IBM Configuration Management Version Control

SC09-1597-01

User's Reference

Version 2 Release 2

IBM Configuration Management Version Control

SC09-1597-01

User's Reference

Version 2 Release 2

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page vii.

Second Edition (Dec 1993)

This edition applies to Version 2, Release 2, Modification Level 0, of IBM Configuration Management Version Control/6000 (Program 5765-207), IBM Configuration Management Version Control for HP systems (Program 5765-202), IBM Configuration Management Version Control for Sun systems (Program 5622-063) 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 through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

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

IBM Canada Ltd. Laboratory
Information Development
2G/345/1150/TOR
1150 Eglinton Avenue East
North York, Ontario, Canada. M3C 1H7

You can also send your comments by facsimile (attention: RCF Coordinator), or you can send your comments electronically to IBM. See “Communicating Your Comments to IBM” for a description of the methods. This page immediately precedes the Readers’ Comment Form at the back of this publication.

Contents

Notices

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 licensed program in this publication is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent product, program or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

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 Commercial Relations, IBM Corporation, Purchase, NY 10577.

Trademarks and Service Marks

The following terms, denoted by an asterisk (*), used in this publication, are trademarks or service marks of International Business Machines Corporation in the United States or other countries:

AIX	AIXwindows	DB2/6000
IBM	IBMLink	Operating System/2
OS/2	PROFS	PowerServer
RISC System/6000		

The following terms, denoted by a double asterisk (**), used in this publication, are trademarks of other companies as follows:

HP	Hewlett-Packard Company
HP-UX	Hewlett-Packard Company
INFORMIX	Informix Software, Inc.
NetLS	Apollo Computer, Inc.
Network File System, NFS,	Sun Microsystems Inc.
Network License System	Apollo Computer, Inc.
ORACLE, SQL*Plus, SQL*Loader	Oracle Corporation
OSF/Motif	Open Software Foundation, Inc.
PVCS Version Manager	INTERSOLV, Inc.
SoftBench	Hewlett-Packard Company
SPARCserver	SPARC International, Inc.
Sun	Sun Microsystems Inc.
SunOS	Sun Microsystems Inc.
SYBASE	Sybase, Inc.
SYBASE SQL Server,	Sybase, Inc.
SYBASE Open Client DB-Library/C	Sybase, Inc.
UNIX	Unix System Laboratories, Inc.

About This Book

This book is part of the documentation library supporting the IBM* Configuration Management Version Control (CMVC) licensed programs. CMVC consists of a CMVC *server* and CMVC *clients*.

This book provides reference information you require to use CMVC and describes the *Broadcast Message Server* messages supported by the message-integrated CMVC.

Who Should Read This Book

System administrators responsible for installing, customizing, and maintaining CMVC, *users* of the message-integrated CMVC, and programmers who want to write programs for use with CMVC or application program interfaces to SDE WorkBench/6000, HP** SoftBench**, or HP SoftBench for Sun** *tools* should read this book.

What You Should Know

You should be familiar with CMVC. The *IBM CMVC Concepts* book introduces the fundamentals of the *configuration management, version control, change control,* and *problem tracking* features in the CMVC licensed programs. This book also defines the concepts that are the foundation of CMVC *actions* and establishes their interrelationships.

How This Book Is Organized

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

- Part 1, "CMVC Tables and Listings" on page 1 contains supplementary tables and listings that support the CMVC user. This part contains the following chapters:
  - Chapter 1, "Shipped Access Authority Groups" on page 3 lists the CMVC *authority* groups and the CMVC actions available to users within each group.
  - Chapter 2, "Shipped Interest Notification Groups" on page 9 lists the CMVC interest notification groups and the CMVC actions available to users within each group.
  - Chapter 3, "Shipped CMVC Processes" on page 15 lists the CMVC *processes* and the *subprocesses* included within each process.
  - Chapter 4, "Authority and Notification for CMVC Actions" on page 17 lists the CMVC actions, the required explicit and *implicit authority* to perform an action, and the users notified when an action is performed.
  - Chapter 5, "Shipped Configuration Table Values" on page 29 lists the configuration table values.
  - Chapter 6, "CMVC Views for Queries" on page 33 lists the field names, their data type and a brief description for all CMVC *views*.

- Chapter 7, “CMVC Tables for Queries” on page 53 lists the field names, their data type and a brief description for all CMVC tables.
- Chapter 8, “Sample Shell Scripts, Tutorial, and Demonstration” on page 63 lists sample shell scripts provided with CMVC.
- Chapter 9, “Supported Keywords” on page 69 lists the PVCS Version Manager** and SCCS keywords supported by CMVC.
- Chapter 10, “State Diagrams” on page 71 contains state diagrams that show the life cycle of *defects* and *features*, and the interaction between *tracks* and *levels*.

- Part 2, “The Message Interface” on page 77 describes how the message-integrated CMVC uses the Broadcast Message Server on SDE WorkBench/6000, HP SoftBench, and HP SoftBench for Sun tools to communicate with other development tools. This part contains the following chapters:
  - Chapter 11, “The Broadcast Message Server” on page 79 describes the Broadcast Message Server and how it enables the message-integrated CMVC to work with the other SDE WorkBench/6000, HP SoftBench, or HP SoftBench for Sun tools. It also describes the general BMS messages that the message-integrated CMVC supports.
  - Chapter 12, “The CM Class Messages” on page 89 describes the general CM tool class messages supported by the message-integrated CMVC.
  - Chapter 13, “Tool-Specific CM Messages” on page 99 describes the tool-specific messages supported by the message-integrated CMVC.

Highlighting Conventions

The following highlighting conventions are used in this book:

Bold *Commands*, flags, *files*, directories, and other items predefined by CMVC appear in bold. Valid abbreviations for commands are also in bold.

Italic Arguments or options whose names or values must be supplied by you appear in italics. Italics are also used for emphasis, for the first occurrence in text of terms that appear in the glossary, and for titles of books.

`Monotype` Examples of specific data values, examples of text that you might see displayed, messages, menu items that initiate an action or information that you should type appear in monotype.

Terminology

In this book, SoftBench is used to refer to HP SoftBench and HP SoftBench for Sun products.

Related Publications

The following books contain additional information about CMVC and are shipped with this product.

- *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 Server Administration and Installation*, SC09-1631 contains detailed information for planning, installing, customizing, operating, and maintaining the CMVC server.
- *IBM CMVC User's Guide*, SC09-1634, describes all CMVC *actions* as implemented in the *graphical user interface (GUI)* on the AIX, Sun-OS, and HP-UX platforms.
- *IBM CMVC Commands Reference*, SC09-1635, describes all CMVC commands, their syntax and use, as implemented in the command-line interface.
- *IBM CMVC Client Installation and Configuration*, SC09-1596, contains detailed information needed to install and configure the CMVC client on *(GUI)* on the AIX*, Sun-OS**, and HP-UX** platforms.
- *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.

These two books are shipped with the OS/2 workstation client for CMVC, and can be ordered separately:

- *IBM CMVC Client/2 Getting Started*, SC09-1599, contains detailed information about installing and configuring the OS/2 workstation client for CMVC.
- *IBM CMVC Client/2 User's Guide*, SC09-1783, contains step-by-step information on how to use the graphical user interface for the OS/2 workstation client.

You may need to refer to your operating system or relational database documentation while installing or using CMVC.

Part 1. CMVC Tables and Listings

Chapter 1. Shipped Access Authority Groups

Figure 1 lists the authority groups supplied by IBM. Each authority group consists of actions normally performed by a particular type of user. Your *family administrator* can modify these groups or create new ones to reflect the needs of your organization. Each authority group has an x under it to identify the specific actions included in that group.

