release name
<b>ReleaseLink</b>	Release name, new release name
<b>ReleaseModify</b>	Release name, new release name
<b>ReleaseRecreate</b>	Release name, mew release name
<b>ReleaseView</b>	Release name, new release name

Figure 40 (Part 3 of 4). Additional Information Provided in the Audit Log

<b>CMVC Action</b>	<b>Additional Information</b>
<b>Report</b>	**With -where flag: view name, criteria **With -help flag: help, none **With -testClient flag: test, none **With -testServer flag: test, none
<b>SizeAssign</b>	Defect/feature identifier, component name, release name
<b>SizeAccept</b>	Defect/feature identifier, component name, release name
<b>SizeCreate</b>	Defect/feature identifier, component name, release name
<b>SizeDelete</b>	Defect/feature identifier, component name, release name
<b>SizeReject</b>	Defect/feature identifier, component name, release name
<b>TestAbstain</b>	Defect identifier, release name, environment name, tester's CMVC user ID
<b>TestAccept</b>	Defect identifier, release name, environment name, tester's CMVC user ID
<b>TestAssign</b>	Defect identifier, environment name, new tester's CMVC user ID
<b>TestReject</b>	Defect identifier, release name, environment name, tester's CMVC user ID
<b>TrackAssign</b>	Defect/feature identifier, release name, new track owner's CMVC user ID
<b>TrackCancel</b>	Defect/feature identifier, release name
<b>TrackCheck</b>	Defect/feature identifier, release name, level name
<b>TrackCommit</b>	Defect/feature identifier, release name
<b>TrackComplete</b>	Defect/feature identifier, release name
<b>TrackCreate</b>	Defect/feature identifier, release name
<b>TrackFix</b>	Defect/feature identifier, release name
<b>TrackIntegrate</b>	Defect/feature identifier, release name
<b>TrackModify</b>	Defect/feature identifier, release name, target
<b>TrackTest</b>	Defect/feature identifier, release name
<b>TrackView</b>	Defect/feature identifier, release name
<b>UserCreate</b>	New user ID
<b>UserDelete</b>	User ID
<b>UserRecreate</b>	User ID
<b>UserModify</b>	User ID
<b>UserView</b>	**No additional information is audited**
<b>VerifyAbstain</b>	Defect/feature identifier, CMVC user ID
<b>VerifyAccept</b>	Defect/feature identifier, CMVC user ID
<b>VerifyAssign</b>	Defect/feature identifier, CMVC user ID of the new verification record owner
<b>VerifyReject</b>	Defect/feature identifier, CMVC user ID

Figure 40 (Part 4 of 4). Additional Information Provided in the Audit Log



---

## Chapter 16. Bringing SCCS Files under CMVC Control

Text files that are currently being developed with SCCS commands can be brought into the CMVC environment if the CMVC server version control environment is SCCS.

Read *IBM CMVC Concepts* before continuing with this chapter. The concepts of component, release and files within the CMVC environment must be thoroughly understood before you can bring any SCCS files into the CMVC environment and place them under CMVC control.

This chapter describes the processes of migrating and importing SCCS files into the CMVC environment.

**Note:** If you are using CMVC for Sun systems, you need a Korn shell to run the scripts that migrate or import files from SCCS into CMVC.

---

### Options for Bringing SCCS Files into CMVC

Two options are available for bringing SCCS files into the CMVC environment:

- Using the SCCS File Migration Function
- Using the SCCS File Import Function.

These two options differ significantly in the amount of space they require on the CMVC server, the amount of time it takes to copy the files, and the amount of advance preparation that is required. The choice of which option to use should be discussed by your entire development team and planned by the CMVC family administrators.

Read the following sections on each option before making any decisions.

### The SCCS File Migration Function

The SCCS File Migration Function imports into the CMVC environment all delta versions, or SCCS identifiers (SIDs), for each specified SCCS file found in specified directories. By migrating all SIDs, future development using CMVC can access all versions of the migrated files. When the files are migrated, they are assigned to a CMVC component for management purposes and a CMVC release for product related activities. The current owners and administrators of SCCS files and the person performing the migration must decide which version of each file a particular release is going to reference. You can reference different versions of one file through different releases; however, each version of a file does not have to be specifically assigned to a release. The CMVC server maintains delta information in case you wish to use other deltas of the file in the future.

#### Advantages

The advantage of migrating SCCS files instead of importing them is that all versions of all migrated files are available for future development using CMVC. The amount of preparatory time required for migrating SCCS files is less than is required for importing, because you do not have to decide which version of each file is most important.

## **Disadvantages**

The disadvantage of migrating SCCS files is that more space is used on the CMVC server. The delta information for each file that is migrated takes more space than that used by the one version of each file that is imported.

## **The SCCS File Import Function**

The SCCS File Import Function imports one delta version, or SCCS identifier (SID), for each specified SCCS file found in specified directories into the CMVC environment. By importing one SID, future development using CMVC can use this imported version of the file as a base. When the files are imported, they are assigned to a CMVC component for management purposes and a CMVC release for product related activities. Any one of the current SIDs of the SCCS files can be chosen as the one to be imported.

The goal of the import function is to allow users to access the most important version of a file and continue developing it from an initial state instead of cluttering the CMVC environment with information about versions of a file that may never be accessed.

## **Advantages**

The advantage of importing SCCS files instead of migrating is that it saves space on the CMVC server, because only one version of each file has to be saved and recorded. Also, future retrievals of a file will take less time because the deltas are compressed into one version when each file is imported. That is, if you import a file with a SID number of 1.6 into the CMVC environment, all of the deltas that are in that version are combined with a version number of 1.1 within the CMVC environment. When an imported file is checked out or extracted for the first time using CMVC, version 1.1 is retrieved. Version 1.1 has no deltas and therefore takes less time to retrieve.

## **Disadvantages**

The disadvantage of importing SCCS files is that the CMVC server retains information only about the file version that is imported. The CMVC server does not retain delta information for other versions of the file. Therefore, a CMVC user cannot access other versions of the file that may have existed in the original SCCS environment.

## **Deciding to Migrate or Import SCCS Files**

Users of CMVC must consider the advantages and disadvantages of both the migration and the import function. If you know for certain that you will need access to all versions of the files that are moved from the intrinsic SCCS environment, you should consider the migration function. If there are only occasional circumstances in which you require other versions of each file, you should consider the import function.

When you migrate files you use more storage space in the CMVC server than you do when you import the files. Therefore, your capacity to grow within the CMVC environment is reduced if you migrate files rather than import them.

You can selectively use the import function and the migration function together if you use different SCCS files for each. This allows you to import files for which you only need one version and to migrate files for which all versions are required.

---

## Preliminary Requirements and Planning

After choosing the function to use, you must plan your release and component structure so that versions of files are placed within appropriate releases and managed by appropriate components. This plan is important. Current SCCS file owners, prospective CMVC component and release owners, the user responsible for performing the migration or import procedure, the CMVC family administrator and the project leaders should all be involved in this planning.

### Prerequisites

All team members involved in the planning must be thoroughly familiar with CMVC functions and concepts. For a detailed discussion of the CMVC concepts and functions, see *IBM CMVC Concepts*. Pay special attention to the way CMVC uses components, releases, and files.

### Comparing SCCS Files and CMVC Files

CMVC files are managed by different control mechanisms than files within the native SCCS environment. A file under CMVC control is implicitly owned by the component owner and a particular version of a file is referenced by means of a release. Component owners decide who can access the files within their component and give access permission to certain users by means of a component access list.

You may be migrating or importing files that contain SCCS header flags which restrict certain users in the SCCS environment from making deltas to a file or retrieving certain versions of a file. When the file is migrated or imported into CMVC, any SCCS keywords imbedded in the body of the file are retained but SCCS header flags are removed. This is done so that CMVC can control who accesses the file. Users of the migration and import functions must know this so that they can decide which component should own a file and which files can be associated together and managed by the same component. After CMVC users have access to perform file-related activity at a component, they can work with any file that is managed by the component. Users of the migration and import functions should also know that all descendant components inherit access to a component, unless the access is specifically restricted.

### Planning Your Component and Release Structures

Before using the SCCS file migration or import function, you must design and create the component and release structures within your CMVC family to meet the needs of the development group. That is, you must define the component and release names and the component hierarchy. You must also choose the component and release owners and create the components and releases.

To design the structure that will work best for your team you need to have a conceptual view of the current directories that contain SCCS files that are to be imported or migrated. You must address the following questions:

1. Which directories contain SCCS files to be migrated or imported?

The names of the directories and the names of the files within the directories must be made available. If directories are nested and contain subdirectories, you have to decide whether you want to maintain this directory structure when you move the files into CMVC. If this is the situation, you must decide the

directory to use at the top of your tree. It is often a good idea to maintain directory structures so that when you use CMVC for build preparation activities you can extract the files from a release or level to the workstation where you are performing the build. The files are then extracted into the directory structure that you have maintained.

2. Are the files being used or are they obsolete?

If the files are being used in a current development effort, then you will want to import or migrate them. If the files are temporary files or if they have been replaced with other files, you probably do not want to import or migrate them.

You may want to use CMVC for all projects even if they are not currently being worked on. This will allow you to do maintenance on the files at a later date. Optionally, you could decide not to import or migrate old projects but keep them on backup storage media so that they are available if required and then import or migrate the files when they are needed.

3. Who currently owns the SCCS files?

Is there a restriction on who can access the files? Can this restriction be lifted? Can each file be associated with a CMVC component which manages many files or must a special component be created for certain files to control access?

4. Who will own the files in the CMVC environment?

When a file is created in the CMVC environment, it is assigned to a component. The component owner becomes the implicit owner of the file. You must find out the CMVC component names, identify the component owners, and design the component hierarchy. Descendant components inherit access, unless you restrict it. Therefore, you must consider access when you design your component hierarchy.

When a file is created in the CMVC environment, it is assigned to a release. You must determine the CMVC release names and identify the release owners.

5. Can you import the most recently created delta or do you want a particular version of a file?

When importing, you must decide which delta version of each SCCS file to import into the CMVC environment.

6. Can you migrate the most recently created delta or do you want a particular version of a file?

When migrating, you must choose the delta version of each file that is to be associated with a release. When migrating files that have multiple branches of development each branch should be identified with a CMVC release for future access.

7. Will a particular version of any SCCS files need to be referenced by more than one CMVC release?

A file with one version common to more than one release is a *common file*. For a detailed description of common files, refer to the *IBM CMVC Concepts* manual.

8. Do the files reside on a CMVC client workstation?

If your SCCS files reside on a machine that is not a CMVC client workstation, you must move the SCCS files and directory structure, or both, to a CMVC client workstation, or install the CMVC client software on that machine. This is

necessary because CMVC commands are issued in order to import or migrate the SCCS files into the CMVC environment.

## Migration and Import Requirements

The following requirements must be met before you can use the SCCS File Migration Function or the SCCS File Import Function:

1. CMVC must be installed, configured and the CMVC daemons for your family must be running.
2. The directories that contain SCCS files must reside on a CMVC client workstation.
3. The user performing the import or migration requires host access from the CMVC client workstation to the CMVC family to which you are migrating or importing.
4. You must create the CMVC releases and components for your SCCS files within the CMVC family to which you are migrating or importing.
5. The user performing the import or migration requires a CMVC authority group access that includes the FileAdd and FileLink actions on the components involved in the process.
6. No files with the same name as those to be migrated or imported can exist in any of the CMVC releases involved in the import or migration.
7. If a process containing the track subprocess is selected for the CMVC releases involved in the import or migration, then a defect or feature must exist with a track in the fix state for each of the releases involved.
8. Sufficient space must exist on the CMVC server and within the underlying database.
9. The client workstation user ID must have write access to the directory in which the import and migration shell scripts are being run, as described in "The SCCS File Migration Function."

---

## The SCCS File Migration Function

There are three stages involved in performing the SCCS File Migration:

1. Stage 1. Running the **Filemap** shell script

This stage consists of running a shell script that searches specified directories for specified SCCS files and creates a map file based on the user's selection criteria.

2. Stage 2. Running the **Filemigrate** shell script

This stage uses the map files that were created for one or more CMVC releases and generates a file that contains the CMVC commands required to migrate the files into the CMVC environment.

3. Stage 3. Running the **file.migrate** script

This stage consists of running the CMVC commands that were generated in stage 2.

The following sections describe each stage in detail.

## Stage 1: Running the Filemap Shell Script

Stage 1 of the SCCS File Migration function is the same as stage 1 of the SCCS File Import function.

The map file generated from the **Filemap** shell script is required for stage 2 of the migration or import. It is the means by which users of SCCS can indicate which SCCS files are to be brought into the CMVC environment, which versions of each file are to be associated with one or more releases, what name CMVC should give to each file, what component should manage each file, and which track is tracking the migration or import of the files into the CMVC environment. All of the entries in an individual map file correspond to files that are referenced by one CMVC release. The name of each generated map file is the name of the specified release.

Run the **Filemap** shell script once for each component managing files in a release. Each time you run the **Filemap** script, it creates a map file under the name of the release specified. Running it for each component that manages files that are grouped in one release adds entries to existing map files of the same name.

If you want to migrate or import SCCS files into more than one release, you must run the shell script with different input arguments to create map files for each CMVC release involved in the migration or import.

The **Filemap** shell script generates an SCCS to CMVC file migration map for one CMVC release. It uses values supplied by the user as selection criteria for searching directories for SCCS files that conform to the *s.* filename convention. When the script finds a file that matches the selection criteria, it makes an entry in a map file, corresponding to the release which refers to the file within the CMVC environment. The map file is created in the directory from which the **Filemap** script is invoked.

When the **Filemap** script encounters errors, such as, file not found, or directory not found, it displays an error message and attempts to continue to find files that match the selection criteria. If no matches are made, the user is informed that a map file was not created.

### Using the Filemap Shell Script

The **Filemap** script is found in the directory `/usr/lpp/cmvc/bin` on the CMVC client workstation. Figure 41 on page 127 displays the optional and required flags and arguments for the **Filemap** shell script.

---

```
Filemap -p releaseName -c componentName -d sccsDirName [-w] [-r SID] [-t] ([-n defect/feature number]...) {[pathNames][sccsfileNames]}
```

---

Where:

-p releaseName	Name of the release that will reference the files within the CMVC environment.
-c componentName	Name of the component that will manage the files within the CMVC environment.
-d sccsDirName	Directory from which the search begins for SCCS files.
-w	Indicates that subdirectories should be walked in search of SCCS files.
-r SID	Indicates the SID of the SCCS files that will be referenced by the release.
-t	Indicates that the most recently created delta of the SCCS files should be referenced by the release.
-n defect/featureNumber	Indicates a CMVC defect or feature identifier that is associated with a release for which the process includes the track subprocess.
pathNames	Indicates directories relative to the sccsDirName that should be searched for SCCS files. Each pathName should be separated by at least one blank. pathNames of '*' are not supported.
sccsFileNames	Indicates which SCCS files should be migrated or imported. Use this parameter only to select specific files instead of selecting all files in a pathName. Each sccsFileName should be separated by at least one blank, sccsFileNames of '*' are not supported.

---

Figure 41. Filemap Syntax Statement

The *pathNames* argument or the *sccsDirNames* argument controls which directories are searched and which SCCS files are selected. Only one **-d** flag is permitted.

The **-w** flag is an optional flag indicating that any subdirectories found, relative to the specified *pathNames*, should also be walked so that SCCS files within those subdirectories can also be selected. Only one **-w** flag is permitted.

The **-r** flag is an optional flag that identifies a specific SID, an SCCS release, or an SCCS release and level of the SCCS files that will be referenced by the CMVC release that is specified by the **-p** *releaseName* flag. See your operating system documentation for more information about SCCS releases and levels.

If you are using the SCCS File Migration Function all versions of the file will be migrated, but one version must be chosen as the one to be referenced by the release. When migrating a file with multiple branches of development, you should define a CMVC release for the tip version of each branch of the file and run subsequent **Filemap** executions to generate the appropriate map file. This makes all branches immediately accessible within CMVC without having to link a release to them in the future.

If you are using the SCCS File Import Function, only one version of the file is imported. This version is referenced by the release. If the specified SID does not exist for a particular SCCS file, an error message is displayed and the script

continues to search for the next SCCS file that matches the selection criteria. Only one **-r** flag is permitted.

If the **-r** flag is omitted, the **-t** flag controls the SID that is selected. If both the **-r** and **-t** flags are absent, the SID with the largest SCCS release and level is selected. If both the **-r** and the **-t** flags are provided, the most recently created delta for the SCCS release and level specified is created.

The **-t** flag is an optional flag indicating that the most recently created delta of the selected SCCS files should be used as the version of the files to be referenced by the release for which the map is being generated. Only one **-t** flag is permitted. If the **-t** flag is omitted, the **-r** flag controls which SID is selected. If both the **-r** and **-t** flags are absent, the SID with the largest SCCS release and level is selected. If both the **-r** and the **-t** flags are provided, the most recently created delta for the SCCS release and level specified is created.

The **-n** flag is an optional flag identifying a CMVC defect or feature identifier that is tracking the work done for CMVC releases for which a process that includes the track subprocess is selected. If you are importing or migrating files into the CMVC environment, and they are going to be grouped by releases that have the track subprocess included in their process, you must provide the defect or feature numbers with this flag. The track for the defect or feature in this release must be in the *fix* state before the files can be migrated or imported successfully. You must provide a separate **-n** flag for each defect or feature that is tracking the work done for a particular release. When you specify a defect or feature, you provide the identifier as an argument to the **-n** flag. You do not indicate whether this identifier references a defect or a feature. The file of CMVC commands generated after stage 2 defaults to a **-defect** flag.

The *pathNames* argument specifies the directories relative to the *sccsDirName* directory that are to be searched for SCCS files. Multiple *pathNames* may be specified but each should be separated by a blank space. If you want to process the *sccsDirName* directory use a *pathName* of *'.'*. If you provide the **-w** flag, any subdirectories that are found relative to the *pathNames* are also searched. Only files within a directory that conform to the *s.* filename convention are considered. If a file is encountered that is of the *s.* format but it is not an SCCS file an error will be generated and the script will continue.

The *sccsfileNames* argument specifies the individual SCCS file names of the SCCS files to be migrated or imported. Use this parameter only if you need to select specific files instead of selecting all of the files in a *pathName*. You may specify multiple *sccsfileNames*. Each should be separated by a blank space and should specify a name or path that is relative to the *sccsDirName* directory. If a specified SCCS file does not exist an error will be generated and the script will continue.

## The Map File

As the **Filemap** script is processed, entries are made in a map file. The map file is created in the directory from which you start the script. You can only invoke the **Filemap** script from a directory for which you have write access permission. To obtain information about the SCCS files, you must have permission to read all SCCS files.



The name of the map file that is generated corresponds to the name of the CMVC release that groups the files within the CMVC environment. Each entry in a map file contains the following:

- Name of the directory in which the SCCS file resides or the name of the directory that is being used as the top of the tree structure, if the directory structure is to be maintained within the CMVC environment
- Name of the CMVC component that manages the file
- Name of the file, including any path name that will be associated with it within the CMVC environment
- Version number or SID of the file which will be referred to by the CMVC release specified by the name of the map file
- Defect or feature identifier (optional) that tracks the work being done for the release, if the process for that release includes the track subprocess

All of the entries in a map file correspond to files that will be referenced by the release specified by the map file name. Each entry in the map file corresponds to the format shown in Figure 42.

---

sccsDirName componentName filePathName SID [defect/feature number]

---

Where:

sccsDirName	Same as that provided with the <b>-d</b> sccsDirName flag. It indicates the top of the directory tree which contains the SCCS file corresponding to the particular entry in the map file.
componentName	Same as that provided with the <b>-c</b> componentName flag. It indicates the name of the component that manages the file.
filePathName	Represents the path name of the SCCS file relative to sccsDirName. This is the name of the file as it will be known in the CMVC environment.
SID	Indicates the version of the file that is going to be referred to by the CMVC release.
defect/feature number	Is included only if the <b>-n</b> flag was supplied when the script was processed.

---

*Figure 42. Format of the Map Files Generated by the Filemap Shell Script*

Figure 43 shows an example of the contents of a map file for the CMVC release, userA\_r1.

---

```

/u/sccsTree compA UserA/include/fileAA.h 2.2.1.6 -defect 243
/u/sccsTree compA UserA/include/fileAB.h 1.5 -defect 243
/u/sccsTree compA UserA/include/fileAC.h 3.2 -defect 243
/u/sccsTree compB UserB/source/fileBA.c 2.1 -defect 243
/u/sccsTree compB UserB/source/fileBB.c 1.2 -defect 243
/u/sccsTree compC UserC/source/fileCA.c 1.2.1.1 -defect 243
/u/sccsTree compB UserB/source/fileBC.asm 1.1 -defect 243

```

---

*Figure 43. Sample Map File for Release userA\_r1*

Do not modify the permissions of the map file. Both the **Filemap** script, the **Filemigrate** script, and the **Fileimport** script must read the map file to continue with the next stages of the SCCS File Migration function or Import Function.

### Common Files and Shared Files

If you are migrating SCCS files into different CMVC releases, you should understand the concepts of common and shared files. For a detailed description of common and shared files, refer to the *IBM CMVC Concepts* manual. If you rerun the **Filemap** script for several CMVC releases, and you specify the same SCCS files with different SIDs, these files become *shared files* within the CMVC environment. If you rerun the **Filemap** script for several CMVC releases, and you specify the same SCCS files and the same SIDs, these files become common files within the CMVC environment.

### Rerunning the Filemap Shell Script

If you run the **Filemap** script for a release and make a mistake in the input parameters, you can either change the map file manually for the corresponding release or rerun the **Filemap** script for that release. If you choose to rerun the **Filemap** script for a release, you must first delete the map file that was originally generated for that release. Otherwise, the **Filemap** script adds your new map file entries to the end of the existing map file and you will not achieve the results you want.

If you proceed to stage 3, and migrate or import the SCCS files into the CMVC environment, you should not return to this first stage and rerun the **Filemap** in reference to the same SCCS files. The SCCS files are already created within the CMVC environment after stage 3. Any attempt to migrate or import them again produces an error indicating that these files already exist under CMVC control. Sequential migration or import of files can occur as long as you refer to different SCCS files each time you perform the SCCS File Migration or Import function.

## Stage 2: Running the Filemigrate Shell Script

The second stage of the SCCS File Migration function uses the map files that were created for one or more CMVC releases and generates a file that contains the required CMVC commands needed to create files within the CMVC environment.

### Using the Filemigrate Shell Script

The **Filemigrate** shell script is found in the `/usr/lpp/cmvc/bin` directory on the CMVC client workstation.

Figure 44 displays the correct syntax for use of the **Filemigrate** shell script.

---

<b>Filemigrate</b> {mapfileNames...}	
Where	
mapfileNames	Indicates the map files that are to be simultaneously processed

---

Figure 44. Filemigrate Syntax Statement

If an SCCS file is to be migrated into a CMVC release, and then referenced by one or more other releases, the releases must be processed at the same time. The

script issues a CMVC **Migrate** command or a CMVC **File -link** command, if appropriate.

### The file.migrate Commands File

The **Filemigrate** shell script reads and processes the map files and creates the commands file **file.migrate** in the current working directory. This file contains the CMVC commands that create the files within the CMVC environment for the specified components and releases.

When the **Filemigrate** script is completed, a message is displayed that indicates an estimate of the number of 1K blocks that are required on the CMVC server in order to migrate the SCCS files into the CMVC environment. The system administrator should verify that this amount of room is available on the CMVC server before the commands in the **file.migrate** file are executed. The relational database you are using also must have room to bring the files into the CMVC environment.

The contents of the **file.migrate** file are overwritten each time the **Filemigrate** script is executed. If you are performing sequential migrations, rename the **file.migrate** file after running the **Filemigrate** script. When you proceed to stage 3, substitute the new name for the **file.migrate** file.

## Stage 3: Running the Commands in the file.migrate File

Stage 3 of the SCCS File Migration function is the same as stage 3 of the SCCS File Import function.

Stage 3 of the SCCS File Migration function consists of running the command file which was created in stage 2. This file is called **file.migrate** if you ran the **Filemigrate** shell script, or **file.import** if you ran the **Fileimport** shell script. Ensure that the preliminary requirements in “Migration and Import Requirements” on page 125 are met before running these commands files.

The files are placed in the version control directory **\$HOME/familyHome/vc** in the *familyname* account on the CMVC server. The **CMVC\_FAMILY** and the **CMVC\_BECOME** environment variables should be set so that the CMVC commands are directed towards the correct CMVC family and that the user performing the file migration or import is using the correct CMVC user ID.

After performing these steps, you can decide which of the following approaches you want to take:

1. Execute the commands in the foreground and wait until they complete before terminating the login session.
2. Execute the commands in the background to free your terminal for other tasks.
3. Use the **xecit** script to monitor the execution of the commands in the **file.migrate** or **file.import** file. With the **xecit** script, if processing of the commands is halted for any reason, the command execution can resume from the last successfully executed command.

**Note:** Use the **xecit** script for recovery purposes. See “Using the xecit Shell Script” on page 132 for more information.

To execute the commands in the foreground enter **file.migrate** at the system prompt, if you are using the **file.migrate** command file. If you are using the **file.import** command file, replace **file.migrate** with **file.import**. If you renamed

the command file during stage 2, substitute that name for `file.migrate` or `file.import`.

To execute the commands in the background, type `file.migrate &` at the system prompt if you are using the **file.migrate** command file. If you are using the **file.import** command file, replace `file.migrate &` with `file.import &`. If you renamed the command file during stage 2, substitute that name for `file.migrate` or `file.import`.

Each command in the command file is processed consecutively until all of the commands are completed. The amount of time this takes is proportional to the number of commands in your command file.

### Using the **xecit** Shell Script

To monitor the execution of the commands, you must run the **xecit** script found in the `/usr/lpp/cmvc/bin` directory on the CMVC client. This script reads and processes each line in the command file. This script creates an error file for all commands that return a nonzero exit status. After the **xecit** script is completed, check the *current directory* for the error file **xecit.errs**. If the file exists, browse it to determine the errors that were encountered during the file migration or import. If the file does not exist, the files were successfully migrated or imported into the CMVC environment. The **xecit** script also creates the file **xecit.tally** which is used to keep track of the commands that are executed successfully. If the execution of the **xecit** script is interrupted for any reason, you can restart the script. The tally file indicates where the processing should be restarted.

The **xecit.errs** file indicates which command was executed before the interrupt occurred. In a successful restart, both **xecit.errs** and **xecit.tally** files are removed from the current directory.

Figure 45 displays the optional and required flags and arguments for the **xecit** shell script.

---

```
xecit [-v] {commandfileNames...}
```

---

Where:

<code>-v</code>	Indicates that a verbose mode is desired
<code>commandFileNames</code>	Indicates the name of one or more command files

---

Figure 45. The **xecit** Syntax Statement

You can execute the **xecit** script from any directory, if you qualify the *commandFileName* argument and provide the full directory path name so that the **xecit** script can find the command files.

## Post-Migration Activities

You should perform the following activities after successfully migrating the desired files into the CMVC environment:

- Change the access permissions to the existing directories that contain the SCCS files that were migrated. This discourages further development using the intrinsic SCCS commands and encourages the use of the CMVC commands and actions.

- Keep the map files available for future reference. This allows you to discover which files were migrated and which CMVC releases are associated with one or more versions of the files.
- After a sufficient period of time has passed, store the directories which contain the original SCCS files on an alternative storage media and remove them from the file system.
- If you are migrating SCCS files into a release for which a process including the track subprocess is selected, you should commit the files by committing a level that includes a track with the defect or feature identifier used during the migration process, or integrate the track if the level subprocess is not included in the process managing the release.

---

## The SCCS File Import Function

There are three stages involved in performing the SCCS File Import:

1. Stage 1: Running the **Filemap** shell script

This stage runs a shell script that searches specified directories for specified SCCS files and generates a map file based on the user's selection criteria.

2. Stage 2: Running the **Fileimport** shell script

This stage uses the map files that were created for one or more CMVC releases, creates a source tree, and generates a file that contains the CMVC commands required to import the files into the CMVC environment.

3. Stage 3: Running the **file.import** script

This stage consists of running the CMVC commands that were generated in stage 2.

The following sections describe each stage.

### Stage 1: Running the Filemap Shell Script

Stage 1 of the SCCS File Import function is the same as stage 1 of the SCCS File Migration function. See “Stage 1: Running the Filemap Shell Script” on page 126 for a detailed description of using the **Filemap** shell script.

### Stage 2: Running the Fileimport Shell Script

The second stage of the SCCS File Import function uses the map files generated in stage 1 and creates a source tree containing all of the versions of the files to be imported. From this source tree it generates a file that contains the CMVC commands that are necessary to create files within the CMVC environment.

#### Using the Fileimport Shell Script

The **Fileimport** shell script is found in the **/usr/lpp/cmvc/bin** directory on the CMVC client workstation. The **Fileimport** script must be run from the directory that contains the map files that were generated in stage 1.

Figure 46 on page 134 displays the optional and required flags and arguments for the **Fileimport** script.

---

**Fileimport** [-d workDirName] {mapFileNames...}

Where:

-d workDirName	Indicates a temporary read/write working directory to be used by the script to hold a copy of the files after they are obtained from SCCS and before they are imported into the CMVC environment
mapFileNames	Indicates which map files are to be processed simultaneously

---

Figure 46. Fileimport Syntax Statement

The optional **-d** flag allows you to specify a temporary working directory which the **Fileimport** script can access with read/write authority. If the flag is not provided, the script uses the **/tmp** directory. A working directory is required so that **Fileimport** can import the same file for more than one CMVC release. The **Fileimport** script uses the *workDirName* or **/tmp** as the base directory and creates a subdirectory for each CMVC release that it processes. The specified version of each file is retrieved from SCCS and temporarily stored in the appropriate subdirectory of the working directory. The current directory which contains the map files should not be used as the temporary working directory for **Fileimport**.

The *mapFileNames* argument indicates the map files that are to be processed at the same time by the **Fileimport** script. If an SCCS file is to be imported into a CMVC release and then referenced by one or more other releases, these releases must all be processed at the same time. This enables the script to issue a CMVC **File -add** command or a CMVC **File -link** command, if appropriate.

### The file.import Commands File

The **Fileimport** shell script reads and processes the map files and creates a commands file called **file.import** in the current working directory. This file contains the CMVC commands that create the files within the CMVC environment for the specified components and releases.

When the **Fileimport** script is completed, a message is displayed that indicates an estimate of the number of 1K blocks that are required on the CMVC server to import the SCCS files into the CMVC environment. The system administrator should verify that this space is available on the CMVC server before the commands in the **file.import** file are executed. The relational database you are using also must have room to bring the files into the CMVC environment.

The contents of the **file.import** file are overwritten each time the **Fileimport** script is executed. If you are performing sequential imports, rename the **file.import** file after running the **Fileimport** script. When you proceed to stage 3, substitute the new name for the **file.import** file.

## Stage 3: Running the Commands in the file.import File

Stage 3 of the SCCS File Import function is the same as stage 3 of the SCCS File Migration function. See “Stage 3: Running the Commands in the file.migrate File” on page 131 for a detailed description of how to run the commands in the **file.import** file.

## Post-Migration Activities

After you have successfully imported the desired files into the CMVC environment, you should do the following:

- Change the access permissions to the existing directories that contain the SCCS files that were imported. This discourages further development using the intrinsic SCCS commands and encourages the use of the CMVC commands and actions.
- Keep the map files available for future reference. The files that are imported are created within the CMVC environment with a version number of 1.1. If you need to determine which version of a particular file was imported you need to reference the map files.
- After a sufficient period of time has passed, store the directories which contain the original SCCS files on an alternative storage media and remove them from the file system.
- If you imported SCCS files into the CMVC environment and associated them with one or more releases managed by processes that include the track and level subprocesses, you should commit these files by committing a level for each of the releases or integrate the track itself if the process which manages the release associated with the track does not include the level subprocess. The defect or feature that was used to import the files should be specified by a *level member* in the level you commit.

The directory that was used as the working directory to hold a copy of the SCCS files after they were obtained from SCCS and prior to importing them into the CMVC environment exists after you have completed importing. You may delete the subdirectories within this working directory.





---

## Chapter 17. Backup and Recovery

After CMVC is in use, you need to think about backing up your CMVC family's file systems, directories, and files, as well as the database tables, views and indexes. This chapter makes recommendations for a backup strategy and describes how to recover a CMVC family.

CMVC also provides an archive function, which allows you to free up file system and database space by archiving levels and releases. See Chapter 18, "Archiving and Restoring" for a description of the archive function and information on how to restore archived data.

---

### Backing Up the CMVC Server

Your CMVC family's file systems, directories, and files, as well as the database tables, views and indexes are critical to the operation of CMVC and represent a significant investment of time and effort. You want to develop a backup strategy that includes regular backups of these critical resources to ensure immediate recovery of recent versions of files, directories, and database information when necessary.

IBM recommends that you backup your CMVC information on a daily basis by doing the following:

1. Issue the **stopCMVC** command to stop the CMVC server daemons. See "Stopping the CMVC Server Daemons" on page 98 for more information.
2. Shut down the database. This insures data consistency during backup and reduces the possibility of a database crash.
3. Back up all database and CMVC data. Do this in such a way that you can easily restore them both if necessary, such as having both backups on the same tape.

**Note:** To reduce the impact to users during backup, first back up the database, then run the database and the CMVC daemons in maintenance mode during the SCCS or PVCS data backup. This gives users read-only access to CMVC and minimizes the time that CMVC is unavailable to them. After the backup is complete, restart the daemons in normal mode.

You can use the backup script called **backupCMVC** that is in the `/usr/lpp/cmvc/samples` directory to perform the backup steps. Use the **backupCMVC** script as it is written to backup a DB2 database. To back up a database other than DB2, make the appropriate changes to the script. Refer to your operating system or database documentation to understand what changes are appropriate for your environment.

Use the **cron** utility to automatically start the **backupCMVC** script at a time when system activity is low. Submit the backup job using the root user ID (during the backup, the user ID changes to the appropriate ID for each family).

Use the following syntax to run the **backupCMVC** script:

---

**backupCMVC** <device> <family>

---

Where:

<i>device</i>	Indicates the backup device (for example, /dev/rmt0)
<i>family</i>	Is a list of family user accounts to be backed up, separated by spaces (for example, family1 family2 family3)

---

Figure 47. Syntax Statement for backupCMVC

**Notes:**

1. The CMVC\_DAEMONS variable must be set in the <family> environment. CMVC\_DAEMONS determines how many cmvcd processes will be started.
2. Data will be backed up to **/home/<family>/db2backup** before tape backup. **/home/<family>/db2backup** is created automatically.
3. DB2\_PASS must be set in the <family> environment.
4. Logs are cleaned up on the first day of each month. Any mount points in **/tmp** are also cleaned up.
5. Failures of the **backupCMVC** command return nonzero exit codes.

It is important that the information in the CMVC family directories and the CMVC family information stored in the database are synchronized. Files created and defined in the CMVC family are stored in the family's directories, whereas, all other data is stored in the database. If these two sources of information become out of sync, unexpected results can occur.

The DB2, ORACLE, INFORMIX and SYBASE databases have sophisticated utilities to enable a recovery to the status just prior to the failure. This type of strategy will enable correct operation of CMVC only if you are also performing frequent backups of the CMVC family's directories during the course of a day.

Store the backup data on external media, preferably tape. The CMVC family directories and database tables, views, and indexes should always be backed up in the same session. If either the CMVC family directory backup or the database backup fails, the one that succeeds is not useful until you can successfully back up both of these sources. It is important that you keep the information synchronized. Always check for successful completion of the backup of each of these sources.

## Backing Up CMVC Family Database Tables, Views, and Indexes

Backing up the relational database tables, views and indexes that are used by your CMVC family is an important part of your backup process. For information on how to back up your relational database refer to the documentation supplied with the database products you are using.

The following information describes how the CMVC family database tables, views and indexes are organized. Use this information to decide on the best backup process for your organization.

## The DB2/6000 Database

Each CMVC family has a DB2/6000 database which is identical to the CMVC *familyname*. This database was created by the **mkdb** CMVC server command.

The authentication of users to DB2/6000 database is done when they logon to the AIX operating system. This means the password of the CMVC family for access to DB2/6000 is the same of the password of CMVC family AIX user ID.

The directory path on which the CMVC family database resides is specified in the DB2\_DBPATH environment variable, if it is defined. Otherwise, it will be the *HOME* directory of the owner of the DB2/6000 instance that the CMVC family is using. The directory path can also be found using the following DB2 command:

```
db2 list database directory
```

For details of the backup and restore commands of DB2/6000, refer to the book *IBM DATABASE 2 AIX/6000 Command Reference* (SC09-1575).

## The ORACLE Database

Each CMVC family has an ORACLE user account which is identical to the CMVC *familyname*. The password for this user account can be found in the ORACLE\_PASS environment variable in the login profile of the *familyname* account.

All the CMVC family tables, views, and indexes in ORACLE are owned by the *familyname* ORACLE user account.

When you first created your family's database, you either put all tables and indexes in the default tablespace, or you specified alternate tablespaces. If you specified alternate tablespaces then the names of these tablespaces can be found in the ORACLE\_TBLSP and ORACLE\_NDXSP environment variables for tables and indexes respectively. These environment variables exist in the login profile of your *familyname* account.

## The INFORMIX Database

Each CMVC family has an INFORMIX user account which is identical to the CMVC *familyname*. Each family also has an INFORMIX database which was created with the LOG and MODE ANSI options.

All the CMVC family tables, views, and indexes in INFORMIX are owned by the INFORMIX *familyname* user account.

When you created your family's database, you either stored the database in the default dbspace or you specified an alternate dbspace. If you specified an alternate dbspace then the name of the dbspace can be found in the INFORMIX\_DBSP environment variable. This environment variable exists in the login profile of your *familyname* account.

## The SYBASE Database

Each CMVC family has a SYBASE database which is identical to the *familyname*. This database was created by the SYBASE system administrator (*sa*).

Each CMVC family also has a SYBASE login which is identical to the CMVC *familyname*. You can find the password for this login in the SYBASE\_PASS environment variable in the login profile of the *familyname* account. The default database for this SYBASE login is the CMVC family's SYBASE database.

An alias is created for the CMVC family's SYBASE login to act as the database owner (dbo) of the CMVC family's SYBASE database. All CMVC family tables, views and indexes in SYBASE are owned by the alias *dbo* of the *familyname* SYBASE login.

When you created your family's database and log, you either placed them in the SYBASE default device or specified alternate devices. If you specified alternate devices then you can find their names in the SYBASE\_DBDEV and/or SYBASE\_LOGDEV environment variables in the login profile of the *familyname* account.

**Note:** If you do not periodically back up your family database or transaction log, then the transaction log for the database will eventually run out of space. You can use the **sp\_dboption** command and select the **trunc. log on chkpt.** option for your family database. This command refreshes the transaction log when the SYBASE server does a checkpoint. Your decision should be compatible with your backup and recovery strategy. Consult your SYBASE documentation for more information.

---

## Recovering CMVC

When recovering a CMVC family, it is important that the CMVC family directories and files remain synchronized with the contents of the database otherwise unexpected results can occur. The family's directories contain files that have been created and defined under CMVC control. The database contains all other data critical to the operation of CMVC.

Refer to your operating system and database documentation for more information on the restoring process.

If the CMVC server experiences a power shortage and this does not result in media failure, the CMVC daemons can usually be restarted correctly without the need for a complete recovery from your backup media. If you experience problems in the operation of CMVC after such an event, you may have to recover the CMVC family directories, files and database.

When performing a recovery from media failure, stop the CMVC server **cmvcd** and **notifyd** daemons. Recover the directories and files from the backup media, and refer to the database administrator guides for specific database recovery procedures. After the database has been recovered and restarted, start the CMVC server **cmvcd** and **notifyd** daemons for each CMVC family affected.

---

## Chapter 18. Archiving and Restoring

CMVC allows administrators to reclaim file system and database space by archiving obsolete or completed projects. Archived data can be restored as required.

The archive and restore functions are designed to be run by the CMVC family administrator. You must know how the CMVC files, file systems, and database tables are organized. Refer to Chapter 17, "Backup and Recovery" for more information about the files and databases used by CMVC.

### Warning:

- You must not issue operating system or version control system commands against the files in the version control directories. Any tampering with these files can result in discrepancies between the information stored in the relational database on the CMVC server and the information in the version control directories.
- We recommend a complete backup of the database and of the file system of the CMVC family account before you attempt to use the CMVC archive function.

---

### Archiving and Restoring CMVC Data

Family administrators or designated users can archive and restore releases, or selected levels of a release, and their related objects. The archive and restore functions are provided by two programs, **cmvcarchive** and **cmvcrestore**, which reside in **/usr/lpp/cmvc/bin** on the CMVC server.

Archiving and restoring must be performed from the CMVC server. To control the integrity of the data being archived or restored, the **cmvcd** daemons must be stopped before archiving or restoring. The archive and restore programs cannot be run while the **cmvcd** daemons are running. Administrators can restart the **cmvcd** daemons in the maintenance mode, thus providing users with read-only access to CMVC during the archive or restore process. Any group of CMVC users who want to archive their data must contact the family administrator and request the archive. The family administrator can then schedule the activity during a maintenance period.

Before running the archive program, the administrator should perform a full backup of the CMVC family's account on the CMVC server, and should export and back up the database tables used by the family.

When the archive program is run, the data (releases or levels and their related objects) being archived may be removed from the CMVC family, if necessary. This recaptures space from the database tables, as well as space within the file system owned by the family's account on the CMVC server.

The restore function allows archived data to be restored into a new CMVC family for maintenance activities or further development.

Certain prerequisites pertaining to CMVC objects must be satisfied before either of the archive or restore programs can be executed. Refer to “Archive and Restore Preparation” on page 143 for more information.

Figure 48 shows a simplified example of a family in which two levels have been archived and restored.

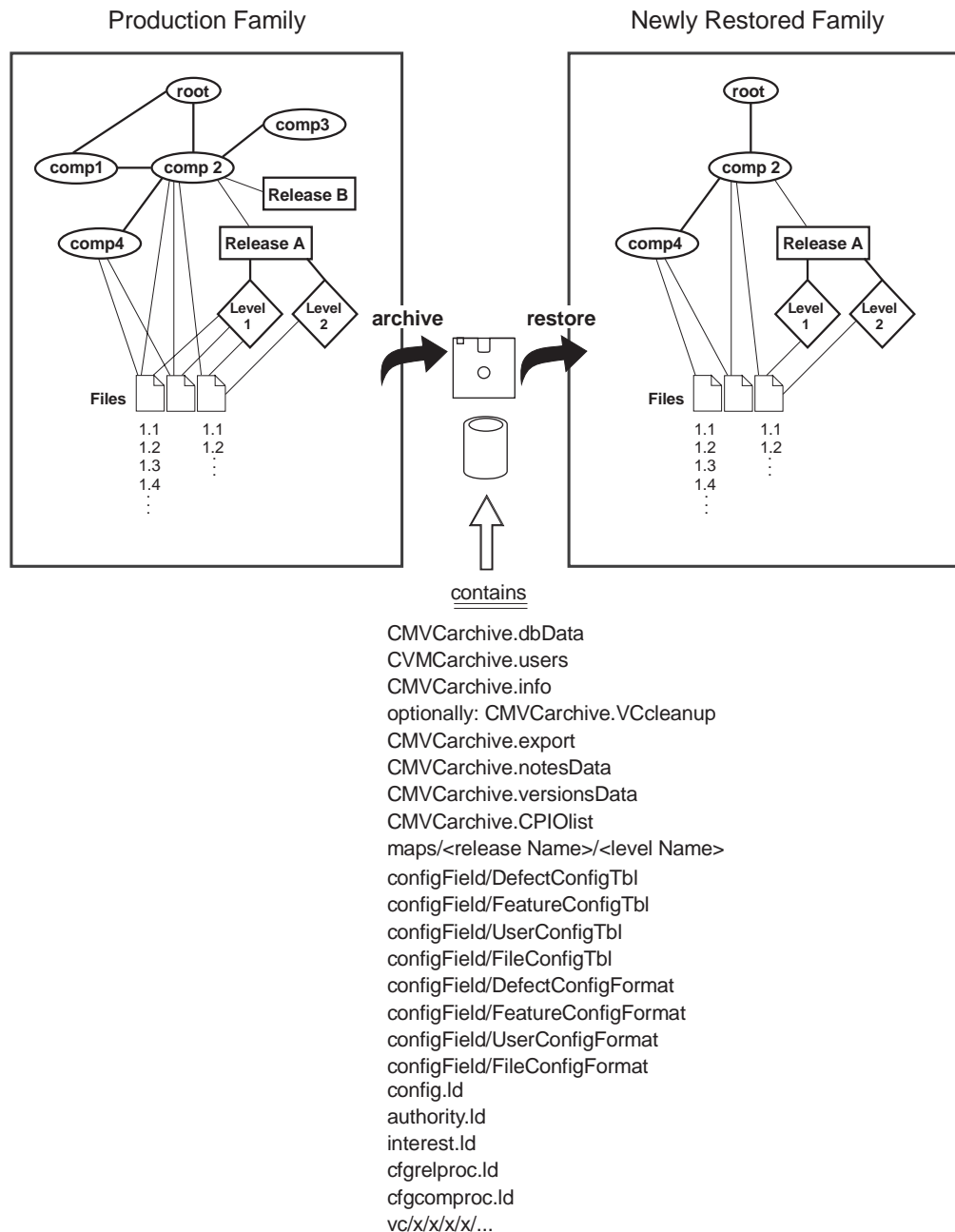


Figure 48. Example of a Family in Which Two Levels Have Been Archived

## Archive Functions

Use the archive function to:

- Estimate the space required to store the archived data.
- Archive all levels in a release up to and including a specified level.
- Archive one or more CMVC releases.
- Find all CMVC objects that are related to the selected archive object or objects and archive them.
- Allow the user to specify whether the archived objects are to be stored in a local empty file system, or stored on media in an attached and configured device, such as, tape or diskette.
- Allow the user to select whether the archived objects are to be removed from the CMVC family. Only objects that have no dependencies with remaining objects in the CMVC family will be removed.
- Create files to store information about the objects and data that were archived. See “Files Created by cmvcarchive” on page 148 for more information.

## Restore Functions

Use the restore function to:

- Restore the archived information into a new CMVC family for maintenance activities or for further development.
- Restore archived information that resides on a local file system, or on media loaded in an attached and configured device, such as, tape or diskette.
- Specify the name of the local file system or device.
- Reconfigure the family, using the configuration files and by running the **chfield** program as necessary.
- Populate the database tables used by the new CMVC family and restore the archived level maps, family configuration files and version control files.

---

## Archive and Restore Preparation

To archive objects, the status of the objects being archived must remain intact for the complete duration of the archive activity. The status of objects can be guaranteed in one of two ways:

- By revoking access to the CMVC development environment by stopping the **cmvcd** server daemons.
- By ensuring that users only perform CMVC report operations which query the CMVC server but do not update information.

If an archive operation is time-consuming, and there are other CMVC users who need to access information, administrators can provide users with read-only access to CMVC by stopping the **cmvcd** daemons and restarting them in the maintenance mode, so that CMVC will accept only reporting commands. See Chapter 13, “CMVC Server Daemons” for more information on running the **cmvcd** daemons in the maintenance mode. After the archive operation is completed, the regular **cmvcd** server daemons have to be restarted so that development can continue.

As mentioned in the previous section, two options can be selected when performing an archive of CMVC objects. Each of these options must meet certain prerequisites before the archive can proceed.

## Level Archive Prerequisites

The following prerequisites must be satisfied to archive all CMVC levels up to and including a specified level.

- The archive program can only be run against CMVC families that have migrated to Version 2 Release 1. That is, database tables for these families must be configured to the Version 2 Release 1 structures.
- The level to be archived must exist in the release specified.
- The specified level must be in the complete state.
- All other levels committed prior to the specified level's commit date must be in the complete state.
- All tracks that are level members of these levels must be in the complete state.
- All tracks that were committed without levels (that is, the process managing the release associated with the track did not include the level subprocess) must be in the complete state.
- All levels for this release that are not being archived must contain level members different to those that are being archived. That is, if the same level member is associated with two or more levels for the release, all of those levels must be included in the archive.
- The database that is being used by the family must be running.

## Objects Archived by the Level Archive Option

When a level archive is selected, the related objects to be archived, in addition to the level(s), include:

- The release associated with the levels
- The environment and approver lists associated with the release
- Level members associated with the levels
- Level maps associated with the levels
- Tracks associated with the levels, as well as any tracks that were committed without levels during the time the first level being archived was committed and the time the last level being archived was committed.
- Approval, fix, test and corequisite records associated with the tracks
- Change records associated with the tracks
- File records associated with the change records of the tracks, as well as files created or changed during any nontracking period between the time the first level being archived was committed and the time the last level being archived was committed
- Version records of the files being archived
- Version control files (SCCS and PVCS) associated with the files being archived
- Path records associated with the files being archived
- Components associated with any of the objects being archived



- Component member relationships for the components being archived
- Defect and feature records associated with the tracks being archived
- Defect and feature notes, histories, sizes and verification records for the defect and features being archived
- Users who own any of the objects to be archived
- Host list entries of those users
- Sequence database table entries (3 in total)
- Authority, Interest, Config, Cfgrelproc, and Cfgcomproc database LOAD files
- All configuration files for the CMVC family (for configurable fields).

## Release Archive Prerequisites

The following prerequisites must be satisfied to archive CMVC releases:

- The archive program can only be run against CMVC families that have migrated to Version 2 Release 1 or later. That is, data tables for these families must be configured to the structures of Version 2 Release 1 or later.
- The specified release or releases must exist.
- All levels associated with the release or releases must be in the complete state.
- All files associated with the release or releases must be checked in.
- All tracks associated with the release or releases must be in the complete state.
- All sizing records associated with the release or releases that are in the notReady, ready, or accept states must be deleted if the defect or feature is in any state except verify or closed.
- The database that is being used by the family must be running.

## Objects Archived by the Release Archive Option

When a release archive is selected, the related CMVC objects include:

- Environment and approver lists associated with the releases
- Levels and level members associated with the releases
- Level maps associated with the levels
- Tracks associated with the releases
- Approval, fix, test, and corequisite records associated with the tracks
- Change records associated with the tracks
- File records associated with the releases
- Version records of the files associated with the releases
- Version control files (SCCS or PVCS) associated with the files
- Path records associated with the files
- Components associated with any of the CMVC objects being archived
- Component member relationships for the components being archived
- Defect and feature records associated with the tracks

- Defect and feature notes, histories, sizes and verification records for the defect and features being archived
- Users who own any of the objects to be archived
- Host list entries of those users
- Sequence database table entries (3 in total)
- Authority, Interest, Config, Cfgrelproc, and Cfgcomproc database LOAD files
- All configuration files for the CMVC family (for configurable fields).

## Restore Prerequisites

The following prerequisites must be satisfied to restore archived CMVC objects:

- The restore program can only be run against CMVC families that have migrated to Version 2 Release 1 or later. That is, the new CMVC family must have been created with the **mkdb** and **mkfamily** scripts from Version 2 Release 1 or later.
- The CMVC family being restored to must be new.
- The new family must not be configured with additional fields for the Defect, Feature, User, and File CMVC objects. These objects assume the characteristics of the archived objects. Any attempt to create additional fields before running the restore program causes unpredictable results. Additional fields can be configured after the restore is completed successfully.
- The first superuser in the new family must not have the same CMVC user ID as any of the users who own any of the archived data that is to be restored. The file **CMVCarchive.users** contains a list of all the CMVC user IDs of users that are archived. Ensure that you select a CMVC superuser whose user ID is not in this list.
- Sufficient space must exist in the file system for the new family to hold the archived data.
- Sufficient space must exist in the database used by the family.
- The database that is being used by the family must be running.
- If your restore family uses PVCS as its underlying version control system, you must keep the vc file system for the family owned by the family user. Following the restore operation, you must change it to the PVCS owner, as detailed in “Using the mkfamily Command” on page 41.

## Archive and Restore Limitations

The limitations of the archive and restore functions include the following:

- After CMVC objects are archived and removed from a CMVC family, they cannot be restored into the original family. They must be restored into a new, empty, family.
- You cannot restore multiple archived packages of CMVC objects into the same CMVC family.
- If multiple releases are archived in one package, you cannot restore individual releases from that package. All releases in an archive package must be restored together.

- You cannot restore data from one family into another if they use different operating systems. For example, data archived from a CMVC family using AIX cannot be restored into a CMVC family using SunOS.
- You cannot restore data from one family into another if they use different database formats. For example, data archived from a CMVC family using the INFORMIX database cannot be restored into a CMVC family using SYBASE.
- You cannot restore data from one family into another if they use different version control systems. For example, data archived from a CMVC family using PVCS cannot be restored into a CMVC family using SCCS.
- The **cmvcarchive** program does not archive the Access and Notification tables for those components that are being archived.
- After data is restored, new defects or features opened in the restored family use the sequence numbers that were archived, if the user does not specify the defect or feature number. The sequence number that is used depends on the sequence number from the archived data.

For example, if the defect number starts at 1345, users might think that defects 1-1344 are lost. This is not the case. Some defects or features are restored from the archived data; others do not exist in the family.

---

## Archive and Restore Procedures

Before running the archive program, the family administrator should back up the CMVC family's account on the CMVC server, and export and back up the database tables used by the family. This backup ensures complete recovery if the wrong information is archived.

### Using the **cmvcarchive** Program

1. Stop all **cmvcd** daemons for the CMVC family.
2. Optionally, start the **cmvcd** daemons in maintenance mode.

For more information on the **cmvcd** daemons, see Chapter 13, "CMVC Server Daemons."

3. Enter **cmvcarchive** from the *familyname* account on the CMVC server.
4. Select one of the following options:

- a. Check archive prerequisites and estimate the storage required to perform an archive.

If you select this option, you are prompted to choose one of the following options:

- 1) Check prerequisites and estimate the storage for archiving levels of a release up to and including a specified level.
- 2) Check prerequisites and estimate the storage for archiving one or more releases.
- 3) Exit.

- b. Perform an archive.
- c. Exit.

5. If you choose to perform an archive, you will be prompted to select one of the following options:
  - a. Archive levels of a release up to and including a specified level.
  - b. Archive one or more releases.
  - c. Exit.
6. Depending on the option selected, the program prompts for:
  - a. Level and release names
  - b. Release names (for example, rel\_1.1 rel\_1.2)
7. If you choose to archive objects, you are prompted as to whether the archived objects should be deleted from the CMVC family after the archive.
 

Deleting the archived objects reclaims file system space on the CMVC server and database space used by CMVC. To improve database performance, a separate database-specific operation to reduce fragmentation must be run. This is a separate function that is not performed by CMVC.
8. If you choose to archive objects, you are asked where the archived objects are to be stored:
  - a. In a local file system
  - b. On media in an external device (for example, a tape device)
9. If you choose to store the objects in a local file system, you are prompted for the name of the file system.
 

If you choose to store the objects on media in an external device, you are prompted for the name of the device and the name of a directory that can be used as a temporary working directory.

When archiving to an external device, you may occasionally be prompted to remove the current media and insert additional media depending on the volume of data being archived. The archive program uses the **cpio** command to archive the data to external media. The **cpio** program prompts you to change or insert additional media when required.
10. After archiving is completed, stop the **cmvcd** maintenance-mode daemons if they were started earlier, and start the **cmvcd** daemons.

### Files Created by cmvcarchive

The **cmvcarchive** program creates several files that store information about the objects and data that is being archived. The files created are:

- **CMVCarchive.dbData**

A database command file that contains the database statements required to populate database tables upon restore.

- **CMVCarchive.export**

This file is created if your family uses the ORACLE database. It contains an export of the family's Notes and Versions tables.

- **CMVCarchive.info**

An archive information file that contains information about the contents of the archive.

- **CMVCarchive.notesData** and **CMVCarchive.versionsData**

These files are created if your family uses a DB2/6000, INFORMIX, or SYBASE database. They contain the archived notes and versions records, respectively, for your family.

- **CMVCarchive.PVCS\_clean**

A file created in the home directory of the archive family, if that family uses PVCS as its version control system. This file contains a list of version control files that can be manually removed to reclaim file system space after the archive is completed.

- **CMVCarchive.users**

A user information file containing the CMVC user IDs of the users that are archived. Upon restore, this file can be used to select a name for the CMVC superuser. The name cannot be one that is on this list.

- **CMVCarchive.VCcleanup**

A version control file that contains the list of deltas that must be removed for each version control file during restoration. This file is only created if you are archiving to a device.

- **CMVCarchive.CPIOlist**

A list of files copied to the device media. This file is only created if you are archiving to a device.