CMVC Action	Shipped Access Authority Groups									
	general	**manager**	**developer**	**developer+**	**writer**	**writer+**	**builder**	**releaselead**	**componentlead**	**projectlead**
AccessCreate									x	x
AccessDelete									x	x
AccessRestrict									x	x
ApprovalAbstain								x		x
ApprovalAccept								x		x
ApprovalAssign								x		x
ApprovalCreate								x		x
ApprovalDelete								x		x
ApprovalReject								x		x
ApproverCreate								x		x
ApproverDelete								x		x
CompCreate								x	x	x
CompDelete									x	x
CompLink									x	x
CompModify									x	x
CompRecreate								x	x	x
CompUnlink									x	x
CompView	x	x	x	x	x	x	x	x	x	x
CoreqCreate				x		x	x	x	x	x
CoreqDelete				x		x	x	x	x	x
DefectAccept								x	x	x
DefectAssign									x	x
DefectCancel									x	x
DefectClose	Automatic action									
DefectComment	*Base authority*									

Figure 1 (Part 1 of 5). Shipped Access Authority Groups

CMVC Action	Shipped Access Authority Groups									
	general	manager	developer	developer+	writer	writer+	builder	releaselead	componentlead	projectlead
DefectDesign				x		x		x	x	x
DefectModify								x		x
DefectOpen	Base authority									
DefectReopen									x	x
DefectReturn								x	x	x
DefectReview				x		x			x	x
DefectSize				x		x			x	x
DefectVerify									x	x
DefectView	x	x	x	x	x	x	x	x	x	x
EnvCreate								x		x
EnvDelete								x		x
EnvModify								x		x
FeatureAccept								x	x	x
FeatureAssign									x	x
FeatureCancel									x	x
FeatureClose	Automatic action									
FeatureComment	Base authority									
FeatureDesign				x		x		x	x	x
FeatureModify								x	x	x
FeatureOpen	Base authority									
FeatureReopen									x	x
FeatureReturn								x	x	x
FeatureReview				x		x			x	x
FeatureSize				x		x			x	x
FeatureVerify									x	x
FeatureView	x	x	x	x	x	x	x	x	x	x
FileAdd			x	x	x	x	x	x	x	x
FileCheckIn										x
FileCheckOut			x	x	x	x	x	x	x	x
FileForceIn				x		x	x	x	x	x
FileForceOut				x		x	x	x	x	x
FileDelete				x		x	x	x	x	x
FileDeleteForce				x		x	x	x	x	x

Figure 1 (Part 2 of 5). Shipped Access Authority Groups

CMVC Action	Shipped Access Authority Groups									
	general	**manager**	**developer**	**developer+**	**writer**	**writer+**	**builder**	**releaselead**	**componentlead**	**projectlead**
FileDestroy										x
FileExtract			x	x	x	x	x	x	x	x
FileLink				x		x	x	x	x	x
FileLock				x		x	x	x	x	x
FileLockForce				x		x	x	x	x	x
FileModify				x		x	x	x	x	x
FileRecreate				x		x	x	x	x	x
FileRecreaForce				x		x	x	x	x	x
FileRename			x	x	x	x	x	x	x	x
FileRenameForce				x		x	x	x	x	x
FileResolve	Base authority									
FileUndo				x		x	x	x	x	x
FileUndoForce				x		x	x	x	x	x
FileUnlock							x	x	x	x
FileView	x	x	x	x	x	x	x	x	x	x
FixActive				x		x	x	x		x
FixAssign								x	x	x
FixComplete				x		x	x	x		x
FixCreate				x		x	x	x	x	x
FixDelete				x		x		x		x
HostCreate	Superuser or *owner* implicit authority									
HostDelete	Superuser or owner implicit authority									
LevelAssign							x			x
LevelCheck							x			x
LevelCommit							x			x
LevelComplete							x			x
LevelCreate							x			x
LevelDelete							x			x
LevelExtract				x		x	x	x	x	x
LevelModify							x			x
LevelView	x	x	x	x	x	x	x	x	x	x
MemberCreate							x			x
MemberDelete							x			x

Figure 1 (Part 3 of 5). Shipped Access Authority Groups

CMVC Action	Shipped Access Authority Groups									
	general	manager	developer	developer+	writer	writer+	builder	releaselead	componentlead	projectlead
NotifyCreate	x	x	x	x	x	x	x	x	x	x
NotifyDelete								x	x	x
ReleaseCreate								x		x
ReleaseDelete										x
ReleaseExtract			x	x	x	x	x	x	x	x
ReleaseLink								x		x
ReleaseModify								x		x
ReleaseRecreate										x
ReleaseView	x	x	x	x	x	x	x	x	x	x
Report	Base authority									
SizeAccept									x	x
SizeAssign									x	x
SizeCreate				x		x	x		x	x
SizeDelete									x	x
SizeReject									x	x
TestAbstain								x		x
TestAccept								x		x
TestAssign								x		x
TestReady	Automatic action									
TestReject								x		x
TrackAssign								x		x
TrackCancel								x		x
TrackCheck							x	x		x
TrackCommit										x
TrackComplete							x	x		x
TrackCreate							x	x	x	x
TrackFix							x	x		x
TrackIntegrate								x		x
TrackModify								x	x	x
TrackTest										x
TrackView	x	x	x	x	x	x	x	x	x	x
UserCreate	Superuser implicit authority									
UserDelete	Superuser implicit authority									

Figure 1 (Part 4 of 5). Shipped Access Authority Groups

CMVC Action	Shipped Access Authority Groups									
	general	manager	developer	developer+	writer	writer+	builder	releaselead	componentlead	projectlead
UserModify	Superuser or owner implicit authority									
UserRecreate	Superuser implicit authority									
UserView	x	x	x	x	x	x	x	x	x	x
VerifyAbstain									x	x
VerifyAccept									x	x
VerifyAssign									x	x
VerifyReady	Automatic action									
VerifyReject									x	x

Figure 1 (Part 5 of 5). Shipped Access Authority Groups

Chapter 2. Shipped Interest Notification Groups

Figure 2 lists the interest groups supplied by IBM. Each group consists of actions that would be of interest to a particular type of user. Each interest group has an x under it, next to each action included in that interest group. Your family administrator can modify these groups or create new ones to reflect the needs of your organization.

CMVC Action	Shipped Interest Notification Groups						
	manager	developer	builder	tester	low	med	high
AccessCreate							x
AccessDelete							x
AccessRestrict							x
ApprovalAbstain	x						x
ApprovalAccept	x						x
ApprovalAssign							x
ApprovalCreate							x
ApprovalDelete							x
ApprovalReject	x						x
ApproverCreate							x
ApproverDelete							x
CompCreate	New owner implicit notification						
CompDelete		x			x	x	x
CompLink		x				x	x
CompModify							x
CompRecreate		x			x	x	x
CompUnlink		x				x	x
CompView	No notification						
CoreqCreate	No notification						
CoreqDelete	No notification						
DefectAccept	x	x	x	x	x	x	x
DefectAssign	x						x
DefectCancel						x	x
DefectClose	x	x			x	x	x
DefectComment		x					x
DefectDesign						x	x

Figure 2 (Part 1 of 5). Shipped Interest Notification Groups

CMVC Action	Shipped Notification Interest Groups						
	manager	developer	builder	tester	low	med	high
DefectModify	x	x					x
DefectOpen	x	x	x	x	x	x	x
DefectReopen	x	x	x	x	x	x	x
DefectReturn	x	x	x	x	x	x	x
DefectReview							x
DefectSize							x
DefectVerify		x					x
DefectView	No notification						
EnvCreate				x			x
EnvDelete				x			x
EnvModify				x			x
FeatureAccept	x	x	x	x	x	x	x
FeatureAssign	x						x
FeatureCancel						x	x
FeatureClose	x	x			x	x	x
FeatureComment		x					x
FeatureDesign						x	x
FeatureModify	x	x					x
FeatureOpen	x	x	x	x	x	x	x
FeatureReopen	x	x	x	x	x	x	x
FeatureReturn	x	x	x	x	x	x	x
FeatureReview							x
FeatureSize							x
FeatureVerify		x					x
FeatureView	No notification						
FileAdd		x	x			x	x
FileCheckIn							x
FileCheckOut							x
FileDelete		x	x			x	x
FileDestroy							x
FileExtract	No notification						
FileForceIn							x
FileForceOut							x
FileLink		x	x			x	x

Figure 2 (Part 2 of 5). Shipped Interest Notification Groups

CMVC Action	Shipped Notification Interest Groups						
	manager	developer	builder	tester	low	med	high
FileLock							x
FileLockForce	FileLock subscribers						
FileModify							x
FileRecreate		x	x			x	x
FileRecreaForce	FileRecreate subscribers						
FileRename		x	x			x	x
FileRenameForce	FileRename subscribers						
FileResolve	No notification						
FileUnlock							x
FileUndo							x
FileUndoForce	FileUndo subscribers						
FileView	No notification						
FixActive		x					x
FixAssign							x
FixComplete		x					x
FixCreate							x
FixDelete							x
HostCreate	No notification						
HostDelete	No notification						
LevelAssign							x
LevelCheck	No notification						
LevelCommit	x	x	x	x	x	x	x
LevelComplete			x			x	x
LevelCreate		x	x			x	x
LevelDelete						x	x
LevelExtract	No notification						
LevelModify							x
LevelView	No notification						
MemberCreate		x		x		x	x
MemberDelete		x		x		x	x
NotifyCreate	No notification						
NotifyDelete	No notification						
ReleaseCreate	x	x	x	x	x	x	x
ReleaseDelete	x	x	x	x	x	x	x

Figure 2 (Part 3 of 5). Shipped Interest Notification Groups

CMVC Action	Shipped Notification Interest Groups						
	manager	**developer**	**builder**	**tester**	**low**	**med**	**high**
ReleaseExtract	No notification						
ReleaseLink						x	x
ReleaseModify							x
ReleaseRecreate	x	x	x	x	x	x	x
ReleaseView	No notification						
Report	No notification						
SizeAccept							x
SizeAssign							x
SizeCreate							x
SizeDelete							x
SizeReject							x
TestAbstain	x					x	x
TestAccept	x					x	x
TestAssign							x
TestReady	Owner implicit notification						
TestReject	x	x				x	x
TrackAssign							x
TrackCancel							x
TrackCheck	No notification						
TrackCommit						x	x
TrackComplete						x	x
TrackCreate						x	x
TrackFix				x		x	x
TrackIntegrate		x	x			x	x
TrackModify							x
TrackTest						x	x
TrackView	No notification						
UserCreate	New user implicit notification						
UserDelete	No notification						
UserModify	No notification						
UserUnDelete	No notification						
UserView	No notification						
VerifyAbstain	x					x	x
VerifyAccept	x					x	x

Figure 2 (Part 4 of 5). Shipped Interest Notification Groups

CMVC Action	Shipped Notification Interest Groups						
	manager	**developer**	**builder**	**tester**	**low**	**med**	**high**
VerifyAssign							x
VerifyReady	Owner implicit notification						
VerifyReject	x	x				x	x

Figure 2 (Part 5 of 5). Shipped Interest Notification Groups

Chapter 3. Shipped CMVC Processes

The following tables list the preconfigured processes supplied by IBM. Each process groups different combinations of CMVC subprocesses. An x under the CMVC subprocess indicates that it has been included in the corresponding process. Your family administrator can modify these processes or configure new ones to reflect the needs of your organization. For more information on configurable processes, refer to the book *IBM CMVC Server Administration and Installation*.

The preconfigured *component* processes shipped with CMVC are listed in Figure 3.

Shipped CMVC Component Process	**CMVC Subprocesses**				
	dsrDefect	**dsrFeature**	**verifyDefect**	**verifyFeature**	**none**
default		x	x	x	
prototype					x
development		x			
test		x	x	x	
preship	x	x	x	x	
maintenance			x	x	
emergency_fix					x

Figure 3. Shipped Component Processes

The preconfigured *release* processes shipped with CMVC are listed in Figure 4.

Shipped CMVC Release Processes	CMVC Subprocesses					
	track	**approval**	**fix**	**level**	**test**	**none**
prototype						x
development	x		x		x	
test	x		x	x	x	
preship	x	x	x	x	x	
maintenance	x		x	x	x	
emergency_fix	x			x		
track_only	x					
track_level	x		x	x		
track_approval	x	x	x	x		
track_test	x		x	x	x	
track_full	x	x	x	x	x	
no_track						x

Figure 4. Shipped Release Processes

Chapter 4. Authority and Notification for CMVC Actions

Figure 5 lists all of the CMVC actions, the required implicit and *explicit authority* to perform the action, and the users who are notified when an action is performed.

Note: Users performing the action are excluded from the notification that is sent out once the action is successfully completed.

CMVC Action	Explicit Authority	Implicit Authority	User Notified
AccessCreate	`AccessCreate` defined in an authority group in the *access list* for the component where access is being added	component owner	user being given new access, subscribers
AccessDelete	`AccessDelete` defined in an authority group in the access list for the component where access is being altered.	component owner	user whose access was *deleted*, subscribers
AccessRestrict	`AccessRestrict` defined in an authority group in the access list for the component where access is being restricted	component owner	user whose access is being restricted, subscribers
ApprovalAbstain	`ApprovalAbstain` defined in an authority group in the access list for the component that manages the associated release	approval record owner	approval record owner, subscribers
ApprovalAccept	`ApprovalAccept` defined in an authority group in the access list for the component that manages the associated release	approval record owner	approval record owner, subscribers
ApprovalAssign	`ApprovalAssign` defined in an authority group in the access list for the component that manages the associated release	approval record owner	new and original approval record owners, subscribers
ApprovalCreate	`ApprovalCreate` defined in an authority group in the access list for the component that manages the associated release	track owner	new approval record owner, subscribers
ApprovalDelete	`ApprovalDelete` defined in an authority group in the access list for the component that manages the associated release	not applicable	approval record owner, subscribers
ApprovalReject	`ApprovalReject` defined in an authority group in the access list for the component that manages the associated release	approval record owner	approval record owner, subscribers
ApproverCreate	`ApproverCreate` defined in an authority group in the access list for the component that manages the associated release	release owner	new *approver*, subscribers

Figure 5 (Part 1 of 11). Authority and Notification for CMVC Actions

CMVC Action	Explicit Authority	Implicit Authority	User Notified
ApproverDelete	`ApproverDelete` defined in an authority group in the access list for the component that manages the associated release	release owner	deleted approver, subscribers
CompCreate	`CompCreate` defined in an authority group in the access list for the *parent component*	parent component owner	new component owner
CompDelete	`CompDelete` defined in an authority group in the access list for the component being removed	component owner	component owner, subscribers
CompLink	`CompLink` defined in an authority group in the access list for the component being linked	component owner of the component being linked	owners of both components, subscribers
CompModify	`CompModify` defined in an authority group in the access list for the component being modified	component owner	new component owner if applicable, subscribers
CompRecreate	`CompRecreate` defined in an authority group in the access list for the parent component	parent component owner	owners of both components, subscribers
CompView	`CompView` defined in an authority group in the access list for the component being viewed	component owner	not applicable
CompUnlink	`CompUnlink` defined in an authority group in the access list for the component being unlinked	component owner of the component being unlinked	owners of both components, subscribers
CoreqCreate	`CoreqCreate` defined in an authority group in the access list for the component managing the associated track and release	track owner of all specified tracks	not applicable
CoreqDelete	`CoreqDelete` defined in an authority group in the access list for the component managing the associated track and release	track owner of all specified tracks	not applicable
DefectAccept	`DefectAccept` defined in an authority group in the access list for the component associated with the defect	defect owner	defect owner, defect *originator*, duplicate defect originators, subscribers
DefectAssign	`DefectAssign` defined in an authority group in the access list for the component associated with the defect **Note:** Originators who do not have `DefectAssign` authority can reassign the defect only when it is in the open state.	defect owner, defect originator	new owner, defect originator, duplicate defect originators, subscribers
DefectCancel	`DefectCancel` defined in an authority group in the access list for the component associated with the defect	defect originator	defect owner, defect originator, duplicate defect originators, subscribers