These files are created in the specified local file system or working directory. If the archive is being routed to a device, the files are deleted after the archive has completed, except the **CMVCarchive.PVCS\_clean** file.

## Using the **cmvcrestore** Program

1. Create the new CMVC family by running the **mkfamily** and **mkdb** shell scripts.  
**Note:** When running the **mkdb** command, you should not specify the **-d** option. Nor should you configure any additional fields for defect, feature, file, and user objects by means of the **chfield** program. Ensure that no files exist in the **\$HOME/configField** directory.
2. Verify that sufficient disk space exists in the CMVC family's home directory to store all the level maps and version control files that are being restored. Also verify that sufficient disk space exists in the database used by the CMVC family.
3. Stop all **cmvcd** daemons.
4. Optionally, start the **cmvcd** daemons in the maintenance mode.  
For more information on the **cmvcd** daemons, see Chapter 13, "CMVC Server Daemons."
5. Enter **cmvcrestore** from the *familyname* account on the CMVC server.
6. Select one of the following restore options:
  - a. Restore archived CMVC objects from a local file system.
  - b. Restore archived CMVC objects from media on an external device.
  - c. Exit.
7. Then, depending on the restore option selected, the program prompts for:
  - a. The name of the file system, for example, /home/<familyName>/archive
  - b. The name of the device, for example, /dev/rmt1 or /dev/rfd0

When restoring from an external device, you may occasionally be prompted to remove the current media and insert additional media depending on the volume of data being restored.

8. If you are using PVCS as your version control system, you should change:
  - The ownership (with the **chown** command) of the **\$HOME/vc** files to pvcs
  - The group (with the **chgrp** command) to system after cmvcrestore is completed.
9. Stop the **cmvcd** maintenance mode daemons, if they were started earlier.
10. Start the **cmvcd** daemons.
11. Inform the users of the new family of its name, the server machine name, and the port number for the family.

## Resetting Operating System Semaphores

The **cmvcarchive** and **cmvcrestore** programs use an operating system semaphore to indicate that they are currently running. There may be instances in which these programs are terminated and the programs have not been able to remove the semaphore. If you attempt to run either of these programs and you receive a message similar to either of the following and you verify that the programs are not running, you must remove the operating system semaphore before you can successfully run these programs:

```
"Another cmvcarchive program is currently running."
```

or

```
"Another cmvcrestore program is currently running."
```

To remove the semaphore, type the following command:

```
ipcs
```

Look for the semaphore held by the name of the CMVC family and then issue the following command to remove the semaphore:

```
ipcrm -s <semaphoreID>
```

## Appendix A. Error Messages and Recovery

The following list of error messages, generated by the CMVC server and shown on either the CMVC client or the CMVC server, can require the attention of the CMVC family administrator or the system administrator for resolution. Only a subset of the CMVC messages are discussed in this chapter. Messages generated by the clients are not discussed.

Some of these messages are routed to the console through the standard error pipe; others, such as, problems with system subroutines, are routed to the **syslog** file assuming the **syslog** daemon is running. The location of the **syslog** file depends on the operating system. Usual locations are:

Operating System	Location of syslog File
AIX	<b>/usr/spool</b>
HP-UX	<b>/usr/adm</b>
SunOS	<b>/var/log</b>
Solaris	<b>/var/adm/messages</b> <b>/var/log/syslog</b>

Contact your system administrator for the exact location of the file.

If the **syslog** daemon is not running, then the messages destined for the **syslog** file are displayed on the console through the standard error pipe.

Publications mentioned in the recovery procedures refer to books provided with your operating system.

If the problems cannot be resolved after following the recommended recovery procedures, contact your IBM Representative for additional support.

---

**0010-023 The extraction of level <levelName> associated with release <releaseName> failed. The internal map for the level is not valid or the level map file is not loaded on the CMVC server disk.**

**Contact the family administrator for assistance.**

**Explanation:** This error occurs on the CMVC server when an internal map which defines the level is empty or is not found.

**User Response:** Verify whether the map file exists on the CMVC server in the CMVC family's maps directory.

To find the map file for the level, check the CMVC family's maps directory for the directory that represents the release that the level is associated with. Within that directory, a map file should exist for each committed level and have the same name as that of the level name.

If the map does not exist, it may have been archived onto external backup media. If so, restore the map file from the backup media into the correct directory relative to the CMVC family's maps directory.

For example, the map file for the committed level, Prototype1, associated with release Prototype exists in the **\$HOME/maps/Prototype** directory with a file name of **Prototype1**.

---

**0010-025 An error occurred when the CMVC server software tried to extract files from the version control file repository.**

**Explanation:** The CMVC server software fails to retrieve the file from the CMVC family's version control directories using the SCCS or PVCS **get** command.

This message is displayed along with other messages that indicate the nature of the SCCS or PVCS problem and the name of the file which was being accessed at the time of the failure.

**User Response:** Refer to your operating system documentation for the SCCS recovery procedure and your PVCS documentation for the PVCS recovery procedure.

**Note:** If this message is displayed with the following PVCS message:

```
get: License notification: user ID not authorized
```

Refer to "Registering Users to the PVCS License Administration Database" on page 25 for more information.

---

**0010-044 An error occurred when the CMVC server software tried to check out or extract file <fileName> from the version control file repository.**

**To solve the problem, contact the family administrator.**

**Explanation:** The CMVC server software fails to retrieve the file from the CMVC family's version control directories using the SCCS or PVCS **get** command.

This message is displayed along with other messages that indicate the nature of the SCCS or PVCS problem and the name of the file which was being accessed at the time of the failure.

**User Response:** Refer to your operating system documentation for the SCCS recovery procedure and your PVCS documentation for the PVCS recovery procedure.

---

**0010-045 An error occurred when the CMVC server software tried to check in file <fileName> to the version control file repository.**

**To solve the problem, contact the family administrator.**

**Explanation:** The CMVC server software fails to check in the file to the CMVC family's version control directories using the SCCS **delta** command or the PVCS **put** command. This message is displayed along with other messages that indicate the nature of the SCCS or PVCS problem and the name of the file which was being accessed at the time of the failure.

**User Response:** Refer to your operating system documentation for the SCCS recovery procedure and your PVCS documentation for the PVCS recovery procedure.

---

**0010-046 An error occurred when the CMVC server software tried to unlock file <fileName> in the version control file repository.**

**To solve the problem, contact the family administrator.**

**Explanation:** The CMVC server software fails to unlock the file in the CMVC family's version control directories using the SCCS **unget** command or the PVCS **vcs** command.

This message is displayed along with other messages that indicate the nature of the SCCS or PVCS problem and the name of the file which was being accessed at the time of the failure.

**User Response:** Refer to your operating system documentation for the SCCS recovery procedure and your PVCS documentation for the PVCS recovery procedure.



---

**0010-059 The audit facility on the CMVC server cannot be initialized.**

**Contact the family administrator to verify the existence of the audit directory and its access permission.**

**Explanation:** The **audit** directory in the CMVC family's home directory does not exist or cannot be accessed by the CMVC server software.

**User Response:** Stop the CMVC server software **cmvcd** daemons if they are running.

Verify that the **audit** directory exists in the CMVC family's home directory on the CMVC server with access permission set to 750, the owner set to the CMVC family name, and the group set to system.

Restart the CMVC server software **cmvcd** daemons.

---

**0010-060 The user's host name and address cannot be resolved.**

**Verify that the user's host name and address are included in the CMVC server's 'etc/hosts' file or the data files of the name server. If the host name and address appear in either of these files, the network or name server may be experiencing a temporary problem. If the problem persists, contact the family administrator.**

**Explanation:** The CMVC server software does not recognize the user's host name.

**User Response:** Verify that the user's host name and TCP/IP network address exist in either the **/etc/hosts** file on the CMVC server or the name server's data files.

---

**0010-061 Database error, <errorNumber> (<errorMsg>), has occurred.**

**If the error is a result of an invalid column name, then check the syntax of the command. If you cannot resolve the problem, contact the family administrator.**

**Explanation:** The specified error occurs while accessing the database. This message is displayed on the CMVC server console or entered into the **syslog** file if the **syslog** daemon is running.

**User Response:** Refer to the administrator's guide of your database program for the appropriate action to recover from the database error. If using DB2, type the following command for more information:

```
db2 ? sqlxxxx
```

Where xxxx is the <errorNumber>.

If the error is due to an invalid column name, refer to the *IBM CMVC User's Reference* to determine the correct column name.

---

**0010-062 The database cannot be initialized.**

**To solve the database problem, contact the family administrator.**

**Explanation:** The CMVC server software **cmvcd** daemon cannot log on to the database. This problem occurs when:

- The database is not running.
- The number of CMVC server software **cmvcd** daemons requested exceeds the database licence agreement.

This error is displayed on the CMVC server console or entered into the **syslog** file if the **syslog** daemon is running.

**User Response:** Message 0010-061 is also displayed on the CMVC server console or entered into the **syslog** file. Take the corrective measures to resolve the database problem indicated in message 0010-061 and start the CMVC server software again.

---

**0010-063** Error, <errorMsg>, occurred when the CMVC server software attempted to process the <functionName>() function.

To solve the operating system problem, contact the system administrator or the family administrator.

**Explanation:** The operating system fails to complete the specified function on the CMVC server.

**User Response:** Verify that the system is configured properly for the specified function.

---

**0010-064** An error occurred when the CMVC server software tried to obtain memory to service your request. The CMVC server software cannot obtain the required memory.

To solve the problem, contact the family administrator.

**Explanation:** The CMVC server has insufficient memory to honor the CMVC command. This may be caused by memory-intensive commands, for example, very large reports, or other activities on the CMVC server.

**User Response:** Check for other applications on the CMVC server that may be consuming memory.

Check the amount of free space in the file systems; pay special attention to the file system containing the /tmp directory and the file system containing the CMVC family's directories. Clean up or expand these file systems if necessary.

---

**0010-065** The action you requested cannot be completed. The CMVC server software could not create a file to hold the results of the database query.

To correct the problem, contact the family administrator.

**Explanation:** A temporary file cannot be created on the CMVC server.

**User Response:** Message 0010-139 is displayed along with this message. Refer to message 0010-139 for more details.

---

**0010-112** A version control error may have caused a file to become corrupted. The previous version control files have been restored.

Contact the family administrator for assistance.

**Explanation:** The CMVC server database and the version control system may be out of synchronization. If you are using SCCS, the CMVC server restores the s.binary and p.binary files. If you are using PVCS, the CMVC server restores the history file.

**User Response:** Retry your last action or request your family administrator to investigate the CMVC family.

---

**0010-113** \*\*\* WARNING: A Version Control Error Has Occurred. \*\*\* The previous version control files have been restored.

Contact the family administrator for assistance.

**Explanation:** The CMVC server database and the version control system may be out of synchronization. If you are using SCCS, the CMVC server restores the s.binary and p.binary files. If you are using PVCS, the CMVC server restores the history file.

**User Response:** Retry your last action or request your family administrator to investigate the CMVC family.

---

**0010-114 This CMVC action is completed, but has not been successfully verified. \ The previous version control files have been restored.**

**Contact the family administrator for assistance.**

**Explanation:** The CMVC server database and the version control system may be out of synchronization. If you are using SCCS, the CMVC server restores the s.binary and p.binary files. If you are using PVCS, the CMVC server restores the history file.

**User Response:** Retry your last action or request your family administrator to investigate the CMVC family.

---

**0010-139 The temporary query file, <queryFileName>, cannot be opened. Your request cannot be completed.**

**Check the permissions of the file and the directory in which it resides. If problems persist, contact the family administrator.**

**Explanation:** The CMVC server software fails when processing the **open** system subroutine on the specified file. This problem occurs when:

- The CMVC server software does not have permission to create files in the directory indicated in the <queryFileName>.
- The file already exists in the directory indicated in the <queryFileName>.
- The CMVC server file system on which the directory resides is full; the directory is indicated in the <queryFileName>.
- There is a depletion of i-nodes on the CMVC server.

**User Response:** Verify that the CMVC server software has permission to access the directory indicated in the <queryFileName>. Clean up or expand the file system on which the directory resides if needed. Verify that the CMVC server has enough i-nodes configured.

Message 0010-350 is also displayed. Refer to message 0010-350 for more details.

---

**0010-140 An error occurred when the CMVC server software tried to create a temporary file with pathname, <tempFileName>, on the CMVC client.**

**Check that the permissions of the directory allow files to be created. Also check that the CMVC client has sufficient storage available for file creation.**

**If problems persist, contact the system administrator or the family administrator.**

**Explanation:** The CMVC server software cannot create a temporary file on the CMVC client. This problem occurs when:

- The CMVC server software does not have permission to create files in the directory indicated in the <tempFileName>.
- The file already exists in the directory indicated in the <tempFileName>.
- The CMVC client file system on which the directory resides is full; the directory is indicated in the <tempFileName>.
- There is a depletion of i-nodes on the CMVC client.

**User Response:** Verify that the CMVC server software has permission to access the directory indicated in the <tempFileName>.

Clean up or expand the file system on which the directory resides if necessary.

Verify that the CMVC client has enough i-nodes configured.

Message 0010-350 is also displayed on the CMVC client. Refer to message 0010-350 for more details.

---

**0010-141** An error occurred when the CMVC server software tried to save a backup copy of file <fileName> to <backupFileName> on the CMVC client.

**Check that the permissions of the file allow others to read it and that the permissions of the directory allow files to be created. Also check that the CMVC client has sufficient storage available.**

**If problems persist, contact the system administrator or the family administrator.**

**Explanation:** The CMVC server software cannot rename a file on the CMVC client in the current working directory. This problem occurs when:

- The CMVC server software does not have permission to rename files in the current working directory.
- The CMVC client file system on which the current working directory resides is full.
- There is a depletion of i-nodes on the CMVC client.

**User Response:** Verify that the CMVC server software has permission to access the current working directory.

Clean up or expand the file system on which the directory resides if necessary.

Verify that the CMVC client has enough i-nodes configured.

Message 0010-063 is also displayed. Refer to message 0010-063 for more details.

---

**0010-142** An error occurred when the CMVC server software tried to restore the name of file <tempFileName> to <fileName> on the CMVC client.

**The operating system could not complete the rename() subroutine. Contact the system administrator or the family administrator for assistance.**

**Explanation:** The CMVC server software cannot rename a file on the CMVC client in the current working directory. This problem occurs when:

- The CMVC server software does not have permission to rename files in the current working directory.
- The CMVC client file system on which the current working directory resides is full.
- There is a depletion of i-nodes on the CMVC client.

**User Response:** Verify that the CMVC server software has permission to access the current working directory.

Clean up or expand the file system on which the directory resides if necessary.

Verify that the CMVC client has enough i-nodes configured.

Message 0010-063 is also displayed. Refer to message 0010-063 for more details.

---

**0010-247** The host name, <hostName>, for the CMVC server cannot be resolved.

**Verify that the CMVC server's host name and address are included in the CMVC client's '/etc/hosts' file or the data files of the name server. If the host name and address appear in either of these files, the network or name server may be experiencing a temporary problem.**

**If the problem persists, contact the family administrator.**

**Explanation:** Unresolved host name.

**User Response:** Verify that the server's host name and TCP/IP network address are included in the CMVC client's **/etc/hosts** file or the data files of the name server. If the host name appears in either of these files, restart the CMVC server software daemons and retry the command.

---

**0010-248** The port number of the CMVC server for family <familyName> cannot be resolved.

Verify that the CMVC client's `/etc/services` file contains the port number of the CMVC server for this family.

**Explanation:** Unresolved port number.

**User Response:** Verify that the CMVC client's `/etc/services` file contains the port number dedicated to the CMVC family to be accessed. If the CMVC family name is not unique then fully define the CMVC family by specifying the family as `familyName@serverHostName@serverPortNumber` when setting the `CMVC_FAMILY` environment variable or when setting the family name in the graphical user interface.

---

**0010-249** The error, <errorMsg>, occurred when the CMVC client software tried to create a socket for communication with the CMVC server software. It is possible that the operating system ran out of file descriptor resources on the CMVC client machine.

To solve the problem, contact the system administrator or the family administrator.

**Explanation:** An error occurs while processing the `socket()` system function on the CMVC server.

**User Response:** The error message identifies the cause of the problem.

---

**0010-250** A connection cannot be established with family <familyName> at node <hostName> on port <portNumber>. The error is: <errorMsg>.

To solve the problem, contact the system administrator or the family administrator.

**Explanation:** An error occurs while processing the `connect()` system function on the CMVC server. The connection request has been rejected by the CMVC server software.

**User Response:** Verify that the connection information displayed in the message is correct and that the CMVC server software daemons are running.

If the error occurs frequently, the CMVC server software daemons may be overloaded by incoming requests. Increase the number of CMVC server software daemons to alleviate this problem.

---

**0010-252** The `cmvcd` daemon has rolled back the previous command that it was running and it is now restarting. Either a database error occurred and the CMVC server software `cmvcd` is attempting to recover, or the `cmvcd` received a `SIGHUP` signal.

**Explanation:** The CMVC server software `cmvcd` daemon encounters a database error or an interrupt signal, `SIGHUP`, and is attempting to recover. This message is displayed on the CMVC server console or entered into the `syslog` file if the `syslog` daemon is running.

**User Response:** None. The `cmvcd` daemon should recover on its own. More messages will be displayed if the recovery is unsuccessful.

---

**0010-253** The CMVC server software `notifyd` daemon has halted because it received a `SIGTERM` signal.

**Explanation:** The CMVC server software `notifyd` daemon receives the normal termination signal, `SIGTERM`, and has halted as a result.

This message is displayed on the CMVC server console or entered into the `syslog` file if the `syslog` daemon is running.

**User Response:** Restart the `notifyd` daemon if desired.

---

**0010-256** An error occurred when the CMVC server software tried to process function <functionName>. It is possible that the network or CMVC server is experiencing problems.

If the network is operational, contact the family administrator. If it is not, contact the system administrator.

**Explanation:** The CMVC client software and the CMVC server software may be incompatible. Or your database file system may be full.

This message is displayed on the CMVC client and on the CMVC server console, or entered into the **syslog** file if the **syslog** daemon is running on the CMVC server.

**User Response:** Verify that the version of the CMVC client software is the same as the version of the CMVC server software and that the port number designated to the CMVC family is unique on the network.

Check the amount of free space in the file systems; verify that your database file system has sufficient space for your CMVC server operations.

Message 0010-257 is also displayed on the CMVC client and on the console of the CMVC server, or entered in the **syslog** file if the **syslog** daemon is running.

---

**0010-257** The communication between the CMVC client, <clientName>, and the CMVC server has ended abnormally.

**Explanation:** The CMVC server software receives an unexpected end-of-file character while reading data from the network. There may be problems with intermittent noise on the network or incompatible CMVC client software or another application program is trying to connect with the CMVC server software.

This message is displayed on the CMVC client and on the CMVC server console or entered into the **syslog** file if the **syslog** daemon is running on the CMVC server.

**User Response:** Verify that the version of the CMVC client software is the same as the version of the CMVC server software and that the port number designated to the CMVC family is unique on the network.

---

**0010-283** The CMVC server software **cmvcd** daemon has halted on signal <signal>.

**Explanation:** The CMVC server software **cmvcd** daemon receives the specified signal and has halted as a result.

This message is displayed on the CMVC server console or entered into the **syslog** file if the **syslog** daemon is running.

**User Response:** Restart the **cmvcd** daemon if desired.

---

**0010-299** The CMVC server software could not create directory <directoryName> on the CMVC client's machine.

**Explanation:** The CMVC server software cannot create the directory on the client's machine. This problem occurs when:

- The CMVC server software does not have permission to create files in the parent directory of <directoryName>.
- The directory already exists on the CMVC client.
- The file system containing the parent directory is full.
- There is a depletion of i-nodes on the CMVC client.

**User Response:** Verify that the CMVC server software has permission to access and write in the parent directory.

Clean up or expand the file system on which the parent directory resides if necessary.

Verify that the CMVC client has enough i-nodes configured.

Message 0010-063 is also displayed. Refer to message 0010-063 for more details.

---

**0010-305** The type of file <fileName> associated with release <releaseName> cannot be determined.

**To solve the problem, contact the family administrator.**

**Explanation:** The CMVC server software fails while processing the **open** system subroutine on the specified file. This problem occurs when:

- The file's type is not text or binary.
- The CMVC server software does not have permission to read files in the directory indicated in the <fileName>.

**User Response:** Verify that the CMVC server software has permission to read the file indicated in the <fileName> and that the file is not damaged.

Message 0010-350 is also displayed on the console of the CMVC server or displayed in the **syslog** file on the CMVC server if the **syslog** daemon is running. Refer to message 0010-350 for more details.

---

**0010-322** The view specified is valid, but no report format entry exists for it.

**Explanation:** An internal CMVC server software problem occurs.

**User Response:** Call your IBM Representative to report this problem.

---

**0010-325** The CMVC server software cannot mount directory <directoryName> from node <hostName>. The CMVC server software requires mount and write access to the directory to perform a remote extraction.

**Check that the directory is exported at the CMVC client and has the appropriate access permissions. Also check that the Network File System (NFS) server is running.**

**Explanation:** The CMVC server software cannot mount the directory and therefore cannot complete the remote extraction. This problem occurs when:

- The directory is not exported at the client.
- The exported directory's permission does not allow the CMVC server software to write to it.
- The directory has been exported with limited access to certain hosts or netgroups which do not include the CMVC server.

**User Response:** Check that the directory on the client has been exported with the appropriate user and file access permission.

---

**0010-326** The CMVC server software cannot create directory <directoryName> relative to the destination extraction directory.

**Check that the parent directory permits the CMVC server software to write to it. Also check for the existence of the destination extraction directory and its access permissions. If the destination extraction directory exists, ensure that its permissions allow the CMVC server software to write to it.**

**Explanation:** The CMVC server software cannot create the directory on the client's machine. This problem occurs when:

- The CMVC server software does not have permission to create files in the parent directory of <directoryName>.
- The directory already exists on the CMVC client with permissions that prevent the CMVC server software from writing to it.
- The file system containing the parent directory is full.

- There is a depletion of i-nodes on the CMVC client.

**User Response:** Verify that the CMVC server software has permission to access and write in the parent directory.

Clean up or expand the file system on which the parent directory resides if necessary.

Verify that the CMVC client has enough i-nodes configured.

Message 0010-063 is also displayed. Refer to message 0010-063 for more details.

**0010-335 The RPC ID, <id>, is out of range. The remote procedure call interface between the CMVC client and CMVC server has failed.**

**To solve the problem, contact the family administrator.**

**Explanation:** The Remote Procedure Call ID cannot be identified by the CMVC server software. The CMVC client software may be incompatible with the CMVC server software or another application program may be sending commands to the CMVC socket.

This message is displayed on the CMVC server console or entered into the **syslog** file if the **syslog** daemon is running.

**User Response:** Verify that the version of the CMVC client software is the same as the version of the CMVC server software and that the port number designated to the CMVC family is unique on the network.

**0010-336 The packet type is not valid. Either there is a network problem or another application program is sending commands that are not valid to the CMVC socket.**

**To solve this problem, contact the family administrator.**

**Explanation:** The data received by the CMVC server software contains an invalid packet type. The CMVC client software may be incompatible with the CMVC server software or another application program may be sending commands to the CMVC socket.

This message is displayed on the CMVC server console or entered into the **syslog** file if the **syslog** daemon is running.

**User Response:** Verify that the version of the CMVC client software is the same as the version of the CMVC server software and that the port number designated to the CMVC family is unique on the network.

**0010-337 An error occurred when the CMVC server software tried to open a temporary file on the CMVC server prior to transferring a file from the CMVC client to the CMVC server.**

**Contact the family administrator for further problem resolution.**

**Explanation:** The CMVC server software fails when processing the **open** system subroutine on the specified file. This problem occurs when:

- The CMVC server software does not have permission to create files in the directory.
- The file already exists in the directory.
- The CMVC server file system on which the directory resides is full.
- There is a depletion of i-nodes on the CMVC server.

**User Response:** Verify that the CMVC server software has permission to access the directory.

Clean up or expand the file system on which the directory resides if necessary.

Verify that the CMVC server has enough i-nodes configured.

Message 0010-350 is also displayed on the console of the CMVC server or entered into the **syslog** file if the **syslog** daemon is running. Refer to message 0010-350 for more details.



---

**0010-339 The CMVC server software cannot unmount the directory <directoryName> which was mounted from node <hostName>.**

**Contact the family administrator to unmount the directory manually.**

**Explanation:** The CMVC server software fails to unmount the directory following file extraction. The specified directory is still mounted on the CMVC server from the specified host.

**User Response:** Unmount the directory manually.

Verify that the access permissions of the mounted directory are properly configured.

---

**0010-340 The CMVC server software cannot remove the directory <directoryName>. The extraction cleanup procedure cannot be completed.**

**Contact the family administrator to delete the directory manually.**

**Explanation:** The CMVC server software fails to remove the specified directory from the CMVC server following the file extraction. The access permission of the directory may have been changed during the extraction.

**User Response:** Remove the directory manually.

---

**0010-341 A mkdir() operating system subroutine failed when the CMVC server software tried to create a file in the version control file repository.**

**Contact the system administrator or the family administrator for further problem resolution.**

**Explanation:** The CMVC server software fails when processing the **mkdir** system subroutine. This problem occurs when:

- CMVC server software does not have permission to create files in the CMVC family's version control directories.
- The file system containing the CMVC family's version control directories is full.
- The directory already exists in the CMVC family's version control directories.
- There is a depletion of i-nodes on the CMVC server.

**User Response:** Verify that the CMVC server software has permission to access the version control directories of the CMVC family (those directories relative to **/u/familyName/vc**).

Clean up or expand the file system on which the directory resides if necessary.

Verify that the CMVC server has enough i-nodes configured.

Message 0010-063 is also displayed on the console of the CMVC server or entered into the **syslog** file if the **syslog** daemon is running. Refer to message 0010-063 for more details.

---

**0010-342 A symlink() operating system subroutine failed when the CMVC server software tried to symbolically link file <fileName> to file <fileName> in the version control file repository.**

**Contact the system administrator or the family administrator for further problem resolution.**

**Explanation:** The CMVC server software fails when processing the **symlink** system subroutine on the CMVC server. This problem occurs when:

- The target file to be linked to already exists.
- The CMVC server software does not have permission to write in the CMVC family's version control directories.
- The file system containing the CMVC family's version control directories is full.

**User Response:** Verify that the CMVC server software has permission to access the version control directories of the CMVC family (the vc directory and its subdirectories relative to the CMVC family home directory).

Clean up or expand the file system on which the directory resides if necessary.

**0010-343** A read() operating system subroutine failed when the CMVC server software tried to read file <fileName> prior to copying it to file <fileName> in the version control file repository.

**Contact the system administrator or the family administrator for further problem resolution.**

**Explanation:** The CMVC server software fails when processing the **read** system subroutine on the CMVC server. This problem occurs when the CMVC server software does not have permission to read the file in the CMVC family's version control directories.

**User Response:** Verify that your family's version control software has been configured correctly.

**0010-344** A write() operating system subroutine failed when the CMVC server software tried to copy file <fileName> to file <fileName> in the version control file repository.

**Contact the system administrator or the family administrator for further problem resolution.**

**Explanation:** The CMVC server software fails when processing the **write** system subroutine on the CMVC server. This problem occurs when the CMVC server software does not have permission to write to files in the CMVC family's version control directories.

**User Response:** Verify that your family's version control software has been configured correctly.

**0010-350** The error (<errorMsg>) occurred when the CMVC server software processed function <functionName>() on the file with path name <pathName>.

**Check that the path name, the file permissions, and the directory permissions are correct. If problems persist, contact the system administrator or the family administrator.**

**Explanation:** The CMVC server software fails to process the specified subroutine.

**User Response:** Check that the path name, the file permissions, and the directory permissions are correct.

**0010-351** An error occurred when the CMVC server software tried to open a temporary file on the CMVC server prior to transferring a file from the CMVC server to the CMVC client.

**Contact the family administrator for further problem resolution.**

**Explanation:** The CMVC server software fails when processing the **open** system subroutine on the specified file. This problem occurs when:

- The CMVC server software does not have permission to create files in the directory.
- The file already exists in the directory.
- The CMVC server file system on which the directory resides is full.
- There is a depletion of i-nodes on the CMVC server.

**User Response:** Verify that the CMVC server software has permission to access the directory.

Clean up or expand the file system on which the directory resides if necessary.

Verify that the CMVC server has enough i-nodes configured.

Message 0010-350 is also displayed on the console of the CMVC server or entered into the **syslog** file if the **syslog** daemon is running. Refer to message 0010-350 for more details.

---

**0010-352 An error occurred when the CMVC server software tried to open the file, <fileName>, on the CMVC client.**

**Check that the access permission of the file allow the CMVC server software to read it.**

**Explanation:** The CMVC server software cannot open the file on the CMVC client. This problem occurs when the CMVC server software does not have permission to read the specified file.

**User Response:** Verify that the CMVC server software has permission to access the file.

Message 0010-350 is also displayed on the console of the CMVC server or entered in the **syslog** file if the **syslog** daemon is running. Refer to message 0010-350 for more details.

---

**0010-354 An error occurred when the CMVC server software tried to open the temporary file, <tempFileName>, on the CMVC server during a version control check in operation.**

**Contact the family administrator for further problem resolution.**

**Explanation:** The CMVC server software cannot open the temporary file on the CMVC server. This problem occurs when the CMVC server software does not have permission to read the file indicated by the <tempFileName>.

**User Response:** Verify that the CMVC server software has permission to access the file indicated by the <tempFileName>.

Message 0010-350 is also displayed on the console of the CMVC server or entered in the **syslog** file if the **syslog** daemon is running. Refer to message 0010-350 for more details.

---

**0010-380 The chmod request on file <fileName> to a mode of <fileMode> failed during the extraction.**

**Explanation:** The CMVC server software fails when processing the **chmod** system subroutine on the specified file on the CMVC client. This problem occurs when the CMVC server software does not have permission to change the mode of the file.

**User Response:** Check that the path name, the file permissions, and the directory permissions are correct.

Message 0010-350 is also displayed on the CMVC client. Refer to message 0010-350 for more details.

---

**0010-381 The CMVC client software is not compatible with the CMVC server software for this action.**

**Contact the family administrator to upgrade the CMVC client software installation.**

**Explanation:** The CMVC server software receives an invalid request.

This message is displayed on the CMVC server console or entered into the **syslog** file if the **syslog** daemon is running.

**User Response:** Verify that the version of the CMVC client software is the same as the version of the CMVC server software.

---

**0010-475** The CMVC server software cannot obtain the required shared memory. The following error occurred: <errorNumber>

The CMVC server software will operate normally, but the CMVC Activity Monitor will not function.

**Explanation:** An error occurs when the CMVC server software processes the **shmget** system subroutine on the CMVC server. The amount of available physical memory may be insufficient to satisfy the request. This message is displayed on the CMVC server console or entered into the **syslog** file if the **syslog** daemon is running.

**User Response:** Refer to your operating system documentation for the error and recovery procedures.

---

**0010-476** The CMVC server software cannot remove the shared memory that it obtained. The following error occurred: <errorNumber>

Contact the family administrator to remove the shared memory manually.

**Explanation:** An error occurs when the CMVC server software processes the **shmctl** system subroutine on the CMVC server.

This message is displayed on the CMVC server console or entered into the **syslog** file if the **syslog** daemon is running.

**User Response:** Refer to your operating system documentation for the error and recovery procedures.

---

**0010-477** The CMVC server software cannot detach itself from shared memory. The following error occurred: <errorNumber>

**Explanation:** An error occurs when the CMVC server software processes the **shmdt** system subroutine on the CMVC server.

This message is displayed on the CMVC server console or entered into the **syslog** file if the **syslog** daemon is running.

**User Response:** Refer to your operating system documentation for the error and recovery procedures.

---

**0010-478** The CMVC server software cannot attach itself to shared memory. The following error occurred: <errorNumber>

The CMVC server software will operate normally, but the CMVC Activity Monitor will not function.

**Explanation:** An error occurs when the CMVC server software processes the **shmat** system subroutine on the CMVC server.

This message is displayed on the CMVC server console or entered into the **syslog** file if the **syslog** daemon is running.

**User Response:** Refer to your operating system documentation for the error and recovery procedures.

---

**0010-479** The CMVC server software cannot attach the parent process to shared memory. The following error occurred: <errorNumber>

**Explanation:** An error occurs when the CMVC server software processes the **shmat** system subroutine on the CMVC server.

This message is displayed on the CMVC server console or entered into the **syslog** file if the **syslog** daemon is running.

**User Response:** Refer to your operating system documentation for the error and recovery procedures.

---

**0010-489 The CMVC server software does not have write access to the directory <directoryName> on node <hostName>.**

**Check the permissions of the exported directory, make the necessary changes, and then retry your request.**

**If problems persist, contact the family administrator.**

**Explanation:** The CMVC server software does not have permission to write to the specified directory. The extraction cannot be completed. This problem occurs when:

- The exported directory's permission does not allow the CMVC server software to write to it.
- The directory has been exported with limited access to certain hosts or netgroups which do not include the CMVC server.

**User Response:** Check that the directory on the client has been exported with the appropriate user and file access permission.

---

**0010-490 The directory, <directoryName>, could not be created on the CMVC server.**

**Contact the family administrator to verify and set the CMVC server software's permission to the directory.**

**Explanation:** The CMVC server software cannot create the directory on the CMVC server. This problem occurs when:

- The CMVC server software does not have permission to create files in the parent directory of <directoryName>.
- The directory already exists on the CMVC client; its permission does not allow the CMVC server software to write to it.
- The file system containing the parent directory is full.
- There is a depletion of i-nodes on the CMVC server.

**User Response:** Verify that the CMVC server software has permission to access and write in the parent directory.

Clean up or expand the file system on which the parent directory resides if necessary.

Verify that the CMVC server has enough i-nodes configured.

Message 0010-063 is also displayed. Refer to message 0010-063 for more details.

---

**0010-491 The extraction failed because the CMVC server software could not create the directory, <directoryName>, to mount <sourceDirectory> from node <hostName>.**

**Contact the family administrator to verify the directory permissions.**

**Explanation:** The CMVC server software cannot create the directory on the CMVC server for mounting the directory from the remote host. This problem occurs when:

- The CMVC server software does not have permission to create files in the parent directory of <directoryName>.
- The directory already exists on the CMVC client; its permission does not allow the CMVC server software to write to it.
- The file system containing the parent directory is full.
- There is a depletion of i-nodes on the CMVC server.

**User Response:** Verify that the CMVC server software has permission to access and write in the parent directory.

Clean up or expand the file system on which the parent directory resides if necessary.

Verify that the CMVC server has enough i-nodes configured.

Message 0010-063 is also displayed. Refer to message 0010-063 for more details.

---

**0010-492 The unmount of directory <directoryName> failed.**

**Contact the family administrator to unmount the directory manually.**

**Explanation:** The CMVC server software fails to unmount the specified directory from the CMVC server. This message is displayed on the CMVC server console or entered into the **syslog** file if the **syslog** daemon is running.

**User Response:** Unmount the directory manually. Message 0010-063 is also displayed. Refer to 0010-063 for more details.

---

**0010-494 The CMVC server software cmvcd daemon is restarting.**

**Explanation:** An error occurs and causes the CMVC server software **cmvcd** daemons to restart.

This message is displayed on the CMVC server console or entered into the **syslog** file if the **syslog** daemon is running.

**User Response:** Additional error messages will be displayed to indicate the reasons for restarting.

---

**0010-495 The CMVC server software cannot obtain the control structure for shared memory. The following error occurred: <errorNumber>**

**The CMVC server software will operate normally, but the CMVC Activity Monitor will not function.**

**Explanation:** An error occurs when the CMVC server software processes the **shmctl** system subroutine on the CMVC server.

This message is displayed on the CMVC server console or entered into the **syslog** file if the **syslog** daemon is running.

**User Response:** Refer to your operating system documentation for the error and recovery procedures.

---

**0010-510 File <fileName> could not be retrieved. A temporary file could not be created because the CMVC server software does not have write access.**

**Contact the family administrator for assistance.**

**Explanation:** The CMVC server software fails to process the **open** system subroutine on the specified file on the CMVC server. This problem occurs when:

- The CMVC server software does not have permission to open the file in read-write mode and create the file if it does not already exist.
- The file already exists on the CMVC server; its permission does not allow the CMVC server software to read from it.
- The file system containing the parent directory is full.
- There is a depletion of i-nodes on the CMVC server.

**User Response:** Verify that the CMVC server software has permission to access the directory indicated in the <fileName>.

Clean up or expand the file system on which the directory resides if necessary.

Verify that the CMVC server has enough i-nodes configured.

Message 0010-350 is also displayed on the console of the CMVC server or entered in the **syslog** file if the **syslog** daemon is running. Refer to message 0010-350 for more details.

---

**0010-511 An error occurred when the CMVC server software tried to get file <fileName>.**

**Contact the family administrator for assistance.**

**Explanation:** The CMVC server software fails to retrieve the file from the CMVC family's version control directories using the SCCS or PVCS **get** command.

This message is displayed along with other messages that indicate the nature of the SCCS or PVCS problem and the name of the file which was accessed at the time of the failure.

**User Response:** Refer to your operating system documentation for the SCCS recovery procedure and your PVCS documentation for the PVCS recovery procedure.

**Note:** If this message is displayed with the following PVCS message:

```
get: License notification: user ID not authorized
```

Refer to "Registering Users to the PVCS License Administration Database" on page 25 for more information.

---

**0010-521 0010-521 The User Exit program, <programName>, was not found.**

**Explanation:** The CMVC server software fails to locate the specified User Exit program.

**User Response:** Check your \$home/bin directory to see if the specified program is present. Contact your system administrator if it is not.

Check your path to see if \$home/bin is included. Include this in your login profile and run the profile. Stop and start **cmvcd** again for the additional path to be recognized.

---

**0010-539 A packet error occurred. The SOH was not valid.**

**To solve the problem, contact the family administrator.**

**Explanation:** The client and server communicate through packets of data. Each packet should start with a unique header, the SOH. The CMVC server software receives a packet with an invalid header. Another application may be sending data to the same TCP/IP port number that is assigned to the CMVC family.

This message is displayed on the CMVC server console or entered into the **syslog** file if the **syslog** daemon is running.

**User Response:** Ensure that the TCP/IP port number assigned to the CMVC family is unique on the network.

---

**0010-540 A packet error occurred. The header was not valid.**

**To solve the problem, contact the family administrator.**

**Explanation:** The CMVC server software receives a packet of data that has an invalid header. Another application may be sending data to the same TCP/IP port number that is assigned to the CMVC family.

This message is displayed on the CMVC server console or entered into the **syslog** file if the **syslog** daemon is running.

**User Response:** Ensure that the TCP/IP port number assigned to the CMVC family is unique on the network.

---

**0010-541 A packet error occurred. The expected block type was not found.**

**To solve the problem, contact the family administrator.**

**Explanation:** The CMVC server software expects to receive a packet of data recognized as a **\*.block** type. It receives something else instead. Another application may be sending data to the same TCP/IP port number that is assigned to the CMVC family.

This message is displayed on the CMVC server console or entered into the **syslog** file if the **syslog** daemon is running.

**User Response:** Ensure that the TCP/IP port number assigned to the CMVC family is unique on the network.

---

**0010-542 A packet error occurred. The expected string type was not found.**

**To solve the problem, contact the family administrator.**

**Explanation:** The CMVC server software expects to receive a packet of data recognized as a **string** type. It receives something else instead. Another application may be sending data to the same TCP/IP port number that is assigned to the CMVC family.

This message is displayed on the CMVC server console or entered into the **syslog** file if the **syslog** daemon is running.

**User Response:** Ensure that the TCP/IP port number assigned to the CMVC family is unique on the network.

---

**0010-543 A packet error occurred. The expected call data type was not found.**

**To solve the problem, contact the family administrator.**

**Explanation:** The CMVC server software expects to receive a packet of data recognized as a **call data** type. It receives something else instead. Another application may be sending data to the same TCP/IP port number that is assigned to the CMVC family.

This message is displayed on the CMVC server console or entered into the **syslog** file if the **syslog** daemon is running.

**User Response:** Ensure that the TCP/IP port number assigned to the CMVC family is unique on the network.

---

**0010-544 A packet error occurred. The expected long type was not found.**

**To solve the problem, contact the family administrator.**

**Explanation:** The CMVC server software expects to receive a packet of data recognized as a **long** type. It receives something else instead. Another application may be sending data to the same TCP/IP port number that is assigned to the CMVC family.

This message is displayed on the CMVC server console or entered into the **syslog** file if the **syslog** daemon is running.

**User Response:** Ensure that the TCP/IP port number assigned to the CMVC family is unique on the network.

---

**0010-581 An error occurred when the CMVC server software tried to open the file <fileName> on the CMVC client during file migration.**

**Check that the access permissions of the file allow the CMVC server software to read it.**

**Explanation:** The CMVC server software fails to execute the **open** system subroutine on the specified file on the CMVC client.

This problem occurs when the CMVC server software does not have permission to open the file in read-only mode.



**User Response:** Verify that the CMVC server software has permission to access the specified file on the CMVC client.

Message 0010-350 is also displayed. Refer to message 0010-350 for more details.

**0010-583 The file migration cannot be completed. The `sclean` command, `<command>`, failed.**

**To solve the problem, contact the family administrator.**

**Explanation:** The CMVC server software is unable to successfully run the `sclean` command to prepare the SCCS input file during premigration processing.

**User Response:** Ensure that the `sclean` shell script exists in the `/usr/lpp/cmvc/bin` directory of the CMVC server with executable permission.

**0010-585 The file migration cannot be completed. The SCCS file, `<fileName>` cannot be cleaned up appropriately for migration.**

**To solve the problem, contact the family administrator.**

**Explanation:** The `sclean` shell script returns an error when preparing the specified SCCS input file for migration into CMVC.

**User Response:** Verify that the input file is a valid SCCS file and that it is not damaged in any way.

**0010-587 A packet error occurred. The expected text type was not found.**

**To solve the problem, contact the family administrator.**

**Explanation:** The CMVC server software expects to receive a packet of data recognized as a `text` type. It receives something else instead. Another application may be sending data to the same TCP/IP port number that is assigned to the CMVC family.

This message is displayed on the CMVC server console or entered into the `syslog` file if the `syslog` daemon is running.

**User Response:** Ensure that the TCP/IP port number assigned to the CMVC family is unique on the network.

**0010-588 The CMVC server software cannot continue. It cannot generate new sequence numbers.**

**Contact the family administrator for assistance.**

**Explanation:** The CMVC server software has used all of the available sequence IDs with which it identifies CMVC objects.

**User Response:** Create a new CMVC family and restore the current release to the newly created family. Extract the full level or release tree and create these files in the new CMVC family.

**0010-591 The CMVC Activity Monitor cannot obtain the required shared memory. The following error occurred: `<errorMsg>`**

**Explanation:** An error occurs when the CMVC Activity Monitor processes the `shmget` system subroutine on the CMVC server. The amount of available physical memory may not be sufficient to satisfy the request.

**User Response:** Refer to your operating system documentation for the error and recovery procedures.

---

**0010-592 The CMVC Activity Monitor cannot attach itself to shared memory. The following error occurred: <errorMsg>**

**Explanation:** An error occurs when the CMVC Activity Monitor processes the **shmat** system subroutine on the CMVC server.

**User Response:** Refer to your operating system documentation for the error and recovery procedures.

---

**0010-593 The CMVC Activity Monitor cannot remove the shared memory that it obtained. The following error occurred: <errorMsg>**

**Explanation:** An error occurs when the CMVC Activity Monitor processes the **shmctl** system subroutine on the CMVC server.

**User Response:** Refer to your operating system documentation for the error and recovery procedures.

---

**0010-604 File <fileName> could not be retrieved. The CMVC server software can not create this file on the CMVC client, <hostName>.**

**Check that the CMVC server software has permission to write to the destination directory on the CMVC client, <clientName>. If the file already exists in the destination directory, make sure that it can be overwritten by the CMVC server software.**

**Explanation:** The CMVC server software fails to process the **open** system subroutine on the specified file on the CMVC client. This problem occurs when:

- The CMVC server software does not have permission to open the file in read-write mode, and create the file if it does not already exist on the CMVC client.
- The file already exists on the CMVC client; its permission does not allow the CMVC server software to read from it.
- The file system containing the parent directory is full.
- There is a depletion of i-nodes on the CMVC client.

**User Response:** Verify that the CMVC server software has permission to access the directory indicated in the <fileName>.

Clean up or expand the file system on which the directory resides if necessary.

Verify that the CMVC client has enough i-nodes configured.

Message 0010-350 is also displayed on the console of the CMVC server or entered in the **syslog** file if the **syslog** daemon is running. Refer to message 0010-350 for more details.

---

**0010-606 The file system has run out of free blocks while attempting to receive file, <fileName>, mounted on the CMVC host, <hostName>.**

**Contact the family administrator for assistance.**

**Explanation:** The CMVC server software fails to process the **open** system subroutine on the specified file on the specified host. This problem occurs when:

- The file system containing the parent directory is full. As a result, the CMVC server software does not have permission to open the file in read-write mode and create the file if it does not already exist.
- There is a depletion of i-nodes on the specified host.

**User Response:** Clean up or expand the file system on which the parent directory resides.

Verify that the specified host has enough i-nodes configured.

---

**0010-607** The **stats** command failed with error code, <errorNumber>, while accessing the file, <fileName> mounted on the CMVC host, <hostName>.

**Contact the family administrator for assistance.**

**Explanation:** The CMVC server software fails to process the **stats** system subroutine on the specified file on the specified host. This message is displayed on the CMVC server console, or entered into the **syslog** file if the **syslog** daemon is running.

**User Response:** Refer to your operating system documentation for the error and recovery procedures.

---

**0010-623** The function <functionName> called by function <functionName> failed.

**Explanation:** This message is displayed along with various other messages when the communication between the client and the server fails. The other messages indicate the reason for the failure; this message is to help you trace the problem.

This message is displayed on the CMVC server console, or entered into the **syslog** file if the **syslog** daemon is running.

**User Response:** Refer to the other messages that are displayed for more information.

---

**0010-634** An error was detected when writing the map of level <levelName>, associated with release <releaseName>.

**Contact the family administrator for assistance.**

**Explanation:** The CMVC server software fails to write the map for the associated level during a level commit action. This problem occurs when:

- The CMVC server software does not have permission to create the directory for the map file relative to the **\$HOME/maps** directory on the CMVC server (where **\$HOME** is the home directory of the CMVC family).
- The CMVC server software does not have permission to open the map file in read-write mode.
- The file system containing the **\$HOME/maps** directory on the CMVC server is full (where **\$HOME** is the home directory of the CMVC family).
- There is a depletion of i-nodes on the CMVC server.

**User Response:** Verify that the CMVC server software has permission to create directories relative to the CMVC family's **\$HOME/maps** directory and that the file system containing this directory has sufficient free space available for map file creation. Also verify that the CMVC server has enough i-nodes configured.

---

**0010-636** Database error, <errorNumber> (<errorMsg>), has occurred.  
ISAM=<errorNumber>

**If the error is a result of an invalid column name, then check the syntax of the command. If you cannot resolve the problem, contact the family administrator.**

**Explanation:** The specified error occurs while accessing the database. This message is displayed on the CMVC server console or entered into the **syslog** file if the **syslog** daemon is running.

**User Response:** Refer to the administrator's guide of your database program for the appropriate action to recover from the database error. If the error is due to an invalid column name, refer to the *IBM CMVC User's Reference* to determine the correct column name.

---

**0010-637 The attribute flag <attribute> is not valid because release <releaseName> is not under tracking control.**

**Explanation:** You are using a flag that is valid only when the release is under tracking control.

**User Response:** If you want the release under tracking control, modify the process of the release and specify one of the options that use tracking control.

---

**0010-638 The version control type, <vcType>, is not valid.**

**To solve the problem, contact the family administrator.**

**Explanation:** The CMVC\_VCTYPE environment variable is either not set or is set to an incorrect value.

**User Response:** Verify that the CMVC family's account on the CMVC server has the environment variable, CMVC\_VCTYPE, set to `sccs`, `SCCS`, `pvcs` or `PVCS` depending on the version control system being used by the CMVC family.

---

**0010-651 A database error occurred and the database rollback failed in recovering from this problem. The server has exited because of this error.**

**To solve the database problem, contact the family administrator.**

**Explanation:** The database on the CMVC server attempted to undo the most recent transactions but was unsuccessful in doing so. The CMVC server software daemons have been stopped.

**User Response:** Restart the database and validate the integrity of the data. Restart the CMVC daemons once the database integrity has been validated. Refer to the administrator's guide of your database program for the appropriate actions to take to validate the integrity of the database contents.

---

**0010-652 The CMVC action has been aborted because the CMVC server is unable to write to the file system <fileSystemName>.**

**Please notify your system administrator.**

**Explanation:** The CMVC server software fails to write to the file system indicated in the message.

**User Response:** Verify that the CMVC server software has permission to write to the file system indicated in the message. Also verify that this file system is not full. Clean up or expand the file system if necessary.

---

**0010-653 The CMVC action has been aborted because the CMVC server file system, <fileSystemName> has run out of free space.**

**Please notify your system administrator.**

**Explanation:** The file system indicated in the message is full. The CMVC server software can not complete the action it was working on.

**User Response:** Clean up or expand the file system on the CMVC server.

---

**0010-676 The CMVC action was aborted because a problem occurred when trying to obtain the file size.**

**Please notify your system administrator.**

**Explanation:** The CMVC server software can not determine the size of the file when creating a file in the CMVC environment, when checking in a file or when migrating an SCCS file into the CMVC environment.

**User Response:** Verify that the file being acted upon is not corrupted and reissue the command.

---

**0010-677** The CMVC action was aborted while attempting to extract file <fileName>. A file system mounted from client <clientName> has insufficient free space.

**Explanation:** The file system indicated in the message is full. The CMVC server software can not complete the extraction of the specified file.

**User Response:** Clean up or expand the file system that is mounted from the CMVC client.

---

**0010-678** The CMVC action was aborted because the CMVC server could not access the extracted file <fileName>.

**Explanation:** The CMVC server software could not access the file indicated in the message. The extraction can not complete.

**User Response:** Verify that the destination extraction directory's permissions allow the CMVC server software to access the files within the directory. Verify that the extracted file is not corrupted in any way.

---

**0010-682** An error occurred in <vcType> when the CMVC server software tried to check out file <fileName>.

The version control failed to create a lock for this file within the version control file tree.

**Explanation:** The version control system failed to create a lock on the file within the version control file tree. The checkout action can not be completed.

**User Response:** Verify that the file system containing the version control file tree is not full. If this file system is full, expand it so that the version control system has sufficient space to create the required files that monitor file locks. Check the CMVC server's console or the **syslog** for error messages indicating that the version control system is not operating successfully and take corrective action to resolve any problems. Retry the checkout request once corrective actions have been completed.

---

**0010-683** An error occurred in <vcType> when the CMVC server software tried to check in file <fileName>.

The version control failed to delete the lock for this file within the version control file tree after the corresponding filesOut record was deleted from the FilesOut table.

**Explanation:** The version control system failed to delete the lock on the file within the version control file tree. The checkin action can not be completed. The file has been relocked and the file changes have not been checked in.

**User Response:** Verify that the file system containing the version control file tree is not full. If this file system is full, expand it so that the version control system has sufficient space to make updates to files. Check the CMVC server's console or the **syslog** for error messages indicating that the version control system is not operating successfully and take corrective action to resolve any problems. Retry the checkin request once corrective actions have been completed.

---

**0010-684** An error occurred when the CMVC server software tried to check out file <fileName>.

The CMVC server software failed to create a filesOut record for this file in the FilesOut table after the file was successfully locked in the version control file tree.

The file has been unlocked in the version control file tree. If problems persist, contact the family administrator.

**Explanation:** The CMVC server software fails to create an entry in the FilesOut table in the database after the file is successfully checked out from the version control system. The

checkout action can not be completed. The file has been unlocked in the version control file tree.

**User Response:** Verify that the database is operating successfully and has sufficient resources available to add records to its' tables. Check the CMVC server's console or the **syslog** for error messages indicating that the database is not operating successfully and take corrective action to resolve any problems. Retry the checkout request once corrective actions have been completed.

**0010-685 An error occurred when the CMVC server software tried to check in file <fileName>.**

**The CMVC server software failed to delete the filesOut record for this file from the FilesOut table after the file was successfully unlocked in the version control file tree.**

**The file has been relocked in the version control file tree. If problems persist, contact the family administrator.**

**Explanation:** The CMVC server software fails to delete an entry in the FilesOut table in the database after the file is successfully checked in to the version control system. The checkin action can not be completed. The file has been relocked in the version control file tree.

**User Response:** Verify that the database is operating successfully and has sufficient resources available to update records within its' tables. Check the CMVC server's console or the **syslog** for error messages indicating that the database is not operating successfully and take corrective action to resolve any problems. Retry the checkin request once corrective actions have been completed.

**0010-777 The CMVC client was attempting to transmit the file: <fileName> to the server and failed.**

**Explanation:** The CMVC server software failed to open the file when preparing to transfer the file from the CMVC client to the CMVC server. The CMVC action can not complete.

**User Response:** Check the permissions of the file and the directory that the file resides in on the CMVC client.

**0010-778 The CMVC client was unable to obtain the file size of file: <fileName>.**

**Explanation:** The CMVC server software failed to determine the size of the file when preparing to transfer the file from the CMVC client to the CMVC server. The CMVC action can not complete.

**User Response:** Verify that the file on the CMVC client is not corrupted in any way.

**0010-781 The CMVC server has determined there is inadequate file space to receive a file of size <fileSize> bytes. Please notify your system administrator.**

**Explanation:** The CMVC action can not be completed due to insufficient space within the CMVC server's file systems.

**User Response:** Expand the file system containing the version control file tree and cleanup or expand the file system containing the **/tmp** directory.

**0010-782 The CMVC server has detected an error in receiving a file from the client.**

**The expected size was <fileSize> bytes, but the size actually received was <fileSize> bytes.**

**Explanation:** The CMVC action can not be completed due to a discrepancy between the size of the file expected from the CMVC client and the size of the file received from the client.

**User Response:** Verify the contents of the file on the CMVC client and reissue the request.

---

**0010-783** A version control error may have caused a file to become corrupt. Should the version control file <filename> need to be restored, a recovery file has been saved on the CMVC server with the file name <fileName>.

**Please save this message and notify your family administrator.**

**Explanation:** A version control operation may have partially but not fully completed and as a result the file within the version control system may be corrupted.

**User Response:** Verify the contents of the version control file. If the file is corrupted, replace it with the recovery file. The recovery file represents the version control file prior to the version control operation.

---

**0010-784** \*\*\* WARNING: A Version Control Error Has Occurred .\*\*\* Should the version control file <filename> need to be restored, a recovery file has been saved on the CMVC server with the file name <fileName>.

**Please save this message and notify your family administrator.**

**Explanation:** A version control operation has failed and the version control file being operated on may be corrupted.

**User Response:** Verify the contents of the version control file. If the file is corrupted, replace it with the recovery file. The recovery file represents the version control file prior to the version control operation.

---

**0010-785** This CMVC action completed, but was not successfully verified. Should the version control file <fileName> need to be restored, a recovery file has been saved on the CMVC server with the file name <fileName>.

**Please save this message and notify your family administrator.**

**Explanation:** The CMVC server software was not able to verify the contents of the version control file after completing the version control operation.

**User Response:** Verify the contents of the version control file. If the file is corrupted, replace it with the recovery file. The recovery file represents the version control file prior to the version control operation.

---

**0010-786** The CMVC server has determined there is inadequate file space in the server's directory: <directoryName> to initiate this action.

**Explanation:** There is not enough space in the specified directory on the CMVC server. The version control operation has not been initiated as a result. The CMVC file operation can not complete.

**User Response:** Clean up or expand the file system that contains the specified directory on the CMVC server and then retry the CMVC file operation.

---

**0010-787** The CMVC server has determined there is inadequate file space in the server's directory <directoryName>. The version control command <vcCommand> was not initiated.

**Explanation:** There is not enough space in the specified directory on the CMVC server. The version control operation has not been initiated as a result. The CMVC file operation can not complete.

**User Response:** Clean up or expand the file system that contains the specified directory on the CMVC server and then retry the CMVC file operation.

---

**0010-788 The CMVC server could not create the version control recovery file.**

**Contact the family administrator for assistance.**

**Explanation:** The CMVC server software could not copy the original version control file to another file for recovery purposes. The version control operation has not been performed. The CMVC file operation can not complete.

**User Response:** Clean up or expand the file system that contains the **/tmp** directory on the CMVC server. Also check that the file system that contains the version control directories has sufficient space to expand.