Figure 5 (Part 2 of 11). Authority and Notification for CMVC Actions

CMVC Action	Explicit Authority	Implicit Authority	User Notified
DefectClose	takes place automatically; no authority is required		defect owner, defect originator, duplicate defect originators, subscribers
DefectComment	not applicable; this is a base authority that can be performed by all users in the *family*		defect owner, defect originator, duplicate defect originators, subscribers
DefectDesign	`DefectDesign` defined in an authority group in the access list for the component associated with the defect	defect owner	defect owner, defect originator, duplicate defect originators, subscribers
DefectModify	`DefectModify` defined in the authority group in the access list for the component associated with the defect can modify: abstract, answer, name, environment, level, originator, prefix, reference release, severity, phaseFound*, phaseInject*, priority*, symptom*, target* *If these fields have been configured by your family administrator, the field names may differ from those shown.	defect owner can modify: answer, abstract, environment, level, prefix, reference, release, and all configurable fields. defect originator can modify: originator, severity, name, abstract, environment, level, prefix, reference, release, and all configurable fields.	defect owner, defect originator, duplicate defect originators, subscribers
DefectOpen	not applicable; this is a base authority that can be performed by all users in the family		component owner, subscribers
DefectReopen	`DefectReopen` defined in an authority group in the access list for the component associated with the defect	defect originator	defect owner, defect originator, duplicate defect originators, subscribers
DefectReturn	`DefectReturn` defined in an authority group in the access list for the component associated with the defect	defect owner	defect originator, duplicate defect originators, subscribers
DefectReview	`DefectReview` defined in an authority group in the access list for the component associated with the defect	defect owner	defect owner, defect originator, duplicate defect originators, subscribers
DefectSize	`DefectSize` defined in an authority group in the access list for the component associated with the defect	defect owner	defect owner, defect originator, duplicate defect originators, subscribers
DefectVerify	`DefectVerify` defined in an authority group in the access list for the component associated with the defect	defect owner	defect owner, defect originator, duplicate defect originators, subscribers

Figure 5 (Part 3 of 11). Authority and Notification for CMVC Actions

CMVC Action	Explicit Authority	Implicit Authority	User Notified
DefectView	`DefectView` defined in an authority group in the access list for the component associated with the defect	defect owner, defect originator	not applicable
EnvCreate	`EnvCreate` defined in an authority group in the access list for the component associated with the release	release owner	*tester*, subscribers
EnvDelete	`EnvDelete` defined in an authority group in the access list for the component associated with the release	release owner	subscribers
EnvModify	`EnvModify` defined in an authority group in the access list for the component associated with the release	release owner	tester, subscribers
FeatureAccept	`FeatureAccept` defined in an authority group in the access list for the component associated with the feature	feature owner	feature owner, feature originator, duplicate feature originators, subscribers
FeatureAssign	`FeatureAssign` defined in an authority group in the access list for the component associated with the feature	feature owner	new owner, feature originator, duplicate feature originators, subscribers
FeatureCancel	`FeatureCancel` defined in an authority group in the access list for the component associated with the feature	feature originator	feature owner, feature originator, duplicate feature originators, subscribers
FeatureClose	takes place automatically; no authority is required		feature owner, feature originator, duplicate feature originators, subscribers
FeatureComment	not applicable; this is a base authority that can be performed by all users in the family		feature owner, feature originator, duplicate feature originators, subscribers
FeatureDesign	`FeatureDesign` defined in an authority group in the access list for the component associated with the feature	feature owner	feature owner, feature originator, duplicate feature originators, subscribers
FeatureModify	`FeatureModify` defined in an authority group in the access list for the component associated with the feature can modify: abstract, name, originator, prefix, reference, priority*, target* *If these fields have been configured by your family administrator, the field names may differ from those shown.	feature owner can modify: abstract, prefix, reference and all configurable fields. feature originator can modify: originator, abstract, name, prefix, reference and all configurable fields.	feature owner, feature originator, duplicate feature originators, subscribers

Figure 5 (Part 4 of 11). Authority and Notification for CMVC Actions

CMVC Action	Explicit Authority	Implicit Authority	User Notified
FeatureOpen	not applicable; this is a base authority that can be performed by all users in the family		component owner, subscribers
FeatureReopen	`FeatureReopen` defined in an authority group in the access list for the component associated with the feature	feature originator	feature owner, feature originator, duplicate feature originators, subscribers
FeatureReturn	`FeatureReturn` defined in an authority group in the access list for the component associated with the feature	feature owner	feature owner, feature originator, duplicate feature originators, subscribers
FeatureReview	`FeatureReview` defined in an authority group in the access list for the component associated with the feature	feature owner	feature owner, feature originator, duplicate feature originators, subscribers
FeatureSize	`FeatureSize` defined in an authority group in the access list for the component associated with the feature	feature owner	feature owner, feature originator, duplicate feature originators, subscribers
FeatureVerify	`FeatureVerify` defined in an authority group in the access list for the component associated with the feature	feature owner	feature owner, feature originator, duplicate feature originators, subscribers
FeatureView	`FeatureView` defined in an authority group in the access list for the component associated with the feature	feature owner	not applicable
FileAdd	`FileAdd` defined in an authority group in the access list for the component associated with the file	component owner	subscribers
FileCheckIn	`FileCheckIn` defined in an authority group in the access list for the component associated with the file	user who *checked out* or *locked* the file originally, component owner	subscribers
FileCheckOut	`FileCheckOut` defined in an authority group in the access list for the component associated with the file	component owner	subscribers
FileDelete	`FileDelete` defined in an authority group in the access list for the component associated with the file	component owner	subscribers
FileDeleteForce	`FileDeleteForce` defined in an authority group in the access list for the component associated with the file	component owner	subscribers
FileDestroy	`FileDestroy` defined in an authority group in the access list for the component associated with the file	component owner	subscribers
FileExtract	`FileExtract` defined in an authority group in the access list for the component associated with the file	component owner	not applicable

Figure 5 (Part 5 of 11). Authority and Notification for CMVC Actions

CMVC Action	Explicit Authority	Implicit Authority	User Notified
FileForceIn	`FileForceIn` defined in an authority group in the access list for the component associated with the file	component owner	subscribers
FileForceOut	`FileForceOut` defined in an authority group in the access list for the component associated with the file	component owner	subscribers
FileLink	`FileLink` defined in an authority group in the access list for the component associated with the file	component owner	subscribers
FileLock	`FileLock` defined in an authority group in the access list for the component associated with the file	component owner	subscribers
FileLockForce	`FileLockForce` defined in an authority group in the access list for the component associated with the file	component owner	subscribers
FileModify	`FileModify` defined in an authority group in the access list for the component associated with the file	component owner	subscribers
FileRecreate	`FileRecreate` defined in an authority group in the access list for the component associated with the file	component owner	subscribers
FileRecreaForce	`FileRecreateForce` defined in an authority group in the access list for the component associated with the file	component owner	subscribers
FileRename	`FileRename` defined in an authority group in the access list for the component associated with the file	component owner	subscribers
FileRenameForce	`FileRenameForce` defined in an authority group in the access list for the component associated with the file	component owner	subscribers
FileResolve	not applicable; this is a base authority that can be performed by all users in the family		not applicable
FileUnlock	`FileUnlock` defined in an authority group in the access list for the component associated with the file	user who checked out or locked the file originally, component owner	subscribers
FileUndo	`FileUndo` defined in an authority group in the access list for the component associated with the file	component owner	subscribers
FileUndoForce	`FileUndoForce` defined in an authority group in the access list for the component associated with the file	component owner	subscribers