---

**0010-789 The CMVC server could not create the version control recovery comparison file.**

**Contact the family administrator for assistance.**

**Explanation:** The CMVC server software could not obtain the file from the version control system in order to verify the contents of the file.

**User Response:** Verify the contents of the file in the version control file system.

---

**0010-790 The CMVC server could not create a file necessary for error detection in the directory <directoryName>.**

**Contact the family administrator for assistance.**

**Explanation:** The CMVC server software could not create the error detection file. The CMVC file operation can not complete.

**User Response:** Verify that the CMVC server software has permission to write to the specified directory and that the directory has sufficient space for file creation.

---

**0010-792 \*\*\* WARNING: A Version Control Error Has Occurred.\*\*\* The version control file <fileName> should be examined.**

**Please save this message and notify your family administrator.**

**Explanation:** A version control operation has failed and the version control file being operated on may be corrupted.

**User Response:** Verify the contents of the version control file. If the file is corrupted, replace it with the recovery file. The recovery file represents the version control file prior to the version control operation.

---

**0010-793 The CMVC action completed, but was not successfully verified. The version control file <fileName> should be examined.**

**Please save this message and notify your family administrator.**

**Explanation:** The CMVC server software was not able to verify the contents of the version control file after completing the version control operation.

**User Response:** Verify the contents of the version control file. If the file is corrupted, replace it with the recovery file. The recovery file represents the version control file prior to the version control operation.

---

**0010-794 The user exit program <programName> could not be executed. Ask your family administrator to check to ensure that the PATH environment variable in the CMVC family profile includes \$HOME/bin.**

**Explanation:** The CMVC server software was not able to execute the user exit program specified in the message.

**User Response:** Verify that the user exit program is in the \$HOME/bin directory, that the PATH environment variable on the CMVC server includes \$HOME/bin, and that the user exit program has execute permissions set.



---

**0010-813** The CMVC server software `cmvcd` daemon cannot be started from root or the family name specified is different from the current login user name.

You must be logged in to the CMVC family account to start the `cmvcd` daemon and specify the same family name for the CMVC family name parameter.

**Explanation:** You must be logged in to the CMVC family account to start the `cmvcd` daemon.

**User Response:** Log in to your family's account and start CMVC again.

---

**0010-871** The requested action, `<actionName>`, cannot be completed. The CMVC server software is in maintenance mode.

Contact the family administrator.

**Explanation:** When running in maintenance mode, the CMVC server software allows some read-only actions to be performed.

**User Response:** Stop the maintenance mode CMVC server daemons and start CMVC in nonmaintenance mode.

---

**0010-876** The Archive/Restore program is running. The `cmvcd` daemons cannot be started in nonmaintenance mode.

**Explanation:** CMVC can be run only in maintenance mode during an archive or restore.

**User Response:** Complete the archive or restore and start CMVC in nonmaintenance mode.

---

**0010-877** Cannot connect to NetLS server.

To solve the problem, contact the CMVC family administrator.

**Explanation:** A problem has occurred with the NetLS software that controls the number of users who are allowed to access CMVC.

**User Response:** Check that the NetLS software is configured correctly. Refer to your NetLS books for more information.

---

**0010-878** Cannot obtain a NetLS floating license.

To solve the problem, contact the CMVC family administrator.

**Explanation:** A problem has occurred with the NetLS software that controls the number of users who are allowed to access CMVC.

**User Response:** Check that the NetLS software is configured correctly and that there are enough NetLS tokens to handle the number of users who are trying to use CMVC. Refer to your NetLS books for more information.

---

**0011-037** The environment variable `DB2_HOME` is not set.

**Explanation:** You have not added the `DB2_HOME` environment variable to your login profile. This environment variable refers to the home directory of the DB2/6000 instance that the CMVC family will use.

**User Response:** Add the `DB2_HOME` environment variable to your login profile. Run your login profile to update your environment. Refer to "For a DB2/6000 Server" on page 36 for more information.

---

---

**0011-038 <Database program name> is not executable. Is the DB2\_HOME environment variable set correctly?**

**Explanation:** The DB2\_HOME environment variable in your login profile is not set to the home directory of the DB2/6000 instance that the CMVC family uses.

**User Response:** Check the setting of the DB2\_HOME environment variable in your login profile. Run your login profile to update your environment. Refer to “For a DB2/6000 Server” on page 36 for more information.

---

**0011-040 The length of the CMVC DB2 family's AIX user login password cannot be more than 18 characters.**

**Explanation:** You have tried to set the DB2\_PASS environment variable with more than 18 characters, which is the maximum allowed. The DB2\_PASS environment variable is set with the password of your CMVC family AIX user login.

**User Response:** Change the DB2\_PASS environment variable in your login profile to less than 18 characters. Change the CMVC family AIX login password to less than 18 characters. Run your login profile to update your environment. Refer to “For a DB2/6000 Server” on page 36 for more information.

---

**0011-108 The environment variable DB2\_PASS is not set.**

**Explanation:** You have not added the DB2\_PASS environment variable to your login profile. This environment variable refers to the password of your CMVC family AIX user login.

**User Response:** Add the DB2\_PASS environment variable to your login profile. Run your login profile to update your environment. Refer to “For a DB2/6000 Server” on page 36 for more information.

---

## Appendix B. Migrating to CMVC Version 2.3

This chapter provides the instructions for migrating to CMVC Version 2.3. After completing migration, you can convert from your current database to another database.:

- To convert ORACLE6 to ORACLE7, refer to Appendix C, “Converting Existing CMVC Server from ORACLE6 to ORACLE7” on page 195.
- To convert ORACLE6, ORACLE7, INFORMIX, or SYBASE to DB2/6000 on the CMVC Server/6000 only, refer to Appendix D, “Migrating to CMVC Server/6000 V2.3.0 for DB2/6000” on page 199.

**Note:** It is important that you complete the migration tasks in this chapter before migrating to a DB2/6000 database.

The most time-consuming migration task is backing up your family’s data. To minimize disruptions for your users, schedule the migration for an evening or a weekend.

After migrating to CMVC Version 2.3.0, your users can use these features:

- Configurable processes
- Configurable fields
- Alphanumeric defect and feature identifiers
- Ability to archive and restore CMVC data
- Ability to restrict access to a component
- Additional user exits
- Enhanced functionality in the graphical user interface.

---

## Databases Supported by Versions of CMVC

A CMVC family has many views, tables, and indexes in its relational database. Some of its views, tables, and indexes must be converted from earlier CMVC versions to CMVC Version 2.3.0 to make use of CMVC’s new features.

Figure 49 shows the various versions and releases of CMVC as well as the different databases supported by each of the releases.

Version	ORACLE 7	INFORMIX	SYBASE 4	SYBASE 10	DB2
CMVC Server/6000					
1.1.0					
1.1.1		X			
1.1.2		X	X		
2.1.0		X	X		
2.1.1	X	X	X		
2.2.0	X	X	X		X
2.3.0	X	X	X		X
CMVC Server for SUN					
2.1.0					
2.1.1	X	X			
2.2.0	X	X			
2.3.0	X	X	X	X	
CMVC Server for HP					
2.1.0					
2.1.1	X	X			
2.2.0	X	X			
2.3.0	X	X			
CMVC Server for Solaris					
2.3.0	X		X	X	

Figure 49. Databases Supported by Versions of CMVC

## Migration Utilities

The following utilities have been provided to help you convert the tables, views, and indexes to CMVC Version 2.3.0:

Table 1 (Page 1 of 2). Database conversion programs

From Version	To Version	Migration Tasks
1.1.0	1.1.1 or 1.1.2	Run <b>dbConvert.v1r1m1</b> (see "Using the dbConvert.v1r1m1 Utility" on page 181).
1.1.0	2.1.0 or 2.1.1	Run <b>dbConvert.v1r1m1</b> (see "Using the dbConvert.v1r1m1 Utility" on page 181). Then run <b>dbConvert.v2r1</b> (see "Using the dbConvert.v2r1 Utility" on page 181). However, do not do "Step 7. Creating Indexes."

Table 1 (Page 2 of 2). Database conversion programs

From Version	To Version	Migration Tasks
1.1.0	2.2.0	Run <b>dbConvert.v1r1m1</b> (see “Using the dbConvert.v1r1m1 Utility” on page 181). Then run <b>dbConvert.v2r1</b> (see “Using the dbConvert.v2r1 Utility” on page 181).
1.1.1	1.1.2	No conversion is required.
1.1.1	2.1.0 or 2.1.1	Run <b>dbConvert.v2r1</b> (see “Using the dbConvert.v2r1 Utility” on page 181). However, do not do “Step 7. Creating Indexes.”
1.1.1	2.2.0	Run <b>dbConvert.v2r1</b> (see “Using the dbConvert.v2r1 Utility” on page 181).
1.1.2	2.1.0 or 2.1.1	Run <b>dbConvert.v2r1</b> (see “Using the dbConvert.v2r1 Utility” on page 181).
1.1.2	2.2.0	Run <b>dbConvert.v2r1</b> (see “Using the dbConvert.v2r1 Utility” on page 181).
2.1.0	2.1.1	No conversion is required.
2.1.0	2.2.0	Create a new index. See “Step 7. Creating Indexes” on page 192 for more information.
2.1.1	2.2.0	Create a new index. See “Step 7. Creating Indexes” on page 192 for more information.
2.2.0	2.3.0	<ol style="list-style-type: none"> <li>1. If you are using DB2 with CMVC Version 2.2.0, run the <b>db2bind</b> program from CMVC Version 2.3.0. For usage information, type db2bind.</li> <li>2. Create new indexes. See “Step 7. Creating Indexes” on page 192 for more information.</li> </ol>

## Using the dbConvert.v1r1m1 Utility

If you are migrating from CMVC Version 1.1.0, run the **dbConvert.v1.r1m1** script before you do the pre-migration tasks (see “Pre-Migration Tasks”).

## Using the dbConvert.v2r1 Utility

Do the following pre-migration and migration tasks when migrating to a Version 2 level of CMVC. If you are migrating from Version 1.1.0, you must first run the **dbConvert.v1r1m1** utility.

## Pre-Migration Tasks

Before performing the migration tasks, do the following tasks:

1. Estimate the additional disk space needed in the relational database.
2. Ensure that the ORACLE rollback segment and related database logs have enough space.

## Step 1. Estimating Additional Relational Database Storage

Estimate the additional disk space needed in the relational database. Figure 50 lists the table names that must be converted and how many additional bytes each row needs. Also, 2 new tables, Cfgcomproc and Cfgrelproc, are created on the CMVC server.

Table Name	Additional Bytes Needed by Each Row
Defects	1500
Files	0035
AccessTable	0015
Releases	0046
Components	0043
Users	Insignificant
Cfgcomproc	Insignificant (new table)
Cfgrelproc	Insignificant (new table)

Figure 50. Bytes Required by Additional Rows in CMVC Database Tables

**Note:** A new Defects table is temporarily created during the conversion. Therefore, the additional space needed for the new Defects table is recoverable after the conversion.

To find out how many rows a table has, you can invoke one of the following commands in your CMVC family account to get into the respective ORACLE, INFORMIX or SYBASE database.

- **\$ORACLE\_HOME/bin/sqlplus familyName/\$ORACLE\_PASS**
- **\$INFORMIXDIR/bin/isql familyName**
- **\$SYBASE/bin/isql -U familyName -P \$SYBASE\_PASS**

Then use the following SQL command:

```
Select count(*) From tableName
```

where *tableName* is one of the table names listed in Figure 50.

Increase the amount of disk space by 50% to allow for indexes and conversion.

After you know how much more disk space you need in your relational database, ensure that you have at least that much free space in your relational database.

Figure 51 on page 183 is a sample estimate:

Table Name	Additional Bytes Required by Each Row	Number of Rows	Total KB	Number of Rows in Your Family	Total KB Required by Your Family
Defects	1500	10000	15000		
Files	35	100000	3500		
AccessTable	15	1000	15		
Releases	46	Insignificant			
Components	43	Insignificant			
Users	Insignificant	Insignificant			
Total			18515		

Figure 51. Example of Additional Space Calculation for the Relational Database

To estimate the additional disk space you require, round up the calculated figure to the nearest megabyte and multiply it by 1.5 to allow for indexes and expansion. In this example, the sample CMVC family needs approximately 30MB (20MB × 1.5) of free space in the relational database for database conversion.

Ensure that this space is added to the database device that is set in the ORACLE\_TBLSP, ORACLE\_NDXSP, INFORMIX\_DBSP, SYBASE\_DBDEV, and SYBASE\_LOGDEV environment variables, if you have them set. Otherwise, ensure that this space is added to the default database device.

Refer to Chapter 6, “Creating a CMVC Family” for the definitions of ORACLE\_TBLSP, ORACLE\_NDXSP, INFORMIX\_DBSP, SYBASE\_DBDEV, and SYBASE\_LOGDEV environment variables.

The conversion script may fail if you underestimate the amount of space to add to the relational database. If the conversion script fails, it will most likely do so when converting the Defects or Files tables. Later sections of this appendix explain how to recover from such a failure.

## Step 2: Reserving Additional Database Space

Ensure that the ORACLE rollback segments and the related database logs have sufficient space.

**Note:** The ORACLE database uses rollback segments and logs. The SYBASE and INFORMIX databases do not use rollback segments but do use logs.

The example in Figure 52 on page 184 assumes that your database has been set up to clean the logs every time a transaction is committed.

The Defects table is updated in a loop of, at most, 10000 rows at each iteration. The Files table is updated in a loop of, at most, 20000 rows at each iteration. If you have more than 10000 defects or 20000 files, or both, you need approximately 20MB of disk space in your ORACLE rollback segments and the related database logs to update these tables.

For other tables, you require disk space in the ORACLE rollback segments and the related database logs that is equal to twice the size of the table.

If these estimates differ, use the largest number (not the total) to set the size for your ORACLE rollback segments and the related database logs.

Figure 52 is a sample estimate:

Table Name	Number of Bytes Needed by Each Row	Number of Rows	Total KB	Number of Rows in Your Family	Total KB Needed in Your Family
Defects	1500	10000 each run	20000		
Files	400	20000 each run	20000		
AccessTable	75	1000	150		
Releases	350	insignificant			
Components	350	Insignificant			
Users	Insignificant	Insignificant			

Figure 52. Example of Additional Space Calculation for ORACLE Rollback Segments and Related Database Logs

The example in Figure 52 indicates that 20MB is required in the ORACLE rollback segments and the related database logs for the conversion.

If you use the INFORMIX-Online relational database, in stage C where the Files table is altered, you need twice the size of the Files table for the logs. In this example, you might need more than 20MB of free space in your INFORMIX logs.

If you use the SYBASE relational database, you may need to use the **sp\_dboption** command to change the database option **trunc. log on chkpt.** to true. Then use the **sp\_configure** command to change the recovery interval to a smaller number during the conversion. For details, refer to your SYBASE documentation.

## Migration Tasks

After completing the pre-migration steps, do the following to migrate to a Version 2 level of CMVC:

1. Stop the **cmvcd** and **notifyd** daemons.
2. Back up the database and the family directories.
3. Install the CMVC server code, Version 2.3 for the current database.
4. Run the **dbConvert.v2r1** conversion shell script.
5. Run the **chfield** command.
6. Update the Authority, Interest, Cfgcomproc, and Cfgrelproc tables.
7. Create an index for the Sequence table.
8. Start the **cmvcd** and **notifyd** daemons.
9. Verify the integrity of the family's data.



## Step 1. Stopping the cmvcd and notifyd Daemons

To ensure the integrity of your CMVC family's data, you must stop the **cmvcd** and **notifyd** daemons before running the database conversion shell script.

Refer to Chapter 13, "CMVC Server Daemons" for instructions on stopping the **cmvcd** and **notifyd** daemons.

## Step 2. Backing Up the Database

If errors occur during the conversion of the CMVC family's database, you may have to restore a table or the entire database before running the conversion script again. You must have a good backup copy of your CMVC family's relational database.

You can back up either all of the data belonging to your CMVC family or just the relational database.

Refer to Chapter 17, "Backup and Recovery" for recommendations on backing up the CMVC family or its relational database, or both.

## Step 3. Installing the CMVC Server Code, Version 2.3

The **dbConvert.v2r1** shell script is shipped with the CMVC server code, Version 2, Release 3. Refer to Chapter 3, "Installing the CMVC Servers" on page 13 for instructions on how to install the CMVC server code.

## Step 4. Running the dbConvert.v2r1 Conversion Script

A shell script called **dbConvert.v2r1** is shipped with CMVC Version 2.3.0 and is located in the **/usr/lpp/cmvc/install** directory. The CMVC family administrator can use this script to convert the database.

This script is written in the Bourne shell so that you can browse the file to find the SQL commands that perform the conversion.

Log in to the CMVC family account and type `dbConvert.v2r1` to start the conversion script. You are prompted to enter the stage to run. If you are running the script for the first time, select stage A. The script will continue to run through to the last stage, which is stage L. If the script fails at a particular stage, refer to the error log to correct the problem, and then do one of the following:

- Recover the whole database from the backup copy and rerun the shell script from stage A.
- Recover the affected table from the backup copy and rerun the shell script from the stage that failed.

**Note:** When recovering a table, you must also rebuild its indexes if they were not already rebuilt.

For a list of indexes for the affected table, refer to the file for the respective database:

Database	Name of File
ORACLE	/usr/lpp/cmvc/install/index.db
INFORMIX	/usr/lpp/cmvc/install/index.ifx
SYBASE	/usr/lpp/cmvc/install/index.syb

Copy the file to your family's **\$HOME** directory. Keep only the entries that relate to the affected table, then run the command:

```
cat fileName | $SQL
```

Where

Variable Name	Description
<i>fileName</i>	Name of the file that contains the commands to create indexes.
<i>\$SQL</i>	Set to one of the following: <ul style="list-style-type: none"><li>– \$ORACLE_HOME/bin/sqlplus familyName/\$ORACLE_PASS</li><li>– \$INFORMIXDIR/bin/isql familyName</li><li>– \$SYBASE/bin/isql -U familyName -P \$SYBASE_PASS</li></ul>

This command runs the SQL commands listed in the file *fileName*.

There is no need to recreate the views individually because all views will be re-created in Stage L.

- Browse the **dbConvert.v2r1** shell script to find the affected stage. Then manually run the remaining commands listed in that stage. Proceed to the next stage by running the **dbConvert.v2r1** shell script from the stage after the one that failed.

The following sections describe the stages of the **dbConvert.v2r1** shell script.

## Stage A

The database conversion script assumes that you do not have a newDefects table in your CMVC database. If you do, rename it before running the database conversion script. Rename it back to newDefects after the conversion.

The script creates the newDefects table and fills it with the data in the original Defects table. The following fields have been changed in the new table:

- **Defect name** and **duplicate name** are now alphanumeric fields.
- The **symptom**, **phaseFound**, **phaseInject**, **priority**, and **target** columns are moved to the end of the table. These columns are now configurable fields that are invisible to users until the **chfield** command is run. Refer to “Step 5. Running the chfield Command” on page 190 for instructions on running the **chfield** command to make these fields visible to users.

**Note:** To limit the amount of disk space required by the conversion, the Defects table is filled in a loop with a maximum of 10000 rows per iteration.

If this stage fails, you probably do not have to recover the database from the backup copy. To recover from the error, refer to the error log to correct the problem, and then run **dbConvert.v2r1** again from stage A. The Defects table is not dropped in stage A because, if stage A fails, the Defects table still remains in the database to be filled into the newDefects table.

## Stage B

In this stage, the original Defects table is dropped and the newDefects table is renamed to Defects.

If stage B fails, manually drop the Defects table and rename the newDefects table to Defects. Then, run **dbConvert.v2r1** from stage C to continue the database conversion. Refer to the “Procedure to run stage b” in the **dbConvert.v2r1** shell script for the commands to drop and rename the Defects table.

## Stage C

Stage C adds **userLogin** and **fmode** columns to the Files table.

If this stage fails, refer to the error log to correct the problem. You may have to recover the Files table from the backup copy if the table is damaged. Use one of the recovery options mentioned previously to continue the database conversion.

If you use the INFORMIX-Online relational database, you need twice the size of the Files table for the logs.

## Stage D

This stage updates the **userLogin** and **fmode** columns in the Files table. The **userLogin** is updated with the CMVC user ID of the person who has the file checked out or locked. The **fmode** column is updated with the value of 0444.

If this stage fails, you probably do not have to recover the database from the backup copy. To recover from the error, refer to the error log to correct the problem. Then, run **dbConvert.v2r1** again from stage D.

**Note:** To limit the amount of space required by the ORACLE rollback segment and the related database logs, the Files table is updated in a loop with a maximum of 20000 rows in each iteration.

## Stage E

This stage adds a new column called **authorityType** to the AccessTable table and initializes it to granted.

If this stage fails, refer to the error log to correct the problem. You may have to recover the AccessTable table from the backup copy if the table is damaged. Use one of the recovery options mentioned previously to continue the database conversion.

## Stage F

This stage adds and initializes some new columns to the Releases table. Refer to the “Procedure to run stage f” in the **dbConvert.v2r1** file for the names and attributes of the new columns and commands to initialize the new columns.

The **process** column is initialized with the values shown in Figure 53 on page 188.

Release Setting in CMVC V1R1	Value of the process Column
No binding control	no_track
Binding control without environment list or approver list	track_level
Binding control with environment list	track_test
Binding control with environment list and approver list	track_full
Binding control with approver list	track_approval

Figure 53. New Values for the process Column in the Releases Table

If this stage fails, refer to the error log to correct the problem. You may have to recover the Releases table from the backup copy if the table is damaged. Use one of the recovery options mentioned previously to continue the database conversion.

### Stage G

This stage adds and initializes some new columns to the Components table. Refer to the “Procedure to run stage g” in the **dbConvert.v2r1** file for the names and attributes of the new columns and commands to initialize the new columns. The **process** column is initialized with the value of default.

If this stage fails, refer to the error log to correct the problem. You may have to recover the Components table from the backup copy if the table is damaged. Use one of the recovery options mentioned previously to continue the database conversion.

### Stage H

This stage inserts a special user into the Users table. Refer to the “Procedure to run stage h” in the **dbConvert.v2r1** file for the values of the columns for this special user.

If this stage fails, probably the special user could not be inserted. Correct the problem and run the **dbConvert.v2r1** command again from this stage.

### Stage I

Stage I creates and initializes the new Cfgrelproc table. Before running this stage, ensure that you do not have a table called Cfgrelproc in your CMVC family database.

Figure 54 on page 189 shows the rows that are inserted into the Cfgrelproc table:

<b>name Column</b>	<b>config Column</b>
no_track	none
track_level	track
track_level	fix
track_level	level
track_test	track
track_test	fix
track_test	level
track_test	test
track_full	track
track_full	fix
track_full	level
track_full	test
track_full	approval
track_approval	track
track_approval	fix
track_approval	level
track_approval	approval

Figure 54. Rows Inserted into the Cfgrelproc Table

If this stage fails, correct the problem and rerun the **dbConvert.v2r1** command from this stage.

### Stage J and K

Stage J creates and initializes the new Cfgcomproc table. Before running this stage, ensure that you do not have a table called Cfgcomproc in your CMVC family database.

Figure 55 shows the rows that are inserted into the Cfgcomproc table:

<b>name Column</b>	<b>config Column</b>
default	verifyDefect
default	dsrFeature
default	verifyFeature

Figure 55. Rows Inserted into the Cfgcomproc Table

There are indexes in the database that need to be recreated and new indexes which must be created. Stage K drops these indexes before recreating them.

If either of these stages fail, correct the problem and rerun the **dbConvert.v2r1** command from the stage that failed.

## Stage L

Stage L drops all views before recreating them.

If this stage fails, correct the problem and rerun the **dbConvert.v2r1** command from this stage.

### Step 5. Running the **chfield** Command

When the Defects table is converted, the **symptom**, **phaseFound**, **phaseInject**, **priority** and **target** fields are moved to the end of the table. These fields are invisible to the users until the **chfield** command is run as follows:

```
chfield -object Defect -source default -s
```

```
chfield -object Feature -source default -s
```

Refer to “Creating and Modifying Configurable Fields” on page 51 for instructions on running the **chfield** command.

### Step 6. Updating Table Entries

CMVC Version 2.1.0 and later contains new entries in the IBM shipped **authority.ld** and **interest.ld** files. If the **authority.ld** and **interest.ld** files in the **\$HOME** directory of your CMVC family’s account are identical to the copies shipped with CMVC, Version 1, Release 1, Modification Level 2, copy the new files from the **/usr/lpp/cmvc/install** directory to replace the old files. Compare your old files with the new shipped copies before replacing the old files.

Figure 56 on page 191 shows the entries which have been added to the IBM shipped **authority.ld** file. Update your **authority.ld** file accordingly.

---

```

developer+|DefectDesign
developer+|DefectReview
developer+|DefectSize
writer+|DefectDesign
writer+|DefectReview
writer+|DefectSize
releaselead|DefectDesign
releaselead|FeatureAccept
releaselead|ReleaseLink
releaselead|TestAbstain
releaselead|TestAccept
releaselead|TestAssign
releaselead|TestReject
componentlead|AccessRestrict
componentlead|DefectCancel
componentlead|DefectDesign
componentlead|DefectReopen
componentlead|DefectReview
componentlead|DefectSize
componentlead|DefectVerify
componentlead|FeatureCancel
componentlead|FeatureReopen
componentlead|FeatureVerify
componentlead|VerifyAbstain
componentlead|VerifyAccept
componentlead|VerifyAssign
componentlead|VerifyReject
projectlead|AccessRestrict
projectlead|DefectCancel
projectlead|DefectDesign
projectlead|DefectReopen
projectlead|DefectReview
projectlead|DefectSize
projectlead|DefectVerify
projectlead|FeatureCancel
projectlead|FeatureReopen
projectlead|FeatureVerify
projectlead|ReleaseLink
projectlead|TestAbstain
projectlead|TestAccept
projectlead|TestAssign
projectlead|TestReject
projectlead|VerifyAbstain
projectlead|VerifyAccept
projectlead|VerifyAssign
projectlead|VerifyReject

```

---

Figure 56. New Entries in the *authority.ld* File Shipped by IBM

Figure 57 on page 192 shows the entries which have been added to the IBM shipped **interest.ld** file. Update your **interest.ld** file accordingly.

---

```

med|DefectDesign
med|ReleaseLink
high|AccessRestrict
high|DefectDesign
high|DefectReview
high|DefectSize
high|ReleaseLink

```

---

Figure 57. New Entries in the *interest.Id* File Shipped by IBM

After editing these files run the **chauth** and **chintr** commands.

Copy **/usr/lpp/cmvc/install/cfgrelproc.Id** to your family's **\$HOME** directory. Copy **/usr/lpp/cmvc/install/cfgcomproc.Id** to your family's **\$HOME** directory. Optionally, modify **\$HOME/cfgrelproc.Id** or **\$HOME/cfgcomproc.Id**, or both.

To change the **cfgrelproc.Id** file, refer to Stage I for the rows that are inserted into the **Cfgrelproc** table. To change these rows after you have brought your CMVC family up again, you must use the **Release -modify** command to modify your releases so the changes will take effect.

To change the **cfgcomproc.Id** file, refer to Stage J for the rows that are inserted into the **Cfgcomproc** table. To change these rows after you have brought your CMVC family up again, you must use the **Component -modify** command to modify your components so the changes will take effect.

Run the **chrelproc** and **chcomproc** commands.

Refer to Chapter 8, "Configuring Components and Releases" for instructions on using these commands.

## Step 7. Creating Indexes

In CMVC Server Version 2.3.0, new indexes are added. Follow this procedure to create the new indexes:

1. Use one of the following commands to gain access to the database:

- For DB2:

```

$DB2_HOME/sql1lib/bin/db2    connect to familyName
$DB2_HOME/sql1lib/bin/db2

```

- For ORACLE:

```

$ORACLE_HOME/bin/sqlplus    familyName/$ORACLE_PASS

```

- For INFORMIX:

```

$INFORMIXDIR/bin/isql       familyName

```

- For SYBASE:

```

$SYBASE/bin/isql           -U familyName -P $SYBASE_PASS

```

2. Use the following command to create a new index called **XSeqName** in the **Sequence** table for the **name** attribute:

```

CREATE UNIQUE INDEX XSeqName ON Sequence (name)

```



3. Use the following command to create new indexes for the **Files** table:

```
CREATE INDEX XFilPid ON Files (pathId)
```

```
CREATE INDEX XFilesSrcId ON Files (sourceId)
```

4. Use the **exit** command to exit the database. For DB2, type terminate.

### **Step 8: Starting the cmvcd and notifyd Daemons**

The procedure to start the **cmvcd** daemon has changed. You must now start it from your CMVC family account. You no longer have to switch to root. Refer to Chapter 13, “CMVC Server Daemons” for instructions on starting the **cmvcd** and **notifyd** daemons.

### **Step 9: Verifying Your Family’s Data**

Focus the verification on those commands that read or write to the converted tables, for example, **Defect**, **Feature**, **File**, **Access**, **Release**, **Component**, and **User** commands.



---

## Appendix C. Converting Existing CMVC Server from ORACLE6 to ORACLE7

There are different ways to migrate an ORACLE6 database to ORACLE7. This appendix is to help you convert your existing CMVC server from ORACLE6 to ORACLE7 by exporting the CMVC family tables, views, and indexes from ORACLE6, and importing them to ORACLE7. Refer to the document *ORACLE7 Server Migration Guide* that comes with the ORACLE7 product for more information.

---

### Prerequisites

You must install the CMVC server code for ORACLE7. See Chapter 3, “Installing the CMVC Servers” on page 13 for detailed information about the installation procedures.

---

### Steps for Migration

This appendix contains instructions for the three steps to convert an ORACLE6 database to an ORACLE7 database for the CMVC Version 2.3.0 server. They are:

- Export the CMVC family tables, views, and indexes from ORACLE6
- Import the CMVC family tables, views, and indexes to ORACLE7
- Run the database conversion routine for the CMVC server.

You should also create a new index for the Sequence table by following the procedure described in “Step 7. Creating Indexes” on page 192.

### Exporting from ORACLE6

To export CMVC family tables, views, and indexes from ORACLE6:

- From your CMVC family login, type:

```
$ORACLE_HOME/bin/exp $ORACLE_DBA BUFFER=40000 FILE=fileName\
GRANTS=N INDEXES=Y ROWS=Y CONSTRAINTS=N COMPRESS=Y FULL=N \
RECORD=N OWNER=familyName
```

Where:

Variable	Description
<i>fileName</i>	The name of the output file for the <b>exp</b> command. Refer to your ORACLE6 documentation for more information about the <b>exp</b> command.
<i>familyName</i>	Your CMVC family name.

### Importing to ORACLE7

To import CMVC family tables, views, and indexes to ORACLE7:

1. From your CMVC family login, enter the *sqlplus* command:

```
$ORACLE_HOME/bin/sqlplus $ORACLE_DBA
```

2. At the *sqlplus* prompt, create a CMVC family ORACLE7 user ID by typing:

```
GRANT CONNECT, RESOURCE TO familyName IDENTIFIED BY password;
```

Where:

Variable	Description
<i>familyName</i>	Your CMVC family name
<i>password</i>	The password of your CMVC family ORACLE7 user ID. This password must be the same as the one defined in the ORACLE_PASS environment variable.

3. Do one of the following:

- If you do not have an ORACLE7 tablespace created for your CMVC family tables, go to Step 4.
- If you have an ORACLE7 tablespace created for your CMVC family tables and you have it set in the ORACLE\_TBLSP environment variable, type the following command at the *sqlplus* prompt:

```
ALTER USER familyName DEFAULT TABLESPACE tablespaceName;
```

Where:

Variable	Description
<i>familyName</i>	Your CMVC family name
<i>tablespaceName</i>	The ORACLE7 tablespace name that you have created for your CMVC family tables.

4. Exit from the *sqlplus* prompt by typing:

```
EXIT
```

5. Import the CMVC family tables, views, and indexes to ORACLE7 by typing:

```
$ORACLE_HOME/bin/imp $ORACLE_DBA BUFFER=40000 FILE=fileName COMMIT=Y \  
SHOW=N IGNORE=N GRANTS=N ROWS=Y DESTROY=N FULL=N INDEXES=x \  
FROMUSER=familyName TOUSER=familyName
```

Where:

Variable	Description
<i>fileName</i>	The name of the output file for the <b>exp</b> command.
<i>x</i>	Select one of the following values: <ul style="list-style-type: none"><li>Specify a value of Y if you do not have an ORACLE7 tablespace created for your CMVC family indexes. Go to the next procedure “Run the Database Conversion Routine” on page 197.</li><li>Specify N if you have a tablespace created. Go to Step 6.</li></ul>
<i>familyName</i>	Your CMVC family name.

**Notes:**

- The DESTROY=N argument is only available in ORACLE7, not in ORACLE6.
- If you have to run the above command again, change the value of the attribute IGNORE from N to Y.
- Refer to your ORACLE7 documentation for the syntax of the **imp** and **sqlplus** commands.

6. Run the following command to create the indexes:

```
sed "s/TABLESPACENAME/TABLESPACE $ORACLE_NDXSP/g" \
$CMVC_HOME/install/index.db | $ORACLE_HOME/bin/sqlplus \
familyName/$ORACLE_PASS
```

Where:

Variable	Description
<i>TABLESPACENAME</i>	The place-holder found in the <b>\$CMVC_HOME/install/index.db</b> file to be replaced with the keyword <b>TABLESPACE</b> and the ORACLE7 tablespace name that you set in the ORACLE_NDXSP environment variable.
<i>familyName</i>	Your CMVC family name.

## Run the Database Conversion Routine

After you have installed the CMVC Server version 2.3.0 for ORACLE7 and have completed the migration of the database from ORACLE6 to ORACLE7, run one of the following database conversion routines from your CMVC family login. Your selection of the routines depends on the previous version of your CMVC Server for ORACLE6 that you upgraded from.

Previous Version of CMVC Server for ORACLE6	Database Conversion Routines
V1.1.0 (for RISC System/6000 only)	dbConvert.v1r1m1 followed by dbConvert.v2r1
V1.1.1 (for RISC System/6000 only)	dbConvert.v2r1
V1.1.2 (for RISC System/6000 only)	dbConvert.v2r1
V2.1.0	dbConvert.v2r1m1
V2.1.1	dbConvert.v2r1m1
V2.2.0	dbConvert.v2r1m1

Follow the procedure listed in “Step 7. Creating Indexes” on page 192 to create a new index for the Sequence table, which has been added to CMVC Server Version 2.3.0.



---

## Appendix D. Migrating to CMVC Server/6000 V2.3.0 for DB2/6000

This appendix is to help you plan the migration from CMVC Server/6000 Version 2.1.0, 2.1.1, 2.2.0 or 2.3.0 for ORACLE, ORACLE7, INFORMIX, or SYBASE databases to CMVC Server/6000 Version 2.3.0 for DB2/6000.

---

### Prerequisites

- If you have CMVC Version 1.1, you must convert to CMVC Version 2.1.0, 2.1.1, 2.2.0, or 2.3.0 before following the instructions in this appendix. See Appendix B, "Migrating to CMVC Version 2.3" on page 179 for the detailed instructions on migrating from earlier versions of CMVC Server.
- You must install the DB2/6000 database on your CMVC Server/6000.

---

### Steps for Migration

The following steps are required to migrate your ORACLE, ORACLE7, INFORMIX, or SYBASE databases to CMVC Server/6000 Version 2.3.0 for DB2/6000:

#### Pre-Migration Tasks

1. Estimate the additional disk space needed during migration
2. Install the DB2/6000 database

#### Migration Tasks

1. Stop the **cmvcd** and **notifyd** daemons
2. Backup the database
3. Install CMVC Server/6000 Version 2.3.0 for DB2/6000
4. Run the migration
5. Start the **cmvcd** and **notifyd** daemons
6. Verify the integrity of the family's data.

#### Post-Migration Task

Reclaim your disk space by dropping the previous database.

Steps 1 and 2 are pre-migration tasks that should be done before you start the migration. It is recommended that you schedule the completion of these two tasks ahead of the migration task. Doing them up front enables you to have enough time to recover in the event that you do not have sufficient disk space to complete the tasks.

During migration, the most time-consuming tasks are backing up your family's data and migration of database. To minimize disruptions for your users, schedule the migration for an evening or a weekend.

## Steps Required to Perform the Database Migration

### Estimating the additional disk space

You need 15MB of hard disk space to install the DB2/6000 database.

A CMVC family has 32 tables in its relational database. All of these tables must be moved to the DB2/6000 database.

All of these tables have columns of numbers or characters. Two of these tables namely Notes and Versions also have remarks column that takes 32KB of characters, including | (vertical bar) and \n (new line).

The Notes and Versions tables will be fetched from the original database and inserted into the DB2/6000 database.

As a result, you must run the **mkdb** command to make the CMVC database in DB2/6000. This requires 15MB of hard disk space.

To find out how many rows a table has:

1. Invoke one of the following commands in your CMVC family account to get into the database:

Database Name	Command to Use
ORACLE6, ORACLE7	<code>\$ORACLE_HOME/bin/sqlplus <i>familyName</i>/\$ORACLE_PASS</code>
INFORMIX	<code>\$INFORMIXDIR/bin/isql <i>familyName</i></code>
SYBASE	<code>\$SYBASE/bin/isql -U <i>familyName</i> -P \$SYBASE_PASS</code>

where *familyName* is the name of the CMVC family you are migrating.

2. Use the following SQL command:

```
Select count(*) from tableName
```

where *tableName* is one of the table names listed in the following sections of this appendix.

It takes about 1KB per Notes row and 0.75KB per Versions row in a DB2/6000 database. As a result, you need the following additional disk space in your DB2/6000 database:

Table name	Bytes per row	Number of rows	Total MB	Number of rows in your family	Total MB required by your family	Moved to DB2/6000	Verified count in DB2/6000
Notes	1000	010000	10				
Versions	0750	100000	75				
Total			85				

Figure 58. Calculating required space for Notes and Versions tables



The above example shows that you need 85MB in your DB2/6000 database for Notes and Versions tables.

All other tables will be dumped to flat files from the original database and imported into the DB2/6000 database.

These tables can be dumped to a disk all at the same time or individually, before backing them up on tapes. Therefore, the required size of the working disk space should be either large enough to hold the flat files for all the tables or the biggest table.

<i>Table 2 (Page 1 of 2). Calculating required space for all CMVC tables</i>								
<b>Table name</b>	<b>Bytes per row</b>	<b>Number of rows</b>	<b>Total MB</b>	<b>Number of rows in your family</b>	<b>Total MB required by your family</b>	<b>Dumped to flat file</b>	<b>Moved to DB2/6000</b>	<b>Verified count in DB2/6000</b>
AccessTable	050	00500	00.025					
Approvals	065	02500	00.163					
Approvers	016	00500	00.008					
Authority	035	00500	00.018					
Cfgcomproc	035	00500	00.018					
Cfgrelproc	035	00500	00.018					
Changes	055	20000	01.100					
CompMembers	032	00500	00.016					
Components	250	00500	00.125					
Config	150	00500	00.075					
Coreqs	016	00500	00.008					
Defects	440	03500	01.540					
Environments	040	00500	00.020					
Files	285	75000	21.375					
FilesOut	080	00500	00.040					
Fix	075	03000	00.225					

<i>Table 2 (Page 2 of 2). Calculating required space for all CMVC tables</i>								
Table name	Bytes per row	Number of rows	Total MB	Number of rows in your family	Total MB required by your family	Dumped to flat file	Moved to DB2/6000	Verified count in DB2/6000
History	050	25000	01.250					
Hosts	135	00500	00.068					
Interest	035	00500	00.018					
LevelMembers	016	02500	00.040					
Levels	150	00500	00.075					
Notification	040	00500	00.020					
Path	155	50000	07.750					
Releases	245	00500	00.123					
Sequence	020	00003	00.001					
Sizes	170	00500	00.085					
Tests	090	05500	00.495					
Tracks	130	02500	00.325					
Users	280	00500	00.140					
Verify	080	02000	00.160					
Total			36MB					

The above example shows that you need 36MB of working disk space to dump all the tables to flat files. If disk space is a concern, you can use a 22MB disk to hold the biggest table (Files) and archive that file to tape before dumping other tables.

**Note:** If you archived the flat file for the Files table to tape, you need to unarchive the flat file from tape before you could import the Files table.

To import these flat files in your DB2/6000 database, you need an additional 50 per cent disk space for the indexes. Therefore, the total disk space required is:

$$36\text{MB} \times 1.5 = 54\text{MB}$$

## Installing the DB2/6000 database

You must install the DB2/6000 database before the conversion. Refer to the DB2/6000 documents for detailed instructions.

## Stopping the cmvcd and notifyd daemons

To ensure the integrity of your CMVC family's data, you must stop the **cmvcd** and **notifyd** daemons before running the database conversion commands.

Refer to Chapter 13, "CMVC Server Daemons" on page 95 for instructions on stopping these two daemons.

## Backing up the database

If errors occur during the conversion of the CMVC family's database, it is very likely that you would not need to restore the database from the backup copy. However, you should have a good backup copy of your CMVC family's relational database for contingency purposes.

You can backup either all the data belonging to your CMVC family or just the relational database.

Refer to Chapter 17, "Backup and Recovery" on page 137 for recommendations on backing up the CMVC family or its relational database or both.

## Installing the CMVC Server/6000 V2.3.0 for DB2/6000

The conversion scripts are shipped with the CMVC Server/6000 for DB2/6000. Refer to Chapter 3, "Installing the CMVC Servers" on page 13 for instructions on how to install the CMVC server code.

If for some reason, you decided not to continue with the conversion, you must install the CMVC Server/6000 Version 2.2.0, 2.1.1, or 2.1.0 for your database such as ORACLE. There is no conversion script involved in upgrading CMVC from Version 2.1.0 to versions 2.1.1 or 2.2.0.

## Running the conversion

### Run mkdb to create a DB2/6000 database

Before you run the conversion scripts, you must run **mkdb** to create the DB2/6000 database. Do not use the **-d** option in the **mkdb** command as the command needs to read the `$HOME/configField` files to ensure that the Defects, Files, and Users tables are created to whether you have had configurable fields for these objects. Refer to "Using the mkdb Command" on page 43 for more information.

### Tools shipped with CMVC Server/6000 for DB2/6000

The following shell scripts and commands are shipped with the CMVC Server/6000 for DB2/6000 code, and are located in the `$CMVC_HOME/db2Convert` directory. You can use them to convert the database.

From database	Move Notes/Versions	Dump tables	Load tables
ORACLE	orac2db2rems	oracdumtbls	orac2db2tbls
ORACLE7	ora72db2rems	ora7dumtbls	ora72db2tbls
INFORMIX	infx2db2rems	dbexport	infx2db2tbls
SYBASE	syb2db2rems	sybdumtbls	syb2db2tbls

### Tools to move Notes and Versions tables to DB2/6000

Use the **xxxx2db2rems** commands to move the Notes and Versions tables to the DB2/6000 database. You run this command in the family login:

```
xxxx2db2rems
```

Besides all the mandatory environment variables in your login profile (see "Mandatory Environment Variables in Your Login Profile" on page 35), this command also requires the following environment variables to be set:

Database	Environment Variable Required
All	CMVC_FAMILY and DB2_PASS
ORACLE6, ORACLE7	ORACLE_PASS
SYBASE	SYBASE_PASS

The **xxxx2db2rems** command deletes old records from the Notes and Versions tables in the DB2/6000 database before inserting them with the rows fetched from the original database.

If you plan to move the DB2/6000 database to another machine after you have run the **xxxx2db2rems** command, it is a good time to do a backup of the DB2/6000 database and move it to another machine before continuing the conversion.

### Tools to dump other tables to flat files

Other tables will be dumped to flat files delimited by "|". As a result, if the value of some of the columns contain "|", you have to replace it with something else (for example, "!") after the dump.

**ORACLE, ORACLE7 and SYBASE families:** Use the **xxxxdumtbls** commands to dump other tables to flat files. You run this command in the family login.

```
xxxxdumtbls
```

Besides all the mandatory environment variables in your login profile, this command also requires the FLATFILEDIR environment variable to be set. This is the directory name where you want the flat files to be created.

If disk space is a concern, you can make enough room to hold the biggest table in this directory. Then you dump one table at a time and back up the table to another medium such as an archive tape. Otherwise, you can dump all tables at the same time.

Depending on the database you use, the following environment variables are also required:

Database	Environment Variable Required
ORACLE6, ORACLE7	ORACLE_PASS
SYBASE	SYBASE_PASS

If you decided to dump one table at a time, check Table 2 on page 201 for the names of tables. Mark the above table with a cross (X) once you have dumped the table.

**INFORMIX families:** The INFORMIX database has a tool called **dbexport** that can be used to dump the whole database to flat files. As a result, it will also dump the Notes and Versions tables that have been moved to the DB2/6000 database.

If disk space is a concern, before you run **dbexport**, you can drop the Notes and Versions tables in the INFORMIX database. (This is not applicable to the DB2/6000 database.)

**Warning:** Before you do this, make sure you have a good backup of your INFORMIX database and that you know how to recover from the backup in the event of a contingency. As a result, the flat files will not include the Notes and Versions tables.

To use the **dbexport** command, it is recommended that you invoke it as follows:

```
dbexport -o dirName familyName
```

Where *dirName* is the directory where you want to dump the flat files.

Then set the FLATFILEDIR environment variable, for example in a Korn shell:

```
export FLATFILEDIR=dirName/familyName.exp
```

Where *familyName* is the name of you family.

### Tools to import other tables from flat files

Use the **xxxx2db2tbls** command to import the flat files into the DB2/6000 database. Before you import the flat files, edit the files to verify that they do not contain the vertical bar character (|) or the next line characters (\n) as the values of columns. If they do contain these characters, change them to something else, such as an exclamation point (!) or blank spaces.

You run this command in the family login:

```
xxxx2db2tbls
```

This command requires the FLATFILEDIR environment variable set to the directory where you had the flat files dumped. It also needs the DB2\_HOME environment variable to be set.

You can import all the tables at the same time if you have all flat files in the FLATFILEDIR directory, or you can import one table at a time.

If you decided to import one table at a time, check Table 2 on page 201 for the names of tables. Mark the above table with a cross (X) once you have imported the table.

After the import, the import log file is kept in `/tmp/$$.$$.tableName.unl` where `$$` is the process ID that runs the `xxxx2db2tbls` command.

Make sure that you check them.

**Notes:**

1. Make sure that you check the import log file.
2. A common error found in the `/tmp/$$.$$.tableName.unl` is that the DB2/6000 `import` command does not import records which has mismatched double quotes in the character columns. You should change them to another character, such as a *blank space*, and rerun `xxxx2db2tbls` for that table.

## Update Releases Table

If you are migrating to DB2 from ORACLE6, ORACLE7, INFORMIX, or SYBASE with CMVC version 2.1.0, 1.1.2, 1.1.1 or 1.1.0., you must set the `compId`, `userId`, and `relSize` in the Releases table to 0 instead of NULL.

To do this, enter the following commands from the `familyName` login:

```
db2 connect to familyName
db2 update Releases set compId=0, userId=0, relSize=0 where id=0
db2 connect reset
```

If you do not perform these instructions, some commands will fail, such as `Defect -assign` and `Level -complete`.

## Starting the cmvcd and notifyd daemons

Refer to Chapter 13, "CMVC Server Daemons" on page 95 for instructions on starting the `cmvcd` and `notifyd` daemons.

## Verifying the data of your family

- After you have converted to the DB2/6000 database, remove the `ORACLE_HOME`, `ORACLE_PASS`, `INFORMIXDIR`, `SYBASE`, and `SYBASE_PASS` environment variables from your profile.
- Rerun the profile to make sure that you are using the correct database.
- If you have tools to start and stop `cmvcd` and `notifyd` automatically, you should change them accordingly.
- After you have made sure that the `cmvcd` daemon is running with the DB2/6000 database, verify all the objects in your family by running some Read and Write operations on some of them.
- You might want to go to the DB2/6000 database by entering:

```
db2 connect to familyName
```

Then use

```
db2 "select count(*) from tableName"
```

to count and verify the counts you had from the original database. Mark them in Figure 58 on page 200.

To reset from the DB2/6000 database, enter:

```
db2 connect reset
```

If you have any performance problems, refer to the documentation for the DB2/6000 database to reorganize your database.

## **Reclaiming disk space**

Once you are satisfied with the new database, you can drop the original database to reclaim disk space.





## Appendix E. Authority Groups Worksheet

The following table lists CMVC actions. Use this table to record the authority groups for your family if you are using groups other than those supplied by IBM. CMVC actions that cannot be included in an authority group are marked with information about how they can be performed.

CMVC Actions	Authority Groups for Family: _____									
AccessCreate										
AccessDelete										
AccessRestrict										
ApprovalAbstain										
ApprovalAccept										
ApprovalAssign										
ApprovalCreate										
ApprovalDelete										
ApprovalReject										
ApproverCreate										
ApproverDelete										
CompCreate										
CompDelete										
CompLink										
CompModify										
CompRecreate										
CompUnlink										
CompView										
CoreqCreate										
CoreqDelete										
DefectAccept										
DefectAssign										
DefectCancel										
DefectClose	Automatic action									
DefectComment	Base authority									
DefectDesign										

CMVC Actions	Authority Groups for Family: _____									
DefectModify										
DefectOpen	Base authority									
DefectReopen										
DefectReturn										
DefectReview										
DefectSize										
DefectVerify										
DefectView										
EnvCreate										
EnvDelete										
EnvModify										
FeatureAccept										
FeatureAssign										
FeatureCancel										
FeatureClose	Automatic action									
FeatureComment	Base authority									
FeatureDesign										
FeatureModify										
FeatureOpen	Base authority									
FeatureReopen	Superuser or originator implicit authority									
FeatureReturn										
FeatureReview										
FeatureSize										
FeatureVerify										
FeatureView										
FileAdd										
FileCheckIn										
FileForceIn										
FileCheckOut										
FileForceOut										
FileDelete										
FileDeleteForce										

CMVC Actions	Authority Groups for Family: _____									
FileDestroy										
FileExtract										
FileLink										
FileLock										
FileLockForce										
FileModify										
FileRecreate										
FileRecreaForce										
FileRename										
FileRenameForce										
FileResolve	Base authority									
FileUndo										
FileUndoForce										
FileUnlock										
FileView										
FixActive										
FixAssign										
FixComplete										
FixCreate										
FixDelete										
HostCreate	Superuser or owner implicit authority									
HostDelete	Superuser or owner implicit authority									
LevelAssign										
LevelCheck										
LevelCommit										
LevelComplete										
LevelCreate										
LevelDelete										
LevelExtract										
LevelModify										
LevelView										
MemberCreate										

CMVC Actions	Authority Groups for Family: _____									
MemberDelete										
NotifyCreate										
NotifyDelete										
ReleaseCreate										
ReleaseDelete										
ReleaseExtract										
ReleaseLink										
ReleaseModify										
ReleaseRecreate										
ReleaseView										
Report	Base authority									
SizeAccept										
SizeAssign										
SizeCreate										
SizeDelete										
SizeReject										
TestAbstain										
TestAccept										
TestAssign										
TestReady	Automatic action									
TestReject										
TrackAssign										
TrackCancel										
TrackCheck										
TrackCommit										
TrackComplete										
TrackCreate										
TrackFix										
TrackIntegrate										
TrackModify										
TrackTest										
TrackView										

CMVC Actions	Authority Groups for Family: _____									
UserCreate	Superuser implicit authority									
UserDelete	Superuser implicit authority									
UserModify	Superuser or owner implicit authority									
UserRecreate	Superuser implicit authority									
UserView										
VerifyAbstain										
VerifyAccept										
VerifyAssign										
VerifyReady	Automatic action									
VerifyReject										



## Appendix F. Interest Groups Worksheet

The following table lists the CMVC actions. Use this table to record the interest groups for your family if you are using groups other than those supplied by IBM. Those CMVC actions which cannot be included in an interest group are marked with information about how users are notified.

CMVC Actions	Interest Groups for Family: _____						
AccessCreate							
AccessDelete							
AccessRestrict							
ApprovalAbstain							
ApprovalAccept							
ApprovalAssign							
ApprovalCreate							
ApprovalDelete							
ApprovalReject							
ApproverCreate							
ApproverDelete							
CompCreate	New owner implicit notification						
CompDelete							
CompLink							
CompModify							
CompRecreate							
CompUnlink							
CompView	No notification						
CoreqCreate	No notification						
CoreqDelete	No notification						
DefectAccept							
DefectAssign							
DefectCancel							
DefectClose							
DefectComment							
DefectDesign							
DefectModify							
DefectOpen							
DefectReopen							

CMVC Actions	Interest Groups for Family: _____						
DefectReview							
DefectReturn							
DefectSize							
DefectVerify							
DefectView	No notification						
EnvCreate							
EnvDelete							
EnvModify							
FeatureAccept							
FeatureAssign							
FeatureCancel							
FeatureClose							
FeatureComment							
FeatureDesign							
FeatureModify							
FeatureOpen							
FeatureReopen							
FeatureReturn							
FeatureReview							
FeatureSize							
FeatureVerify							
FeatureView	No notification						
FileAdd							
FileCheckIn							
FileCheckOut							
FileDelete							
FileDestroy							
FileExtract	No notification						
FileForceIn							
FileForceOut							
FileLink							
FileLock							
FileLockForce	FileLock subscribers						
FileModify							
FileRecreate							



CMVC Actions	Interest Groups for Family: _____						
FileRecreaForce	FileRecreate subscribers						
FileRename							
FileRenameForce	FileRename subscribers						
FileResolve	No notification						
FileUnlock							
FileUndo							
FileUndoForce	FileUndo subscribers						
FileView	No notification						
FixActive							
FixAssign							
FixComplete							
FixCreate							
FixDelete							
HostCreate	No notification						
HostDelete	No notification						
LevelAssign							
LevelCheck	No notification						
LevelCommit							
LevelComplete							
LevelCreate							
LevelDelete							
LevelExtract	No notification						
LevelModify							
LevelView	No notification						
MemberCreate							
MemberDelete							
NotifyCreate	No notification						
NotifyDelete	No notification						
ReleaseCreate							
ReleaseDelete							
ReleaseExtract	No notification						
ReleaseLink							
ReleaseModify							
ReleaseRecreate							
ReleaseView	No notification						

CMVC Actions	Interest Groups for Family: _____						
Report	No notification						
SizeAccept							
SizeAssign							
SizeCreate							
SizeDelete							
SizeReject							
TestAbstain							
TestAccept							
TestAssign							
TestReady	Owner implicit notification						
TestReject							
TrackAssign							
TrackCancel							
TrackCheck	No notification						
TrackCommit							
TrackComplete							
TrackCreate							
TrackFix							
TrackIntegrate							
TrackModify							
TrackTest							
TrackView	No notification						
UserCreate	New user implicit notification						
UserDelete	No notification						
UserModify	No notification						
UserUnDelete	No notification						
UserView	No notification						
VerifyAbstain							
VerifyAccept							
VerifyAssign							
VerifyReady	Owner implicit notification						
VerifyReject							

## Appendix G. Configurable Processes Worksheets

The following worksheets list the CMVC subprocesses. Use these worksheets to create processes from a combination of CMVC subprocesses. Separate worksheets are provided for component and release processes. For more information on configurable processes, refer to Chapter 8, "Configuring Components and Releases" on page 59.

CMVC Component Subprocesses	Component Processes for _____					
dsrDefect						
dsrFeature						
verifyDefect						
verifyFeature						
none						

CMVC Release Subprocesses	Release Processes for _____					
track						
approval						
fix						
level						
test						
none						



## Appendix H. User Exit Parameters

The following table shows the parameters passed to each user exit program defined for a specific CMVC action and ExitID. A description of the parameters follows the table.

**Note:** Parameters are not shown for ExitID 3. The parameters for ExitID 3 are the same as those passed to ExitID 0, with an additional parameter as the last parameter to indicate the last user ExitID that has been executed successfully, for example, 0 or 1.

A parameter name followed by *not used* indicates that CMVC passes an empty string.

See Chapter 12, "Providing User Exits" for more information on user exits.

### Parameters Passed to User Exit Programs

Figure 59 shows the parameters passed to each user exit program defined for a specific CMVC action and ExitID.