Figure 5 (Part 6 of 11). Authority and Notification for CMVC Actions

CMVC Action	Explicit Authority	Implicit Authority	User Notified
FileView	`FileView` defined in an authority group in the access list for the component associated with the file	component owner	not applicable
FixActive	`FixActive` defined in an authority group in the access list for the component associated with the *fix record*	fix record owner, component owner, track owner	subscribers
FixAssign	`FixAssign` defined in an authority group in the access list for the component associated with the fix record	fix record owner, component owner, track owner	new fix record owner, subscribers
FixComplete	`FixComplete` defined in an authority group in the access list for the component associated with the fix record	fix record owner, component owner, track owner	subscribers
FixCreate	`FixCreate` defined in an authority group in the access list for the component associated with the defect or feature	defect or feature owner, track owner	subscribers
FixDelete	`FixDelete` defined in an authority group in the access list for the component associated with the defect or feature	defect or feature owner, track owner	subscribers
HostCreate	superuser	owner of the user ID for which a *host list* entry is being created	not applicable
HostDelete	superuser	owner of the user ID for which a host list entry is being deleted	not applicable
LevelAssign	`LevelAssign` defined in an authority group in the access list for the component associated with the release	level owner	new owner, subscribers
LevelCheck	`LevelCheck` defined in an authority group in the access list for the component associated with the release	level owner	not applicable
LevelCommit	`LevelCommit` defined in an authority group in the access list for the component associated with the release	not applicable	subscribers
LevelComplete	`LevelComplete` defined in an authority group in the access list for the component associated with the release	not applicable	subscribers
LevelCreate	`LevelCreate` defined in an authority group in the access list for the component associated with the release	release owner	subscribers

Figure 5 (Part 7 of 11). Authority and Notification for CMVC Actions

CMVC Action	Explicit Authority	Implicit Authority	User Notified
LevelDelete	`LevelDelete` defined in an authority group in the access list for the component associated with the release	level owner	subscribers
LevelExtract	`LevelExtract` defined in an authority group in the access list for the component associated with the release	level owner	not applicable
LevelModify	`LevelModify` defined in an authority group in the access list for the component associated with the release	level owner	level owner, subscribers
LevelView	`LevelView` defined in an authority group in the access list for the component associated with the release	level owner	not applicable
MemberCreate	`MemberCreate` defined in an authority group in the access list for the component associated with the release	level owner	level owner, subscribers
MemberDelete	`MemberDelete` defined in an authority group in the access list for the component associated with the release	level owner	level owner, subscribers
NotifyCreate	`NotifyCreate` defined in an authority group in the access list for the component associated with the *notification list*	component owner	not applicable
NotifyDelete	`NotifyDelete` defined in an authority group in the access list for the component associated with the notification list **Note:** Users can delete themselves from a notification list without requiring any authority	component owner, owner of user ID	not applicable
ReleaseCreate	`ReleaseCreate` defined in an authority group in the access list for the component associated with the new release	not applicable	new release owner, component owner, subscribers
ReleaseDelete	`ReleaseDelete` defined in an authority group in the access list for the component associated with the release	release owner	release owner, component owner, subscribers
ReleaseLink	`ReleaseLink` defined in an authority group in the access list for the component associated with the release	release owner	release owner, subscribers

Figure 5 (Part 8 of 11). Authority and Notification for CMVC Actions

CMVC Action	Explicit Authority	Implicit Authority	User Notified
ReleaseExtract	`ReleaseExtract` defined in an authority group in the access list for the component associated with the release	release owner	not applicable
ReleaseModify	`ReleaseModify` defined in an authority group in the access list for the component associated with the release **Note:** To modify the component name, you must have `ReleaseCreate` defined in an authority group in the component that you are modifying	release owner	release owner, subscribers, new owner (if applicable)
ReleaseRecreate	`ReleaseRecreate` defined in an authority group in the access list for the component associated with the release	release owner	release owner, component owner, subscribers
ReleaseView	`ReleaseView` defined in an authority group in the access list for the component associated with the release	release owner	not applicable
Report	not applicable; this is a base authority that can be performed by all users in the family		not applicable
SizeAccept	`SizeAccept` defined in an authority group in the access list for the component associated with the sizing record	sizing record owner	subscribers
SizeAssign	`SizeAssign` defined in an authority group in the access list for the component associated with the sizing record	sizing record owner	new sizing record owner, feature owner, subscribers
SizeCreate	`SizeCreate` defined in an authority group in the access list for the component associated with the feature	feature owner	component owner, feature owner, subscribers
SizeDelete	`SizeDelete` defined in an authority group in the access list for the component associated with the feature	feature owner	subscribers, sizing record owner, feature owner
SizeReject	`SizeReject` defined in an authority group in the access list for the component associated with the sizing record	sizing record owner	subscribers
TestAbstain	`TestAbstain` defined in an authority group in the access list for the component associated with the test record's release	test record owner	subscribers

Figure 5 (Part 9 of 11). Authority and Notification for CMVC Actions

CMVC Action	Explicit Authority	Implicit Authority	User Notified
TestReject	`TestReject` defined in an authority group in the access list for the component associated with the test record's release	test record owner	subscribers
TestAccept	`TestAccept` defined in an authority group in the access list for the component associated with the test record's release	test record owner	subscribers
TestAssign	`TestAssign` defined in an authority group in the access list for the component associated with the test record's release	test record owner	new test record owner, subscribers
TrackAssign	`TrackAssign` defined in an authority group in the access list for the component associated with the release	track owner	new track owner, subscribers
TrackCancel	`TrackCancel` defined in an authority group in the access list for the component associated with the defect or feature	defect or feature owner	subscribers, owners of approval records for track being canceled
TrackCheck	`TrackCheck` defined in an authority group in the access list for the component associated with the release	track owner	not applicable
TrackCommit	`TrackCommit` defined in an authority group in the access list for the component associated with the release	track owner	subscribers
TrackComplete	`TrackComplete` defined in an authority group in the access list for the component associated with the release	track owner	subscribers
TrackCreate	`TrackCreate` defined in an authority group in the access list for the component associated with the defect or feature	defect or feature owner	track owner, subscribers
TrackFix	`TrackFix` defined in an authority group in the access list for the component associated with the release	track owner	subscribers
TrackIntegrate	`TrackIntegrate` defined in an authority group in the access list for the component associated with the release	track owner	subscribers
TrackModify	`TrackModify` defined in an authority group in the access list for the component associated with the release	track owner	subscribers

Figure 5 (Part 10 of 11). Authority and Notification for CMVC Actions

CMVC Action	Explicit Authority	Implicit Authority	User Notified
TrackTest	`TrackTest` defined in an authority group in the access list for the component associated with the release	track owner	subscribers
TrackView	`TrackView` defined in an authority group in the access list for the component associated with the release	track owner	not applicable
UserCreate	superuser	not applicable	new user
UserDelete	superuser	not applicable	not applicable
UserModify	must be a superuser in order to grant the superuser privilege	owner of the user object can modify all characteristics except the *superuser privilege*	not applicable
UserRecreate	superuser	not applicable	not applicable
UserView	not applicable; this is a base authority that can be performed by all users in the family		not applicable
VerifyAbstain	`VerifyAbstain` defined in an authority group in the access list for the component associated with the verification record's defect or feature	*verification record* owner	subscribers
VerifyAccept	`VerifyAccept` defined in an authority group in the access list for the component associated with the verification record's defect or feature	verification record owner	subscribers
VerifyAssign	`VerifyAssign` defined in an authority group in the access list for the component associated with the verification record's defect or feature	verification record owner	new verification record owner, subscribers
VerifyReady	takes place automatically; no authority is required		verification record owners
VerifyReject	`VerifyReject` defined in an authority group in the access list for the component associated with the verification record's defect or feature	verification record owner	subscribers

Figure 5 (Part 11 of 11). Authority and Notification for CMVC Actions

Chapter 5. Shipped Configuration Table Values

Configuration table values are shown in the following tables under the following column headings:

Type Configuration types supported by CMVC.