CMVC Action	ExitID	Parameters Passed to the User Exit Program
<b>Access</b>		
AccessCreate	0	CMVC user ID, component name, authority group name, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
AccessDelete	0	CMVC user ID, component name, authority group name, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
AccessRestrict	0	CMVC user ID, component name, authority group name, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
<b>Approval</b>		
ApprovalAbstain	0	release name, defect or feature number, approver's name, effective CMVC user ID, real Unix login, verbose flag
	1	release name, defect or feature number, approver's name, track type, effective CMVC user ID, verbose flag
	2	same as ExitID 1

Figure 59 (Part 1 of 15). The CMVC Action, Exit ID and Parameters That Are Passed to User Exit Programs

<b>CMVC Action</b>	<b>ExitID</b>	<b>Parameters Passed to the User Exit Program</b>
ApprovalAccept	0	release name, defect or feature number, approver's name, effective CMVC user ID, real Unix login, verbose flag
	1	release name, defect or feature number, approver's name, track type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
ApprovalAssign	0	release name, defect or feature number, old approver's name, new approver's name, effective CMVC user ID, real Unix login, verbose flag
	1	release name, defect or feature number, old approver's name, new approver's name, track type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
ApprovalCreate	0	release name, defect or feature number, approver's name, effective CMVC user ID, real Unix login, verbose flag
	1	release name, defect or feature number, approver's name, track type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
ApprovalDelete	0	release name, defect or feature number, approver's name, effective CMVC user ID, real Unix login, verbose flag
	1	release name, defect or feature number, approver's name, track type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
ApprovalReject	0	release name, defect or feature number, approver's name, effective CMVC user ID, real Unix login, verbose flag
	1	release name, defect or feature number, approver's name, track type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
<b>Approver</b>		
ApproverCreate	0	CMVC user ID, release name, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
ApproverDelete	0	CMVC user ID, release name, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
<b>Component</b>		
CompCreate	0	component name, parent component, owner, component process, description, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
CompDelete	0	component name, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login

Figure 59 (Part 2 of 15). The CMVC Action, Exit ID and Parameters That Are Passed to User Exit Programs

<b>CMVC Action</b>	<b>ExitID</b>	<b>Parameters Passed to the User Exit Program</b>
CompLink	0	component name, parent component, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
CompModify	0	component name, new component name, owner, new description, new component process, effective CMVC user ID, real Unix login, verbose flag
	1	component name, new component name, old owner, new owner, old description, new description, old component process, new component process, date of last update, effective CMVC user ID, verbose flag
	2	same as ExitID 0 without real Unix login
CompRecreate	0	component name, parent component, effective CMVC user ID, real Unix login, verbose flag
	1	component name, parent component, old dropDate, effective CMVC user ID, verbose flag
	2	same as ExitID 1
CompUnlink	0	component name, parent component, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
CompView	0	component name, display type, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
<b>Coreq</b>		
CoreqCreate	0	release name, prime track name, second track name, effective CMVC user ID, real Unix login, verbose flag
	1	release name, prime track name, second track name, prime track type, second track type, effective CMVC user ID, verbose flag
	2	same as Exit ID 1
CoreqDelete	0	release name, defect or feature number, effective CMVC user ID, real Unix login, verbose flag
	1	release name, defect or feature number, track type, effective CMVC user ID, verbose flag
	2	same as Exit ID 1
<b>Defect</b>		
DefectAccept	0	defect number, original defect number (not used), answerAccept, remarks, effective CMVC user ID, real Unix login, verbose flag
	1	defect number, answerAccept, remarks, effective CMVC user ID, verbose flag
	2	same as ExitID 0 without real Unix login

Figure 59 (Part 3 of 15). The CMVC Action, Exit ID and Parameters That Are Passed to User Exit Programs

<b>CMVC Action</b>	<b>ExitID</b>	<b>Parameters Passed to the User Exit Program</b>
DefectAssign	0	defect number, new component name, new owner's name, remarks, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
DefectCancel	0	defect number, original defect number (not used), answer (not used), remarks, effective CMVC user ID, real Unix login, verbose flag
	1	defect number, remarks, effective CMVC user ID, verbose flag
	2	same as ExitID 0 without real Unix login
DefectComment	0	defect number, remarks, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
DefectDesign	0	defect number, original defect number (not used), answer (not used), remarks, effective CMVC user ID, real Unix login, verbose flag
	1	defect number, remarks, effective CMVC user ID, verbose flag
	2	same as ExitID 0 without real Unix login
DefectModify	0	defect number, new defect number, severity, environment name, prefix, reference, level name, abstract, originator, answer, remarks, release name, configurable field string, effective CMVC user ID, real Unix login, verbose flag
	1	defect number, new defect number, severity, environment name, prefix, reference, level name, abstract, originator, answer, remarks, release name, configurable field string, date of last update, effective CMVC user ID, verbose flag
	2	same as ExitID 0 without real Unix login
DefectOpen	0	component name, prefix, severity, reference, environment name, remarks, level name, abstract, release name, configurable field string, defect number, effective CMVC user ID, real Unix Login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
DefectReopen	0	defect number, original defect number (not used), answer (not used), remarks, effective CMVC user ID, real Unix login, verbose flag
	1	defect number, remarks, effective CMVC user ID, verbose flag
	2	same as ExitID 0 without real Unix login
DefectReturn	0	defect number, original defect number, answerReturn, remarks, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
DefectReview	0	defect number, original defect number (not used), answer (not used), remarks, effective CMVC user ID, real Unix login, verbose flag
	1	defect number, remarks, effective CMVC user ID, verbose flag
	2	same as ExitID 0 without real Unix login

Figure 59 (Part 4 of 15). The CMVC Action, Exit ID and Parameters That Are Passed to User Exit Programs



<b>CMVC Action</b>	<b>ExitID</b>	<b>Parameters Passed to the User Exit Program</b>
DefectSize	0	defect number, original defect number (not used), answer (not used), remarks, effective CMVC user ID, real Unix login, verbose flag
	1	defect number, remarks, effective CMVC user ID, verbose flag
	2	same as ExitID 0 without real Unix login
DefectVerify	0	defect number, original defect number (not used), answer (not used), remarks, effective CMVC user ID, real Unix login, verbose flag
	1	defect number, remarks, effective CMVC user ID, verbose flag
	2	same as ExitID 0 without real Unix login
DefectView	0	defect number, display type, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
<b>Environment</b>		
EnvCreate	0	environment name, release name, tester's name, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
EnvDelete	0	environment name, release name, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
EnvModify	0	environment name, release name, new tester's name, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
<b>Feature</b>		
FeatureAccept	0	feature number, original feature number (not used), remarks, effective CMVC user ID, real Unix login, verbose flag
	1	feature number, remarks, effective CMVC user ID, verbose flag
	2	same as ExitID 0 without real Unix login
FeatureAssign	0	feature number, new component name, new owner's name, remarks, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
FeatureCancel	0	feature number, original feature number (not used), remarks, effective CMVC user ID, real Unix login, verbose flag
	1	feature number, remarks, effective CMVC user ID, verbose flag
	2	same as ExitID 0 without real Unix login
FeatureComment	0	feature number, remarks, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login

Figure 59 (Part 5 of 15). The CMVC Action, Exit ID and Parameters That Are Passed to User Exit Programs

<b>CMVC Action</b>	<b>ExitID</b>	<b>Parameters Passed to the User Exit Program</b>
FeatureDesign	0	feature number, original feature number (not used), remarks, effective CMVC user ID, real Unix login, verbose flag
	1	feature number, remarks, effective CMVC user ID, verbose flag
	2	same as ExitID 0 without real Unix login
FeatureModify	0	feature number, new feature number, prefix, reference, abstract, originator, remarks, configurable field string, effective CMVC user ID, real Unix login, verbose flag
	1	feature number, new feature number, prefix, reference, abstract, originator, remarks, configurable field string, date of last update, effective CMVC user ID, verbose flag
	2	same as ExitID 0 without real Unix login
FeatureOpen	0	component name, prefix, reference, remarks, abstract, configurable field string, feature number, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
FeatureReopen	0	feature number, original feature number (not used), remarks, effective CMVC user ID, real Unix login, verbose flag
	1	feature number, remarks, effective CMVC user ID, verbose flag
	2	same as ExitID 0 without real Unix login
FeatureReturn	0	feature number, original feature number, remarks, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
FeatureReview	0	feature number, original feature number (not used), remarks, effective CMVC user ID, real Unix login, verbose flag
	1	feature number, remarks, effective CMVC user ID, verbose flag
	2	same as ExitID 0 without real Unix login
FeatureSize	0	feature number, original feature number (not used), remarks, effective CMVC user ID, real Unix login, verbose flag
	1	feature number, remarks, effective CMVC user ID, verbose flag
	2	same as ExitID 0 without real Unix login
FeatureView	0	feature number, display type, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
FeatureVerify	0	feature number, original feature number (not used), remarks, effective CMVC user ID, real Unix login, verbose flag
	1	feature number, remarks, effective CMVC user ID, verbose flag
	2	same as ExitID 0 without real Unix login

Figure 59 (Part 6 of 15). The CMVC Action, Exit ID and Parameters That Are Passed to User Exit Programs

CMVC Action	ExitID	Parameters Passed to the User Exit Program
<b>File</b>		
FileAdd	0	file path name, temp file on server, release name, component name, file type, remarks, fileMode, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
FileCheckIn	0	file path name, temp file on server, release name, force flag, remarks, common flag, effective CMVC user ID, real Unix login, verbose flag
	1	file path name, temp file on server, release name, file type, component name, force flag, remarks, common release names, effective CMVC user ID, verbose flag
	2	file path name, temp file on server, release name, file type, component name, new SID, force flag, remarks, common release names, effective CMVC user ID, verbose flag
FileCheckOut	0	file path name, temp file on server, release name, force flag, effective CMVC user ID, real Unix login, verbose flag
	1	file path name, temp file on server, release name, file type, component name, old SID, force flag, effective CMVC user ID, verbose flag
	2	same as ExitID 1
FileDelete	0	file path name, release name, force flag, common flag, effective CMVC user ID, real Unix login, verbose flag
	1	file path name, release name, force flag, common flag, component name, effective CMVC user ID, verbose flag
	2	same as ExitID 1
FileDestroy	0	file path name, release name, effective CMVC user ID, real Unix login, verbose flag
	1	file path name, release name, component name, SID, effective CMVC user ID, verbose flag
	2	same as ExitID 1
FileExtract	0	file path name, temp file on server, release name, nokeys flag, SID, effective CMVC user ID, real Unix login, verbose flag
	1	file path name, temp file on server, release name, nokeys flag, SID, component name, effective CMVC user ID, verbose flag
	2	same as ExitID 1  <b>Note:</b> If the <b>-stdout</b> flag is used for this command, then all messages from the user exit program will be suppressed.
FileLink	0	file path name, release name, new release name, SID, effective CMVC user ID, real Unix login, verbose flag
	1	file path name, release name, new release name, SID, component name, effective CMVC user ID, verbose flag
	2	same as ExitID 1
FileLock	0	file path name, temp file on server, release name, force flag, effective CMVC user ID, real Unix login, verbose flag
	1	file path name, release name, force flag, file type, component name, SID, effective CMVC user ID, verbose flag
	2	same as ExitID 1

Figure 59 (Part 7 of 15). The CMVC Action, Exit ID and Parameters That Are Passed to User Exit Programs

CMVC Action	ExitID	Parameters Passed to the User Exit Program
FileMigrate	0	file path name, temp file on server, release name, component name, SID, fileMode, effective CMVC user ID, real Unix login, verbose flag This action uses the <b>FileAdd</b> action index. If you have a user exit program for <b>FileAdd</b> , it will apply to <b>FileMigrate</b> as well. You cannot specify a user exit program for <b>FileMigrate</b> .
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
FileModify	0	file path name, release name, new component name, new fileMode, configurable field string, effective CMVC user ID, real Unix login, verbose flag
	1	file path name, release name, old component name, new component name, old fileMode, new fileMode, configurable field string, date of last update, effective CMVC user ID, verbose flag
	2	same as ExitID 1 without date of last update
FileRecreate	0	file path name, release name, force flag, common flag, effective CMVC user ID, real Unix login, verbose flag
	1	file path name, release name, force flag, common flag, component name, old dropDate, effective CMVC user ID, verbose flag
	2	same as ExitID 1
FileRename	0	file path name, release name, new file path name, force flag, common flag, effective CMVC user ID, real Unix login, verbose flag
	1	file path name, release name, new file path name, force flag, common flag, component name, effective CMVC user ID, verbose flag
	2	same as ExitID 1
FileUndo	0	file path name, release name, force flag, common flag, effective CMVC user ID, real Unix login, verbose flag
	1	file path name, release name, force flag, common flag, component name, SID, effective CMVC user ID, verbose flag
	2	same as ExitID 1
FileUnlock	0	file path name, release name, effective CMVC user ID, real Unix login, verbose flag
	1	file path name, release name, component name, effective CMVC user ID, verbose flag
	2	same as ExitID 1
FileView	0	file path name, release name, display type, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
<b>Fix</b>		
FixActive	0	defect or feature number, release name, component name, effective CMVC user ID, real Unix login, verbose flag
	1	defect or feature number, release name, component name, type, effective CMVC user ID, verbose flag
	2	same as ExitID 1

Figure 59 (Part 8 of 15). The CMVC Action, Exit ID and Parameters That Are Passed to User Exit Programs

<b>CMVC Action</b>	<b>ExitID</b>	<b>Parameters Passed to the User Exit Program</b>
FixAssign	0	defect or feature number, release name, component name, user ID of the new fix record owner, effective CMVC user ID, real Unix login, verbose flag
	1	defect or feature number, release name, component name, user ID of the new fix record owner, type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
FixComplete	0	defect or feature number, release name, component name, effective CMVC user ID, real Unix login, verbose flag
	1	defect or feature number, release name, component name, type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
FixCreate	0	defect or feature number, release name, component name, user ID of the fix record owner, effective CMVC user ID, real Unix login, verbose flag
	1	defect or feature number, release name, component name, user ID of the fix record owner, type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
FixDelete	0	defect or feature number, release name, component name, effective CMVC user ID, real Unix login, verbose flag
	1	defect or feature number, release name, component name, type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
<b>Host</b>		
HostCreate	0	CMVC user ID, login@host name, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
HostDelete	0	CMVC user ID, login@host name, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
<b>Level</b>		
LevelAssign	0	level name, release name, new owner, effective CMVC user ID, real Unix login, verbose flag
	1	level name, release name, new owner, level state, level type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
LevelCheck	0	level name, release name, long flag, effective CMVC user ID, real Unix login, verbose flag
	1	level name, release name, long flag, level state, level type, effective CMVC user ID, verbose flag
	2	same as ExitID 1

Figure 59 (Part 9 of 15). The CMVC Action, Exit ID and Parameters That Are Passed to User Exit Programs

<b>CMVC Action</b>	<b>ExitID</b>	<b>Parameters Passed to the User Exit Program</b>
LevelComplete	0	level name, release name, effective CMVC user ID, real Unix login, verbose flag
	1	level name, release name, level type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
LevelCommit	0	level name, release name, effective CMVC user ID, real Unix login, verbose flag
	1	level name, release name, level type, effective CMVC user ID, verbose flag
	2	same as Exit ID 1
LevelCreate	0	level name, release name, level type, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
LevelDelete	0	level name, release name, effective CMVC user ID, real Unix login, verbose flag
	1	level name, release name, level state, level type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
LevelExtract	0	level name, release name, root, node, nokeys flag, type of level extract, fmask, dmask, uid, gid, crlf flag (not used), effective CMVC user ID, real Unix login, verbose flag
	1	level name, release name, root, node, nokeys flag, type of level extract, fmask, dmask, uid, gid, crlf flag (not used), level state, level type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
LevelModify	0	level name, new level name, release name, new level type, effective CMVC user ID, real Unix login, verbose flag
	1	level name, new level name, release name, old type, new type, level state, date of last update, effective CMVC user ID, verbose flag
	2	level name, new level name, release name, old type, new type, level state, effective CMVC user ID, verbose flag
LevelView	0	level name, release name, display type, effective CMVC user ID, real Unix login, verbose flag
	1	level name, release name, display type, level state, level type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
<b>LevelMember</b>		
MemberCreate	0	level name, release name, defect or feature number, effective CMVC user ID, real Unix login, verbose flag
	1	level name, release name, defect or feature number, track state, track type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
MemberDelete	0	level name, release name, defect or feature number, effective CMVC user ID, real Unix login, verbose flag
	1	level name, release name, defect or feature number, track state, track type, effective CMVC user ID, verbose flag
	2	same as ExitID 1

Figure 59 (Part 10 of 15). The CMVC Action, Exit ID and Parameters That Are Passed to User Exit Programs

<b>CMVC Action</b>	<b>ExitID</b>	<b>Parameters Passed to the User Exit Program</b>
<b>Notification</b>		
NotifyCreate	0	CMVC user ID, component name, interest group name, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
NotifyDelete	0	CMVC user ID, component name, interest group name, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
<b>Release</b>		
ReleaseCreate	0	release name, component name, new release process, environment name, tester's name, approver's name, description, release owner, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
ReleaseDelete	0	release name, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
ReleaseExtract	0	release name, root, node, nokeys flag, committed flag, date, fmask, dmask, uid, gid, crlf flag, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
ReleaseLink	0	release name, link to release name, committed flag, date, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
ReleaseModify	0	release name, new release name, component name, description, new release process, environment name, tester's name, approver's name, user ID of the new release owner, effective CMVC user ID, real Unix login, verbose flag
	1	release name, new release name, old component name, new component name, old description, new description, old release process, new release process, environment name, tester's name, approver's name, user ID of the old release owner, user ID of the new release owner, date of last update, effective CMVC user ID, verbose flag
	2	same as ExitID 0 without real Unix login
ReleaseRecreate	0	release name, environment name, tester's name, approver's name, effective CMVC user ID, real Unix login, verbose flag
	1	release name, last drop date, effective CMVC user ID, verbose flag
	2	same as ExitID 1
ReleaseView	0	release name, report type, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login

Figure 59 (Part 11 of 15). The CMVC Action, Exit ID and Parameters That Are Passed to User Exit Programs

<b>CMVC Action</b>	<b>ExitID</b>	<b>Parameters Passed to the User Exit Program</b>
<b>Report</b>		
Report	0	view name, report criteria, report type, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
<b>Size</b>		
SizeAccept	0	defect or feature number, component name, release name, size text, effective CMVC user ID, real Unix login, verbose flag
	0	defect or feature number, component name, release name, user ID of the new sizing record owner, effective CMVC user ID, real Unix login, verbose flag
	1	defect or feature number, component name, release name, size text, size type, effective CMVC user ID, verbose flag
SizeAssign	1	defect or feature number, component name, release name, user ID of the new sizing record owner, size type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
	2	same as ExitID 1
	2	same as ExitID 1
SizeCreate	0	defect or feature number, component name, release name, effective CMVC user ID, real Unix login, verbose flag
	1	defect or feature number, component name, release name, size type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
SizeDelete	0	defect or feature number, component name, release name, effective CMVC user ID, real Unix login, verbose flag
	1	defect or feature number, component name, release name, size type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
SizeReject	0	defect or feature number, component name, release name, effective CMVC user ID, real Unix login, verbose flag
	1	defect or feature number, component name, release name, size type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
<b>Test</b>		
TestAbstain	0	defect or feature number, user ID of the test record owner, release name, environment name, effective CMVC user ID, real Unix login, verbose flag
	1	defect or feature number, user ID of the test record owner, release name, environment name, type, effective CMVC user ID, verbose flag
	2	same as ExitID1
TestAccept	0	defect or feature number, user ID of the test record owner, release name, environment name, effective CMVC user ID, real Unix login, verbose flag
	1	defect or feature number, user ID of the test record owner, release name, environment name, type, effective CMVC user ID, verbose flag
	2	same as ExitID 1

Figure 59 (Part 12 of 15). The CMVC Action, Exit ID and Parameters That Are Passed to User Exit Programs



<b>CMVC Action</b>	<b>ExitID</b>	<b>Parameters Passed to the User Exit Program</b>
TestAssign	0	defect or feature number, user ID of the old test record owner, user ID of the new test record owner, release name, environment name, effective CMVC user ID, real Unix login, verbose flag
	1	defect or feature number, user ID of the old test record owner, user ID of the new test record owner, release name, environment name, track type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
TestReject	0	defect or feature number, user ID of the test record owner, release name, environment name, effective CMVC user ID, real Unix login, verbose flag
	1	defect or feature number, user ID of the test record owner, release name, environment name, type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
<b>Track</b>		
TrackAssign	0	release name, defect or feature number, user ID of the new track owner, effective CMVC user ID, real Unix login, verbose flag
	1	release name, defect or feature number, user ID of the new track owner, track type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
TrackCancel	0	release name, defect or feature number, effective CMVC user ID, real Unix login, verbose flag
	1	release name, defect or feature number, track type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
TrackCheck	0	release name, defect or feature number, level, effective CMVC user ID, real Unix login, verbose flag
	1	release name, defect or feature number, level name, track type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
TrackComplete	0	release name, defect or feature number, effective CMVC user ID, real Unix login, verbose flag
	1	release name, defect or feature number, track type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
TrackCommit	0	release name, defect or feature number, effective CMVC user ID, real Unix login, verbose flag
	1	release name, defect or feature number, track type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
TrackCreate	0	release name, defect or feature number, target user ID of the track owner, effective CMVC user ID, real Unix login, verbose flag
	1	release name, defect or feature number, target user ID of the track owner, track type, effective CMVC user ID, verbose flag
	2	same as ExitID 1

Figure 59 (Part 13 of 15). The CMVC Action, Exit ID and Parameters That Are Passed to User Exit Programs

<b>CMVC Action</b>	<b>ExitID</b>	<b>Parameters Passed to the User Exit Program</b>
TrackFix	0	release name, defect or feature number, effective CMVC user ID, real Unix login, verbose flag
	1	release name, defect or feature number, track type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
TrackIntegrate	0	release name, defect or feature number, effective CMVC user ID, real Unix login, verbose flag
	1	release name, defect or feature number, track type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
TrackModify	0	release name, defect or feature number, new target, effective CMVC user ID, real Unix login, verbose flag
	1	release name, defect or feature number, old target, new target, track type, effective CMVC user ID, verbose flag
	2	release name, defect or feature number, new target, track type, effective CMVC user ID, verbose flag
TrackTest	0	release name, defect or feature number, effective CMVC user ID, real Unix login, verbose flag
	1	release name, defect or feature number, track type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
TrackView	0	release name, defect or feature number, long flag, effective CMVC user ID, real Unix login, verbose flag
	1	release name, defect or feature number, long flag, track type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
<b>User</b>		
UserCreate	0	login, user's full name, area, sendmail address, superuser privilege flag, configurable field string, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
UserDelete	0	login, effective CMVC user ID, real Unix login, verbose flag
	1	login, user's full name, effective CMVC user ID, verbose flag
	2	same as ExitID 1
UserModify	0	login, new login, new user's full name, new area, new user's sendmail address, new superuser privilege flag, configurable field string, effective CMVC user ID, real Unix login, verbose flag
	1	login, new login, old user's full name, new user's full name, old area, new area, old sendmail address, new sendmail address, old superuser privilege flag, new superuser privilege flag, configurable field string, date of last update, effective CMVC user ID, verbose flag
	2	same as ExitID 0 without real Unix login

Figure 59 (Part 14 of 15). The CMVC Action, Exit ID and Parameters That Are Passed to User Exit Programs

CMVC Action	ExitID	Parameters Passed to the User Exit Program
UserRecreate	0	login, effective CMVC user ID, real Unix login, verbose flag
	1	login, user's full name, old dropDate, effective CMVC user ID, verbose flag
	2	same as ExitID 1
UserView	0	login, display type, effective CMVC user ID, real Unix login, verbose flag
	1	same as ExitID 0 without real Unix login
	2	same as ExitID 0 without real Unix login
<b>Verify</b>		
VerifyAbstain	0	defect or feature number, user ID of the verification record owner, effective CMVC user ID, real Unix login, verbose flag
	1	defect or feature number, user ID of the verification record owner, type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
VerifyAccept	0	defect or feature number, user ID of the verification record owner, effective CMVC user ID, real Unix login, verbose flag
	1	defect or feature number, user ID of the verification record owner, type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
VerifyAssign	0	defect or feature number, user ID of the old verification record owner, user ID of the new verification record owner, effective CMVC user ID, real Unix login, verbose flag
	1	defect or feature number, user ID of the old verification record owner, user ID of the new verification record owner, type, effective CMVC user ID, verbose flag
	2	same as ExitID 1
VerifyReject	0	defect or feature number, user ID of the verification record owner, effective CMVC user ID, real Unix login, verbose flag
	1	defect or feature number, user ID of the verification record owner, type, effective CMVC user ID, verbose flag
	2	same as ExitID 1

Figure 59 (Part 15 of 15). The CMVC Action, Exit ID and Parameters That Are Passed to User Exit Programs

## User Exit Parameter Definitions

The following list provides definitions for most of the parameters passed to user exit programs. Parameters are listed in alphabetical order. Parameter names are in lower case, except where they are the name of a field in a CMVC database table. For more information on these and other parameters, see the *IBM CMVC Commands Reference*.

<b>abstract</b>	Defect or feature abstract.
<b>answerAccept</b>	Accept answer type.
<b>answerReturn</b>	Return answer type.
<b>approver's name</b>	Approver's CMVC user ID.
<b>area</b>	User's work area or department.

<b>committed flag</b>	This flag is used for ReleaseExtract and ReleaseLink actions to specify whether the user wants the last committed (as opposed to the current) versions of files in the release. 0 is off, 1 means to use the last committed version.
<b>common flag</b>	Indicates whether the file is common with other releases or not. 0 is no and 1 is yes.
<b>common release names</b>	For a common file, this parameter specifies the other releases the file is common with (on the FileCheckIn action). Release names are separated by blanks.
<b>configurable field string</b>	This has the format: <pre> attribute name    content attribute name    content :                 null string </pre> For ExitID 0, the attribute name may appear in abbreviated form, as it is not processed by the CMVC server software.
<b>crlf flag</b>	Reserved for future use.
<b>defectDSR flag</b>	Indicates whether the defect Design/Size/Review subprocess is on or off for a component. A value of yes is on and a value of no is off.
<b>defectVerify flag</b>	Indicates whether the defect verification subprocess is on or off for a component. A value of yes is on and a value of no is off.
<b>description</b>	Description of component or release.
<b>display type</b>	This is used on all <b>View</b> actions. The type of view format requested, where: <pre> <b>0</b>      stanza <b>1</b>      raw <b>2</b>      table <b>3</b>      long <b>4</b>      process </pre>
<b>dmask</b>	Specifies the read, write, and execute directory permissions for the extracted files in octal notation.
<b>effective CMVC user ID</b>	The effective CMVC user ID is the CMVC user ID which initiated the transaction.
<b>featureDSR flag</b>	Indicates whether the feature Design/Size/Review subprocess is on or off for a component. A value of yes is on and a value of no is off.
<b>featureVerify flag</b>	Indicates whether the feature verification subprocess is on or off for a component. A value of yes is on and a value of no is off.

<b>fileMode</b>	<p>This specifies the read, write, and execute permissions of the file. This parameter is defined numerically, in octal notation. The fileMode code is constructed by combing the logical OR of the following values:</p> <table> <tr> <td><b>4000</b></td> <td>setuid</td> </tr> <tr> <td><b>2000</b></td> <td>setgid</td> </tr> <tr> <td><b>0400</b></td> <td>Permits read by owner</td> </tr> <tr> <td><b>0200</b></td> <td>Permits write by owner</td> </tr> <tr> <td><b>0100</b></td> <td>Permits execute or search by owner</td> </tr> <tr> <td><b>0040</b></td> <td>Permits read by group</td> </tr> <tr> <td><b>0020</b></td> <td>Permits write by group</td> </tr> <tr> <td><b>0010</b></td> <td>Permits execute or search by group</td> </tr> <tr> <td><b>0004</b></td> <td>Permits read by all others</td> </tr> <tr> <td><b>0002</b></td> <td>Permits write by all others</td> </tr> <tr> <td><b>0001</b></td> <td>Permits execute or search by all others</td> </tr> </table> <p>For example, 0755 would permit read, write and execute for the owner and read and execute for all others.</p>	<b>4000</b>	setuid	<b>2000</b>	setgid	<b>0400</b>	Permits read by owner	<b>0200</b>	Permits write by owner	<b>0100</b>	Permits execute or search by owner	<b>0040</b>	Permits read by group	<b>0020</b>	Permits write by group	<b>0010</b>	Permits execute or search by group	<b>0004</b>	Permits read by all others	<b>0002</b>	Permits write by all others	<b>0001</b>	Permits execute or search by all others
<b>4000</b>	setuid																						
<b>2000</b>	setgid																						
<b>0400</b>	Permits read by owner																						
<b>0200</b>	Permits write by owner																						
<b>0100</b>	Permits execute or search by owner																						
<b>0040</b>	Permits read by group																						
<b>0020</b>	Permits write by group																						
<b>0010</b>	Permits execute or search by group																						
<b>0004</b>	Permits read by all others																						
<b>0002</b>	Permits write by all others																						
<b>0001</b>	Permits execute or search by all others																						
<b>file type</b>	<p>File type for the FileAdd action will be 0 if binary is not specified during the file creation or 1 if binary is specified during the file creation.</p> <p>File type for the other <b>File</b> actions will be <i>text</i> for text files, <i>binary</i> for binary files, and <i>long</i> for files with a record length greater than 510 characters.</p>																						
<b>fmask</b>	<p>Specifies the read, write, and execute file permissions for the extracted files in octal notation. See the <i>IBM CMVC Commands Reference</i> for details.</p>																						
<b>force flag</b>	<p>Indicates whether the force option was chosen on <b>File</b> actions (0 indicates no and 1 indicates yes). The force option is used to force a break between common files when using FileLock, FileCheckOut, FileCheckIn, FileDelete, FileRecreate, FileRename and FileUndo actions.</p>																						
<b>gid</b>	<p>Specifies ownership of extracted files by identifying the internal number that uniquely identifies the group to the system. See the <i>IBM CMVC Commands Reference</i> for details.</p>																						
<b>level state</b>	<p>Values can be working, integrate, commit or complete.</p>																						
<b>level type</b>	<p>Specified by the user when a level is created, for example, development, production or prototype.</p>																						
<b>long flag</b>	<p>Indicates whether the long flag is specified or not specified. 0 is not specified and 1 is specified. The <b>-long</b> flag is available on some of the <b>View</b> actions and is used to display more detailed information of the object being viewed. The <b>-long</b> flag on the LevelCheck action will display details about prerequisites and corequisites.</p>																						
<b>node</b>	<p>Specifies a remote host on which to place the extracted file tree.</p>																						

<b>nokeys flag</b>	For extract actions, indicates whether you want to substitute assigned values in place of keywords imbedded in the extracted files. 0 means not to substitute assigned values, 1 means to substitute assigned values.
<b>originator</b>	The CMVC user ID of the user who opens the defect or feature.
<b>prefix</b>	Defect or feature prefix.
<b>prime track name</b>	Prime corequisite track name.
<b>prime track type</b>	Prime corequisite track type (defect or feature).
<b>real Unix login</b>	The user's Unix login on the client workstation.
<b>reference</b>	Defect or feature reference.
<b>remarks</b>	For <b>Defect</b> or <b>Feature</b> actions, this will be defect remarks or feature remarks. For <b>File</b> actions, this will be file remarks added when a new version is created.
<b>report criteria</b>	The criteria entered as the <b>-where</b> clause for the <b>Report</b> action.
<b>report type</b>	The type of report format requested, where: <ul style="list-style-type: none"> <li><b>0</b> stanza</li> <li><b>1</b> raw</li> <li><b>2</b> table</li> <li><b>3</b> long</li> <li><b>4</b> process</li> </ul> <p>User exit messages are not displayed if the <b>-raw</b> format is selected.</p>
<b>root</b>	This is the specified directory on the designated host where the extracted file tree is to be placed.
<b>second track name</b>	Second corequisite track name.
<b>second track type</b>	Second corequisite track type (defect or feature).
<b>severity</b>	Defect severity level.
<b>SID</b>	File version number.
<b>superuser privilege flag</b>	A value of yes is on and a value of no is off.
<b>temp file on server</b>	For some <b>File</b> commands, the contents of the file on the client are copied to a temporary file on the server. This parameter is the file name for the temporary file on the server.
<b>track state</b>	The value can be approve, fix, integrate, commit, test or complete.
<b>track type</b>	The value can be defect or feature.
<b>type of level extract</b>	Indicates whether this is a full or delta <b>Level -extract</b> . 0 is delta, 1 is full.
<b>uid</b>	Specifies ownership of extracted files by identifying the internal number that uniquely identifies the user to the system.

**verbose flag**

Specifies that you want to see a confirmation message after you issue this command. 0 is off, 1 is on. The user exit program can use this flag to include confirmation or status messages only when the verbose flag is on.

**view name**

The name of the View (for example, FileView) which is being reported on.





---

# Glossary

This glossary contains definitions of data processing terms that might be unfamiliar to you. It includes terms and definitions from the *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

## A

**access list.** A set of objects that controls access to development data. Each object consists of a component, a user, and the authority that the user is granted or is restricted from in that component. See also *authority*, *granted authority*, and *restricted authority*.

**action.** A task performed by the CMVC server and requested by a CMVC client. A CMVC action is the same as issuing one CMVC command.

**approver.** A user who has the authority to accept, reject, or abstain changes within a specific release.

**approver list.** A list of user IDs attached to a release, representing the users who must review file changes that are required to resolve a defect or implement a feature in that release.

**attribute.** Information contained in a field that is accessible to the user. CMVC enables family administrators to customize defect, feature, user, and file tables by adding new attributes.

**authority.** The right to access development objects and perform CMVC commands. See also *access list*, *base authority*, *explicit authority*, *granted authority*, *implicit authority*, *restricted authority*, and *superuser privilege*.

## B

**base authority.** The set of actions granted to a user when a user ID is created within a CMVC family. See also *authority*. Contrast with *implicit authority* and *explicit authority*.

**Broadcast Message Server (BMS).** A facility that coordinates the SDE WorkBench/6000 or HP SoftBench tools. Messages from tools are sent to the Broadcast Message Server, which routes the messages to other tools.

## C

**change control.** The process of limiting and auditing changes to files through the mechanism of checking files in and out of a central, controlled, storage location. Change control for individual releases can be integrated with problem tracking by specifying a process for the release that includes the tracking subprocess.

**check in.** The return of a CMVC file to version control.

**check out.** The retrieval of a revision of a CMVC file from version control.

**child component.** Any component in a CMVC family, except the root component, that is created in reference to an existing component. The existing component is the parent component, and the new component is the child component. A parent component can have more than one child component, and a child component can have more than one parent component. See also *component* and *parent component*.

**client.** A functional unit that receives shared services from a server. Contrast with *server*.

**CMVC client.** A workstation that connects to the CMVC server by a TCP/IP connection and that is running the CMVC client software.

**CMVC file.** A file that is stored by the CMVC server and retrieved by a path name and release. See also *file*, *common file*, and *shared file*.

**CMVC server.** A workstation running the CMVC server software.

**command.** A request to perform an operation or run a program from the command line interface. In CMVC, a command consists of the command name, one action flag, and zero or more attribute flags.

**common file.** A file that is shared by two or more releases, and the same version of the file is the current version for those releases. See also *shared file*.

**component.** A CMVC object that organizes project data into structured groups, and controls configuration management properties. Component owners can control access to development data and configure notification about CMVC actions. Components exist in a parent-child hierarchy, with descendent components inheriting access and notification information from ancestor components. See also *access list* and *notification list*.

**configuration management.** The process of identifying, managing, and controlling software modules as they change over time.

**current working directory.** (1) The directory that is the starting point for relative path names. (2) The directory in which you are working.

## D

**daemon.** A program that runs unattended to perform a standard service. Some daemons are triggered automatically to perform their task; others operate periodically.

**daemon process.** A process begun by the root user or by the root shell that can be stopped only by the root user. Daemon processes generally provide services that must be available at all times, such as sending data to a printer.

**database.** A organized collection of data that can be accessed and operated upon by a data processing system for a specific purpose.

**default.** A value that is used when an alternative is not specified by the user.

**defect.** A CMVC object used to formally report a problem. The user who opens a defect is the defect originator.

## E

**end user.** See *user*.

**environment.** A user-defined testing domain for a particular release. Also used as a defect field, in which case, it is the environment where the problem occurred.

**environment list.** A CMVC object used to specify environments in which a release should be tested. A list of environment-user ID pairs attached to a release, representing the user responsible for testing each environment. Only one tester can be identified for an environment.

**explicit authority.** The ability to perform an action against a CMVC object because you have been granted the authority to perform that action. Contrast with *implicit authority* and *base authority*.

## F

**family.** A logical organization of related development data. A single CMVC server can support multiple families. The data in one family cannot be accessed from another family.

**family administrator.** A user who is responsible for all nonsystem-related tasks for one or more CMVC families, such as planning, configuring, and maintaining the CMVC environment and managing user access to those families.

**feature.** A CMVC object used to formally request and record information about a functional addition or enhancement. The user who opens a feature is the feature originator.

**file.** A collection of data that is stored by the CMVC server and retrieved by a path name. Any text or binary file used in a development project can be created as a CMVC file. Examples include source code, executable programs, documentation, and test cases. See *common file* and *shared file*.

**fix record.** A status record that is associated with a track and that is used to monitor the phases of change within each component that is affected by a defect or feature for a specific release.

## G

**graphical user interface (GUI).** The OSF/Motif-based CMVC graphical user interface program.

## H

**host.** A host node, host computer, or host system.

**host list.** A list associated with each CMVC user ID that indicates the client hosts that can access the CMVC server and act on behalf of the CMVC user. The CMVC server uses the list to authenticate the identity of a CMVC client when the CMVC server receives a CMVC command. Each entry consists of a login, a CMVC user ID, and a host name.

**host name.** The identifier associated with the host computer.

## I

**implicit authority.** The ability to perform an action on a CMVC object without being granted explicit authority. This authority is implicitly granted through inheritance or object ownership. Contrast with *base authority* and *explicit authority*.

**import.** Bring selected items to a field in a dialog box from a matching main CMVC GUI window.

**inheritance.** The passing of configuration management properties from parent to child component. The configuration management properties that are inherited are access and notification. Inheritance within each CMVC family or component hierarchy is cumulative.

**interest group.** The list of actions that trigger notification to the user IDs associated with those actions listed in the notification list.

## L

**level.** A collection of tracks that represent a set of changed files within a release. Levels are only associated with releases whose processes include the track and level subprocesses.

**level member.** A track that is added to a level.

**lock.** An action that prevents editing access to a file stored in the CMVC development environment so that only one user can change a file at a time.

**login.** User identifier.

## M

**megabyte.** 1 048 576 bytes.

## N

**NetLS.** Network License System.

**Network File System (NFS).** A program that enables you to share files with other computers in one or more networks over a variety of machine types and operating systems.

**Network License System (NetLS).** A program that controls the number of users who can simultaneously access CMVC.

**notification list.** A CMVC object allowing component owners to configure notification. A list of user ID-interest group pairs attached to a component, designating users and the corresponding notification interest they are being granted for all objects managed by this component or any of its descendants.

## O

**originator.** The user who opens a defect or feature and is responsible for verifying the outcome of the defect or feature on a verification record. This responsibility can be reassigned.

**owner.** The user who is responsible for a CMVC object within a CMVC family, either because the user created the object or was assigned ownership of the object.

## P

**parent component.** All components in each CMVC family, except the root component, are created in reference to an existing component. The existing component is the parent component. See also *child component* and *component*.

**path name.** The name of the file under CMVC control. A path name can be a set of directory names and a base name or just a base name. It must be unique within the release that groups the files.

**problem tracking.** The process of tracking all reported defects through to resolution and all proposed features through to implementation.

**process.** A combination of CMVC subprocesses, configured by the family administrator, that controls the general movement of CMVC objects (defects, features, tracks, and levels) from state to state within a component or release. See also *subprocess* and *state*.

## Q

**query.** A structured request for information from a database, for example, a search for all defects that are in the open state. See also *default query* and *search*.

## R

**release.** A CMVC object defined by a user that groups all the files that must be built, tested, and distributed as a single entity.

**restricted authority.** The restriction of a user's ability to perform certain actions at a specific component. Authority can be restricted by the superuser, the component owner, or a user with AccessRestrict authority. See also *authority*.

**root component.** The initial component that is created when a CMVC family is configured. All components in a CMVC family are descendants of the root component. Only the root component has no parent component.

See also *component*, *child component*, and *parent component*.

## S

**search.** To scan one or more data elements of a set in a database to find elements that have certain properties.

**server.** A workstation that performs a service for another workstation.

**shared file.** A file that is shared between two or more releases. See also *common file*.

**shell script.** A series of commands combined in a file that carry out a function when the file is run.

**sizing record.** A status record created for each component-release pair affected by a proposed defect or feature. The sizing record owner must indicate whether the defect or feature affects the specified component-release pair and the approximate amount of work needed to resolve the defect or implement the feature within the specified component-release pair.

**SMIT.** System Management Interface Tool.

**state.** Tracks, levels, features, and defects move through various states during their life cycles. The state of an object determines the actions that can be performed on it. See also *process* and *subprocess*.

**subprocess.** CMVC subprocesses govern the state changes for CMVC objects. The design, size, review (DSR) and verify subprocesses are configured for component processes. The track, approve, fix, level, and test subprocesses are configured for release processes. See also *process* and *state*.

**superuser privilege.** This privilege allows a user to perform any action available in the CMVC family. **Note:** Superuser privilege is internal to CMVC and not related to operating system superuser authority.

**system administrator.** A user who is responsible for all system-related tasks involving the CMVC server, such as installing, maintaining, and backing up the CMVC server and the relational database it uses.

**System Management Interface Tool (SMIT).** An AIX tool used to perform system administration tasks, such as installing software or creating logins for new users.

## T

**TCP/IP.** A set of communications protocols that support peer-to-peer connectivity functions for both local and wide area networks.

**tester.** A user responsible for testing the resolution of a defect or the implementation of a feature for a specific level of a release and recording the results on a test record.

**test record.** A status record used to record the outcome of an environment test performed for a resolved defect or an implemented feature in a specific level of a release.

**tool.** An SDE WorkBench/6000 or HP SoftBench application.

**track.** A CMVC object created to monitor the progress of changes made to files within a release to resolve a specific defect or to implement a specific feature.

**track subprocess.** An attribute of a CMVC release process that specifies that the change control process for that release will be integrated with the problem tracking process.

**Transmission Control Protocol/Internet Protocol.** See *TCP/IP*.

## U

**user.** A person with an active CMVC user ID and access to one or more CMVC families.

**user exit.** A user exit allows CMVC to call a user-defined program during the processing of CMVC transactions. User exits provide a means by which users can specify additional actions that should be performed before completing or proceeding with a CMVC action.

**user ID.** The identifier assigned by the system administrator to each CMVC user.

## V

**verification record.** A status record that the originator of a defect or a feature must mark before the defect or feature can move to the closed state. Originators use verification records to verify the resolution or implementation of the defect or feature they opened.

**version control.** The storage of multiple versions of a single file along with information about each version.

**view.** An alternative and temporary representation of data from one or more tables.



---

# Index

## A

- access control 67
- access list 67, 73
- actions
  - grouping 68, 74
  - in user exit programs 88
  - restricting access to 67
- Activity Monitor 105
- administration
  - family 2
  - system 2
- administrator
  - family 2
  - system 2
- age shell script 103
- aging defects/features 103
- AIX operating system
  - Base Application Development Toolkit 121
  - Base Operating System 8, 14
- answer codes 78
- approver list 65
- archiving
  - level 144
  - limitations 146
  - prerequisites 144, 145
  - release 145
- audit log
  - format 116
  - managing 101, 115
- authority
  - base 67
  - controlling 67
  - explicit 67
  - groups
    - creating 68
    - worksheet 209
  - implicit 67
  - restricted 67
  - table 70
- authority.ld
  - copying 34, 190
  - editing 69
  - migration from CMVC V1R1 190
- authority.log 43, 44, 70
- automatic notification 73, 102

## B

- backup
  - database 137, 141
  - file system 137, 141

- backup copy 13, 185
- bad.authority 43, 70
- bad.cfgcomproc 43, 65
- bad.cfgrelproc 43, 65
- bad.config 43
- bad.interest 43, 76
- base authority 67
- Broadcast Message Server (BMS) xii

## C

- cfgcomproc.ld
  - copying 34
  - editing 63, 192
  - migration from CMVC V1R1 190
- cfgcomproc.log 43, 44, 65
- cfgrelproc.ld
  - copying 34
  - editing 63, 192
  - migration from CMVC V1R1 190
- cfgrelproc.log 43, 44, 65
- change control 61
- chauth 70, 192
- chcfg 84
- chcomproc 64, 192
- check in 27
- check out 27
- chfield 51, 190
- chintr 75, 192
- chrelproc 64, 192
- client
  - description 1
  - installation 15
- client-server model 1
- CLIENT\_HOSTNAME 35
- CLIENT\_LOGIN 35
- CMVC
  - access 125
  - Activity Monitor 105
  - backup recommendations 137
  - files 123
  - recovery 140
  - roles 2
  - system 1
  - tasks 2
  - user interfaces 2
- CMVC client 1
- CMVC daemons
  - maintenance mode 96
  - starting 95
  - stopping 98

- CMVC families
  - archiving and restoring 141
  - planning 31
  - sample sizes 31
- CMVC server
  - backup 137
  - configuration 32
  - installation 13
  - migration from ORACLE6 to ORACLE7 195
  - migration to CMVC Version 2.3 179
  - packaging 13
  - requirements 7
  - tuning 105
- CMVC\_BINARY\_THRESHOLD 39
- CMVC\_HOME 35
- CMVC\_SUPERUSER 35
- CMVC\_VCBIN 35
- CMVC\_VCLOGIN 35
- CMVC\_VCTYPE 35, 172
- cmvcarchive 147
- cmvcd daemon
  - Process ID 106, 116
  - running in maintenance mode 96
  - starting 95
  - stopping 98
- cmvclog.cleanu shell script 115
- cmvcrestore 149
- command 23
- command line 2
- common file 124, 130
- component
  - planning 59, 123
  - processes 60
  - root 64
- config.ld
  - copying 34
  - editing 83
- config.log 43, 44
- configurable fields
  - creating 51
  - defaults 48
  - Defect table 50
  - Feature table 49
  - properties 47, 51
  - reports 55
  - updating 54
- configurable processes
  - component 60
  - conditions 65
  - procedure 63
  - release 61
- configuration database table 77
- conversion utilities 180
- converting from earlier versions 180
- creating index 192

- cron daemon 115
- crontab 116
- current directory 132
- current working directory 28, 131

## D

- daemons
  - cmvcd 95
  - cron 103, 115
  - glbd 20, 98
  - llbd 20, 98
  - netlsd 19, 20, 98
  - notifyd 97
  - nrglbd 20
  - sendmail 97, 101
  - syslog 112, 151
- database
  - backup 138
  - synchronization 23, 138
- DB2\_DBPATH 36
- DB2\_HOME 36
- DB2\_PASS 36
- DB2/6000
  - backup 139
- DB2INSTANCE 36
- dbConvert.v1r1m1 utility 181
- dbConvert.v2r1 shell script 185
- defect
  - aging 103
  - description 60
  - symptoms 78
- directory
  - current 132
  - current working 28, 131
- DSQUERY 38

## E

- end user 3
- environment list 66
- environment variable
  - CLIENT\_HOSTNAME 35
  - CLIENT\_LOGIN 35
  - CMVC\_BINARY\_THRESHOLD 39
  - CMVC\_HOME 35
  - CMVC\_SUPERUSER 35
  - CMVC\_VCBIN 35
  - CMVC\_VCLOGIN 35
  - CMVC\_VCTYPE 35, 172
  - DB2\_DBPATH 36
  - DB2\_HOME 36
  - DB2\_PASS 36
  - DB2INSTANCE 36
  - DSQUERY 38
  - INFORMIX\_DBSP 39, 183



environment variable (*continued*)  
INFORMIXDIR 37  
ORACLE\_DBA 36  
ORACLE\_HOME 36  
ORACLE\_NDXSP 39, 183  
ORACLE\_PASS 36  
ORACLE\_SID 36  
ORACLE\_TBLSP 39, 183  
PATH 35, 87  
SYBASE 37  
SYBASE\_DBDEV 40, 183  
SYBASE\_LOGDEV 40, 183  
SYBASE\_PASS 37  
SYBASE\_SA\_PASS 37

error messages  
  in audit log 116  
  list 151  
errors 112  
etc/hosts 33  
etc/services 33  
explicit  
  authority 67  
  notification 73

## F

family  
  account 33  
  configuration 31  
  planning 31, 59  
family administrator 2  
feature  
  aging 103  
  description 60  
file  
  binary 24  
  check in 27  
  check out 27  
  common 124, 130  
  naming 110  
  obsolete 124  
  owner 124  
  shared 130  
  version labels 24  
  versioning 23

## G

graphical user interface (GUI) xii, 2  
  UNIX platforms xii

## H

hardware requirements 7  
highlighting style xii

history of CMVC 179  
home directory 33  
host 107  
host list 145  
HP-UX operating system 10

## I

implicit authority 67  
indexes  
  converting from earlier versions 180  
INFORMIX  
  backup 139  
  INFORMIX-SQL 8, 10  
INFORMIX\_DBSP 39, 183  
INFORMIXDIR 37  
inheritance 123  
installation  
  client 15  
  server  
    CD-ROM 16  
    tape 15  
integrated development environment xii  
interest groups  
  configuring 74  
  grouping actions 74  
  notification 73  
  worksheet 215  
interest table 75  
interest.ld  
  copying 34, 190  
  editing 75  
  migration from CMVC V1R1 190  
interest.log 43, 44, 76  
interface  
  command line 2  
  graphical user 2  
Internet Protocol (IP) 8  
INTERSOLV PVCS Version Manager  
  See PVCS

## L

level  
  archiving 144  
  committed 111  
  map file 112  
  maps 111  
  member 144  
leveltype 78  
lock 109  
login 34

## M

- mail
  - addressing 102
  - queue-processing 102
  - recommendations 101
  - returned 102
- maintenance
  - files 124
  - ongoing 101
- maps directory 111
- migrating
  - tasks 184
- migration
  - pre-migration tasks 181
  - utilities 180
- migration from CMVC V1R1
  - updating processes 66
- mkdb 34
- mkfamily 41
- monitor 105

## N

- Network File System (NFS) 8, 14
- Network License System (NetLS)
  - daemons 19
  - installation from CD-ROM 17, 18
  - installation from tape 15, 16
  - obtaining the password 7
  - packaging 13
  - password 20
- NFS
  - See Network File System (NFS)
- notification
  - automatic 73, 102
  - daemon 97
  - explicit 73
  - list 73
- notifyd daemon
  - starting 97
  - stopping 98

## O

- ongoing maintenance 101
- ORACLE
  - backup 139
  - SQL\*Loader 43
  - SQL\*Plus 8
  - Transaction Processing Options (TPO) 8, 9, 11
- ORACLE\_DBA 36
- ORACLE\_HOME 36
- ORACLE\_NDXSP 39, 183
- ORACLE\_PASS 36

- ORACLE\_SID 36
- ORACLE\_TBLSP 183
- ORACLE\_TBSLSP 39
- originator 102
- owner 60, 67

## P

- paging 104
- PATH 35, 87
- path name 90, 129
- phases 78
- planning 60
- pre-migration tasks 181
- prefixes 78
- priority 78
- problem tracking 78
- process
  - component 60
  - release 61
  - root 64
  - track 61
- process ID 98, 107
- profile 34, 139
- publications
  - Network License System (NetLS) 19
  - related xii
- PVCS
  - commands 24
  - compression 27
  - configuration parameters 27
  - journal 28
  - keywords 27
  - license administration database 25
  - master configuration file 25
  - registering users 25
  - semaphores 27
  - translate 28
  - vconfig 25
  - Version Manager
    - as version control system 8
    - installation 24
    - resetting 24
    - working directory 28

## Q

- query 109

## R

- recovery
  - database 140
  - file system 140
- relational database
  - backup 137

- relational database (*continued*)
  - DB2/6000 8
  - IBM Database 2 AIX/6000 8
  - INFORMIX 8, 9, 11
  - migration from CMVC V1R1 183
  - ORACLE 8
  - SYBASE 8
- release
  - archiving 145
  - planning 59, 123
  - processes 61
- requirements
  - hardware 7
  - software 7
- resetAge shell script 104
- restoring
  - limitations 146
  - prerequisites 146
- restricted authority 67
- rmdb 46
- rmfamily 45
- root component 64

## S

- SCCS
  - as version control system 8
  - commands 24
  - delta versions 121
  - File Import
    - advantages 122
    - choosing files 124
    - disadvantages 122
    - requirements 125
    - stages 133
  - File Migrate
    - advantages 121
    - disadvantages 122
    - requirements 125
    - stages 125
  - file.import 134
  - file.migrate 131
  - Fileimport 133, 134
  - Filemap 126, 133
  - Filemigrate 130
  - files
    - bringing under CMVC control 121
    - owners 123, 124
  - header flags 123
  - identifiers 121, 127
  - keywords 123
  - map file 128
  - xecit 131, 132
- search 47
- sendmail 97, 101

- server
  - See CMVC server
- severity 78
- shared file 130
- shared memory 109
- sizing record 145
- SMIT
  - See System Management Interface Tool (SMIT)
- software requirements 7
- Solaris operating system 11
- Source Code Control System
  - See SCCS
- SQL\*Loader 43
- SQL\*Plus 8, 9, 10, 11
- state xii
- subprocess 60
- subscribers 73
- SunOS operating system 9
- superuser privilege 68
- SYBASE (environment variable) 37
- SYBASE SQL 8
- SYBASE\_DBDEV 40, 183
- SYBASE\_LOGDEV 40, 183
- SYBASE\_PASS 37
- SYBASE\_SA\_PASS 37
- synchronization
  - correcting with resetAge 103
  - database and vc tree 138
- syslog 113
- syslogd 112
- system administrator 2
- System Management Interface Tool (SMIT) 17

## T

- tables
  - converting from earlier versions 180
- track 61
- track subprocess
  - fix state 125
  - monitoring file changes 61
- Transaction Processing Options (TPO) 8
- Transmission Control Protocol (TCP) 8

## U

- user 2
- user exit
  - guidelines 88
  - parameters 221

## V

- vc tree 110
- vcPath 110

- version control
  - configuration 23
  - pathfinder tool 110
- versions of CMVC 179
- view 43
- views
  - converting from earlier versions 180



---

# Please Tell Us What You Think!

IBM Configuration Management Version Control  
Server Administration and Installation  
Version 2 Release 3  
Publication No. SC09-1631-02

We hope you found this book useful and informative. If you like what we've done, please let us know; if not, please tell us why. We'll use your comments to make the book better.

Please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number to receive a reply.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate, without obligation.

- To send comments by mail or fax, use the form titled "What Do You Think?" on the following page.

If you're mailing from a country other than the United States, you can give the form to the local IBM branch office or IBM representative for postage-paid handling.

To fax the form, use this number: (919) 469-7718.

- To send comments electronically, use one of the following network IDs:

**IBM Mail Exchange    USIB5DNQ at IBMMAIL**  
**Internet                KFRYE@CARVM3.VNET.IBM.COM**


Thank you! Your comments help us make the information more useful for you.

# What Do You Think?

**IBM Configuration Management Version Control  
Server Administration and Installation  
Version 2 Release 3  
Publication No. SC09-1631-02**

We're in business to satisfy you. If we're succeeding, please tell us; if not, let us know how we can do better.

## Overall, how satisfied are you with this book?

						
	<b>Very Satisfied</b>	<b>Satisfied</b>	<b>Neither Satisfied nor Dissatisfied</b>	<b>Dissatisfied</b>	<b>Very Dissatisfied</b>	<b>No Opinion</b>
<b>Overall satisfaction</b>						

## How satisfied are you that the information in this book is:

<b>Accurate</b>						
<b>Complete</b>						
<b>Easy to find</b>						
<b>Easy to understand</b>						
<b>Well organized</b>						
<b>Applicable to your tasks</b>						

## Please tell us how we can improve this book:

---



---



---



---



---



---



---



---

May we contact you to discuss your responses?  Yes  No ☎ \_\_\_\_\_

Fax \_\_\_\_\_

Note that IBM may use or distribute the responses to this form without obligation.

\_\_\_\_\_  
Name

\_\_\_\_\_  
Address

\_\_\_\_\_  
Company or Organization

\_\_\_\_\_

\_\_\_\_\_  
Phone No.

\_\_\_\_\_



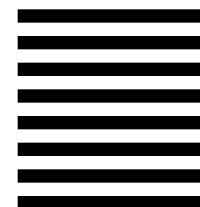
Fold and Tape

Please do not staple

Fold and Tape



NO POSTAGE  
NECESSARY  
IF MAILED IN THE  
UNITED STATES



# BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation  
Information Development  
Department T45  
PO Box 60000  
Cary, NC 27512-9968



Fold and Tape

Please do not staple

Fold and Tape







Program Number: 5622-063  
5765-202  
5765-207  
5765-397

Printed in U.S.A.

SC09-1631-02