Name Shipped values for the various configuration types.

Note: Your CMVC family administrator can add names for each configuration type.

Description A description of each name value shipped with CMVC.

Note: Your CMVC family administrator can add descriptions for fields.

Your CMVC family administrator can set defaults. For information on setting defaults, see the book *IBM CMVC Server Administration and Installation*.

The timing or scheduling requirements for resolving a defect or implementing a feature

Type	Name	Description
priority	mustfix	Defect or feature must be resolved in this release
priority	candidate	Defect or feature is a candidate if time permits
priority	deferred	Defect or feature deferred to next release
priority	easy	Defect or feature is easy to solve or implement
priority	moderate	Defect or feature is moderately difficult to resolve
priority	difficult	Defect or feature is difficult to solve or implement
priority	n/a	Priority does not apply to this defect or feature

The type of level

Type	Name	Description
leveltype	development	Development level
leveltype	production	Production level
leveltype	integration	Integration level
leveltype	prototype	Prototype level
leveltype	other	Other type of level

The symptom of the problem a defect was opened to resolve

Type	Name	Description
symptom	incorrect_i/o	Incorrect or unexpected input or output
symptom	program_defect	Program defect
symptom	design_wrong	Original design is incorrect; redesign required
symptom	function_needed	Additional function is required
symptom	plans_incorrect	Plans need to be changed or enhanced
symptom	docs_incorrect	Documentation is incorrect
symptom	prog_suspended	Program suspended during normal operation
symptom	core_dump	Core dump occurred during normal operation
symptom	lost_data	Data loss occurred during normal operation
symptom	usability	Program or application is not usable as is
symptom	test_failed	Test failed
symptom	build_failed	Build, compile, or module integration failed
symptom	install_failed	Installation failed
symptom	obsolete_code	Remove obsolete code
symptom	intgr_problem	Integration problems with other applications
symptom	performance	Performance problems; code needs to be optimized
symptom	reliability	Reliability problems; code needs more work
symptom	non-standard	Coding practices or program execution is non-standard
symptom	not_to_spec	Program or application does not function as specified

The severity of the problem that a defect was opened to resolve

Type	Name	Description
severity	1	Wrong results or failure; critical to program execution
severity	2	Wrong results; not critical to program execution
severity	3	Unexpected behavior
severity	4	Suggestion or enhancement request

The development phase in progress when a defect was found or injected

Type	Name	Description
phase	design	Design Phase
phase	planning	Planning Phase
phase	strategy	Strategic Planning Phase
phase	prototyping	Prototyping Phase
phase	development	Development Phase

Type	Name	Description
phase	documenting	Documentation or Publication Phase
phase	inspections	Inspection Phase
phase	maintenance	Maintenance Phase
phase	building	Building, Compiling, or Module Integration Phase
phase	unit_test	Unit Test
phase	functional_test	Functional Test
phase	regression_test	Regression Test
phase	install_test	Installation Test
phase	config_test	Configuration Test
phase	integrate_test	Integration Test
phase	quality_test	Quality Assurance Test
phase	usability_test	Usability Test
phase	ship_test	Ship Test
phase	beta_test	Beta Test
phase	n/a	Not applicable to any particular phase

The type of defect or feature

Type	Name	Description
defectPrefix	c	Defect reported by a customer
defectPrefix	d	Defect reported by internal users
featurePrefix	s	Suggestion made by a customer
featurePrefix	f	Feature requested by internal user

The reason a defect or feature is being accepted

Type	Name	Description
answerAccept	program_defect	The problem was due to a program error
answerAccept	docs_defect	Documentation needs to be changed
answerAccept	docs_change	Documentation needs to address new features
answerAccept	plans_change	Plans or schedules need to be changed
answerAccept	new_function	New function will be added
answerAccept	redesign	Current function needs to be redesigned
answerAccept	fix_testcase	Test case needs to be fixed
answerAccept	remove_code	Obsolete code needs to be removed
answerAccept	remove_support	Nonsupported functions need to be removed
answerAccept	comply_with	Coding practices and operation must comply with standards

The reason a defect or feature is being returned

Type	Name	Description
answerReturn	fixed	The problem is already fixed
answerReturn	future	Future releases or versions will address the defect or feature
answerReturn	duplicate	This is a duplicate of an existing defect or feature
answerReturn	usage_error	The problem is caused by incorrect usage
answerReturn	hardware_error	The problem is caused by a hardware error
answerReturn	info_needed	More information is required
answerReturn	limitation	This problem is a current limitation
answerReturn	suggestion	This problem is a suggestion, not an error
answerReturn	unrecreatable	The problem cannot be re-created
answerReturn	as_designed	The program works as designed
answerReturn	deviation	Code or documentation will deviate from the standards

Chapter 6. CMVC Views for Queries

This chapter shows the field names for all of the CMVC views. The view names that you must use with **Report -view** are marked with a double dagger symbol(‡). View names not marked with a double dagger are available only as subselect criteria. The fields are listed in the order in which they appear in the database. Field names in italics may be used as search criteria, but they do not provide data output. Specify values for date fields in the format *yy/mm/dd hh:mm:ss*.

When entering a query, do not abbreviate the field names. Depending on the database used by your installation, some abbreviations can be interpreted incorrectly and give inaccurate information. In most situations, you will receive only an error message if you abbreviate a field name.

For example, if your installation uses the ORACLE** database and you replace the field names `userLogin` or `userName` with `user`, you do not receive an error message. The search, however, is ended with no records found.

Note: The INFORMIX** data types are listed in the following tables. The ORACLE equivalents are integer for number, varchar for char, and text in table for long. The SYBASE** equivalents are integer for number, varchar for char, and text for long. The DB2/6000* equivalents are integer for number, varchar for char, and long varchar for long.

If your family administrator configures the database fields, the following field names may differ from those in your database.

AccessDownView‡

Access List Entries Including Those of All Child Components

Fieldname[length]	Datatype	Description
compId	*number*	*database id of the component where access is granted or restricted*
userId	*number*	*database id of the user who owns the access authority*
authorityName[31]	char	Access Authority Group Name
authorityType[15]	char	Access Authority Type ('granted' or 'restricted')
compName[63]	*char*	*component name*
childCompName[63]	char	Child Component *Name*
userLogin[31]	char	User's CMVC User ID
userName[63]	char	User's Full Name
userArea[31]	char	User's Work Area or Department

AccessFastView

Low Level Actions Granted to Users at a Component

Fieldname[length]	Datatype	Description
userId	*number*	*database id of the user who owns the access authority*
action[15]	*char*	*CMVC action name*
authorityType[15]	*char*	*access authority type* (‘granted’ or ‘restricted’)
compId	*number*	*database id of the component where access is granted or restricted*

AccessNInheritView

Low Level Actions Restricted to Users at a Component

Fieldname[length]	Datatype	Description
userId	*number*	*database id of the user who owns the access authority*
action[15]	*char*	*CMVC action name*
authorityType[15]	*char*	*access authority type* (‘granted’ or ‘restricted’)
compId	*number*	*database id of the component where access is granted or restricted*

AccessUpView

Access List Entries Including Those of All Parent Components

Fieldname[length]	Datatype	Description
compId	*number*	*database id of the component where access is granted or restricted*
userId	*number*	*database id of the user who owns the access authority*
authorityName[31]	*char*	*access authority group name*
authorityType[15]	*char*	*access authority type (‘granted’ or ‘restricted’)*
compName[63]	*char*	*component name*
parentName[63]	*char*	*parent component name*
userLogin[31]	*char*	*user's CMVC user ID*
userName[63]	*char*	*user's full name*
userArea[31]	*char*	*user's work area or department*

AccessView‡

Component Access List Entries

Fieldname[length]	Datatype	Description
compId	*number*	*database id of the component where access is granted or restricted*
userId	*number*	*database id of the user who owns the access authority*
authorityName[31]	char	Access Authority Group Name
authorityType[15]	char	Access Authority Type ('granted' or 'restricted')
compName[63]	char	Component Name
userLogin[31]	char	User's CMVC User ID
userName[63]	char	User's Full Name
userArea[31]	char	User's Work Area or Department

ApprovalView‡

Track Approval Records

Fieldname[length]	Datatype	Description
trackId	*number*	*database id of the track for the approval record*
userId	*number*	*database id of the user who owns the approval record*
state[15]	char	Approval Record State
addDate[25]	char	Date Created
lastUpdate[25]	char	Date of Last Update
releaseName[31]	char	Release Name
defectPrefix[31]	char	Defect or Feature Prefix
defectName[31]	char	Defect or Feature Identifier
defectReference[31]	char	Defect or Feature Reference
defectAbstract[127]	char	Defect or Feature Abstract
defectType[7]	char	'defect' or 'feature'
userLogin[31]	char	Approver's CMVC User ID
userName[63]	char	Approver's Full Name
userArea[31]	char	Approver's Work Area or Department

ApproverView‡

Release Approver List Entries

Fieldname[length]	Datatype	Description
releaseId	*number*	*database id of the release associated with the approver list*

Fieldname[length]	Datatype	Description
userId	*number*	*database id of the user who is the approver*
releaseName[31]	char	Release Name
userLogin[31]	char	Approver's CMVC User ID
userName[63]	char	Approver's Full Name
userArea[31]	char	Approver's Work Area or Department

BchangeView

File Changes (Tree View Information)

Fieldname[length]	Datatype	Description
pathId	*number*	*database id of the file's path name*
trackId	*number*	*database id of the track for this file change*
fileId	*number*	*database id of the file*
versionId	*number*	*database id of the file's version*
levelId	*number*	*database id of the level where the change is included and committed*
type[8]	*char*	*type of file change*
defectPrefix[31]	*char*	*defect or feature prefix*
defectName[31]	*char*	*defect or feature identifier*
defectReference[31]	*char*	*defect or feature reference*
defectAbstract[127]	*char*	*defect or feature abstract*
defectType[7]	*char*	*'defect' or 'feature'*
releaseName[31]	*char*	*release name*
pathName[195]	*char*	*file's full path name*
versionSID[47]	*char*	*version number of the changed file*
userId	*number*	*database id of the user who made the change to the file*
levelName[31]	*char*	*level name of the level where the change is included*
compName[63]	*char*	*component name of the file where the change is included*

ChangeExtractView

File Changes

Fieldname[length]	Datatype	Description
pathId	*number*	*database id of the file's path name*
trackId	*number*	*database id of the track for this file change*
fileId	*number*	*database id of the file*
versionId	*number*	*database id of the file's version*
levelId	*number*	*database id of the level where the change is included and committed*

Fieldname[length]	Datatype	Description
type[8]	*char*	*type of file change*
pathName[195]	*char*	*file's full path name*
fileSourceId	*number*	*database id of the file's source. Defines where the file is derived from and is used to identify common files*
versionSID[47]	*char*	*version number of the changed file*

ChangeView‡

File Changes, Including Defect and Feature Information

Fieldname[length]	Datatype	Description
pathId	*number*	*database id of the file's path name*
trackId	*number*	*database id of the track for this file change*
fileId	*number*	*database id of the file*
versionId	*number*	*database id of the file's version*
levelId	*number*	*database id of the level where the change is included and committed*
type[8]	char	Type of File Change
defectPrefix[31]	char	Defect or Feature Prefix
defectName[31]	char	Defect or Feature Identifier
defectReference[31]	char	Defect or Feature Reference
defectAbstract[127]	char	Defect or Feature Abstract
defectType[7]	*char*	*'defect' or 'feature'*
releaseName[31]	char	Release Name
pathName[195]	char	File's Full Path Name
versionSID[47]	char	Version number of the Changed File
userId	*number*	*database id of the user who made the change to the file*
levelName[31]	char	Level Name of the Level where the Change is Included

CompMemberView

Component Hierarchy Linkages

Fieldname[length]	Datatype	Description
parentId	*number*	*database id of the parent component*
childId	*number*	*database id of the component*
rank	*number*	*number of links or generations between parent and child*
reference	*number*	*number of references to this parent-child-rank configuration*
parentCompName[63]	*char*	*parent component name*

Fieldname[length]	Datatype	Description
childCompName[63]	*char*	*component name*

CompView‡

Component Properties

Fieldname[length]	Datatype	Description
id	*number*	*database id of the component*
name[63]	char	Component Name
userId	*number*	*database id of the user who owns the component*
description[127]	char	Component Description
addDate[25]	char	Date Created
dropDate[25]	char	Date Deleted
lastUpdate[25]	char	Date of Last Update
compProcess[31]	char	Component Process Name
featureDSR[3]	char	Feature DSR (design/size/review) Subprocess
featureVerify[3]	char	Feature Verify Subprocess
defectDSR[3]	char	Defect DSR (design/size/review) Subprocess
defectVerify[3]	char	Defect Verify Subprocess
userLogin[31]	char	Component Owner's CMVC User ID
userName[63]	char	Component Owner's Full Name
userArea[31]	char	Component Owner's Work Area or Department

DefectDownView‡

Defect Properties Including Those of Child Components

Fieldname[length]	Datatype	Description
id	*number*	*database id of the defect*
type[7]	*char*	*'defect'*
prefix[31]	char	Defect Prefix
name[31]	char	Defect Identifier
compId	*number*	*database id of the component associated with the defect*
envName[31]	char	Environment where Discovered
state[15]	char	Defect State
severity[31]	char	Severity Level
abstract[127]	char	Defect Abstract
reference[31]	char	Defect Reference
answer[31]	char	Accept or Return Answer Type

Fieldname[length]	Datatype	Description
levelName[31]	char	Name of Level where Discovered
lastUpdate[25]	char	Date of Last Update
addDate[25]	char	Date Created
assignDate[25]	char	Date when Reassigned
responseDate[25]	char	Date Accepted or Returned
endDate[25]	char	Date Closed or Canceled
originId	*number*	*database id of the defect originator*
ownerId	*number*	*database id of the defect owner*
age	number	Age of the Defect (dependent on the family's aging mechanism)
duplicate[31]	char	Duplicate Defect or Feature Identifier
releaseName[31]	char	Name of Release where Discovered
compName[63]	*char*	*component name*
childCompName[63]	char	Child Component Name
originLogin[31]	char	Defect Originator's CMVC User ID
originName[63]	char	Defect Originator's Full Name
originArea[31]	char	Defect Originator's Work Area or Department
ownerLogin[31]	char	Defect Owner's CMVC User ID
ownerName[63]	char	Defect Owner's Full Name
ownerArea[31]	char	Defect Owner's Work Area or Department

DefectView‡

Defect Properties

Fieldname[length]	Datatype	Description
id	*number*	*database id of the defect*
type[7]	*char*	*'defect'*
prefix[31]	char	Defect Prefix
name[31]	char	Defect Identifier
compId	*number*	*database id of the component associated with the Defect*
envName[31]	char	Environment where Discovered
state[15]	char	Defect State
severity[31]	char	Severity Level
abstract[127]	char	Defect Abstract
reference[31]	char	Defect Reference
answer[31]	char	Accept or Return Answer Type
levelName[31]	char	Name of Level where Discovered
lastUpdate[25]	char	Date of Last Update
addDate[25]	char	Date Created

Fieldname[length]	Datatype	Description
assignDate[25]	char	Date when Reassigned
responseDate[25]	char	Date Accepted or Returned
endDate[25]	char	Date Closed or Canceled
originId	*number*	*database id of the defect originator*
ownerId	*number*	*database id of the defect owner*
age	number	Age of the Defect (dependent on the family's aging mechanism)
duplicate[31]	char	Duplicate Defect or Feature Identifier
releaseName[31]	char	Name of Release where Discovered
compName[63]	char	Component Name
originLogin[31]	char	Defect Originator's CMVC User ID
originName[63]	char	Defect Originator's Full Name
originArea[31]	char	Defect Originator's Work Area or Department
ownerLogin[31]	char	Defect Owner's CMVC User ID
ownerName[63]	char	Defect Owner's Full Name
ownerArea[31]	char	Defect Owner's Work Area or Department

EnvView‡

Release Environment List Entries

Fieldname[length]	Datatype	Description
name[31]	char	Environment Name
releaseId	*number*	*database id of the release*
userId	*number*	*database id of the user*
releaseName[31]	char	Release Name
userLogin[31]	char	Tester's CMVC User ID
userName[63]	char	Tester's Full Name
userArea[31]	char	Tester's Work Area or Department

FeatureDownView‡

Feature Properties Including Those of Child Components

Fieldname[length]	Datatype	Description
id	*number*	*database id of the feature*
type[7]	*char*	*'feature'*
prefix[31]	char	Feature Prefix
name[31]	char	Feature Identifier
compId	*number*	*database id of the component associated with the feature*
envName[31]	*char*	N/A

Fieldname[length]	Datatype	Description
state[15]	char	Feature State
severity[31]	*char*	N/A
abstract[127]	char	Feature Abstract
reference[31]	char	Feature Reference
answer[31]	*char*	N/A
levelName[31]	*char*	N/A
lastUpdate[25]	char	Date of Last Update
addDate[25]	char	Date Created
assignDate[25]	char	Date when Reassigned
responseDate[25]	char	Date Accepted or Returned
endDate[25]	char	Date Closed or Canceled
originId	*number*	*database id of the feature originator*
ownerId	*number*	*database id of the feature owner*
age	number	Age of the feature (dependent on the family's aging mechanism)
duplicate[31]	char	Duplicate Defect or Feature Identifier
releaseName[31]	*char*	N/A
compName[63]	*char*	N/A
childCompName[63]	char	Child Component Name
originLogin[31]	char	Feature Originator's CMVC User ID
originName[63]	char	Feature Originator's Full Name
originArea[31]	char	Feature Originator's Work Area or Department
ownerLogin[31]	char	Feature Owner's CMVC User ID
ownerName[63]	char	Feature Owner's Full Name
ownerArea[31]	char	Feature Owner's Work Area or Department

FeatureView‡

Feature Properties

Fieldname[length]	Datatype	Description
id	*number*	*database id of the feature*
type[7]	*char*	*'feature'*
prefix[31]	char	Feature Prefix
name[31]	char	Feature Identifier
compId	*number*	*database id of the component associated with the feature*
envName[31]	*char*	N/A
state[15]	char	Feature State
severity[31]	*char*	N/A
abstract[127]	char	Feature Abstract

Fieldname[length]	Datatype	Description
reference[31]	char	Feature Reference
answer[31]	*char*	N/A
levelName[31]	*char*	N/A
lastUpdate[25]	char	Date of Last Update
addDate[25]	char	Date Created
assignDate[25]	char	Date when Reassigned
responseDate[25]	char	Date Accepted or Returned
endDate[25]	char	Date Closed or Canceled
originId	*number*	*database id of the feature originator*
ownerId	*number*	*database id of the feature owner*
age	number	Age of the feature (dependent on the family's aging mechanism)
duplicate[31]	char	Duplicate Defect or Feature Identifier
releaseName[31]	*char*	N/A
compName[63]	char	Component Name
originLogin[31]	char	Feature Originator's CMVC User ID
originName[63]	char	Feature Originator's Full Name
originArea[31]	char	Feature Originator's Work Area or Department
ownerLogin[31]	char	Feature Owner's CMVC User ID
ownerName[63]	char	Feature Owner's Full Name
ownerArea[31]	char	Feature Owner's Work Area or Department

FileView‡

File Properties

Fieldname[length]	Datatype	Description
id	*number*	*database id of the file*
compId	*number*	*database id of the component associated with the file*
sourceId	*number*	*database id of the file's source. Defines where the file was derived from and is used to identify common files.*
pathId	*number*	*database id of the file's path name*
nuPathId	*number*	*database id of the file's pending path name*
versionId	*number*	*database id of the file's version*
nuVersionId	*number*	*database id of the file's pending version*
addDate[25]	char	Date Created
nuAddDate[25]	char	New Creation Date
dropDate[25]	char	Date Deleted
nuDropDate[25]	char	New Deletion Date

Fieldname[length]	Datatype	Description
lastUpdate[25]	char	Date of Last Update
baseName[127]	char	File Base Name
type[7]	*char*	*'text', 'binary', or 'long'*
releaseId	*number*	*database id of the release associated with the file*
userLogin[31]	char	CMVC user ID who locked or checked out the file.
fmode[4]	char	File Mode
compName[63]	char	Component Name
versionSID[47]	char	Last Committed File Version Number
nuVersionSID[47]	char	Current File Version Number
pathName[195]	char	File Path Name
nuPathName[195]	char	Pending New File Path Name
releaseName[31]	char	Release Name

FilesOutView‡

Files Currently Locked for Editing

Fieldname[length]	Datatype	Description
fileId	*number*	*database id of the file*
versionId	*number*	*database id of the file's version*
userId	*number*	*database id of the user who has the file locked*
checkOutDate[25]	char	Date File was Locked
newSID[47]	char	File's Pending Version Number or Checkin
fileNuPath[195]	char	File Path Name
releaseName[31]	char	Release Name
userLogin[31]	char	User's CMVC User ID
userName[63]	char	User's Full Name
userArea[31]	char	User's Work Area or Department

FixDownView

Fix Records Including Those For Child Components

Fieldname[length]	Datatype	Description
trackId	*number*	*database id of the track*
userId	*number*	*database id of the user who owns the fix record*
compId	*number*	*database id of the component associated with the fix record*
state[15]	*char*	*fix record state*
addDate[25]	*char*	*date created*

Fieldname[length]	Datatype	Description
lastUpdate[25]	*char*	*date of last update*
releaseName[31]	*char*	*release name*
defectPrefix[31]	*char*	*defect or feature prefix*
defectName[31]	*char*	*defect or feature identifier*
defectAbstract[127]	*char*	*defect or feature abstract*
defectReference[31]	*char*	*defect or feature reference*
defectType[7]	*char*	*'defect' or 'feature'*
compName[63]	*char*	*component name*
childCompName[63]	*char*	*child component name*
userLogin[31]	*char*	*fix record owner's CMVC User ID*
userName[63]	*char*	*fix record owner's full name*
userArea[31]	*char*	*fix record owner's work area or department*

FixView‡

Fix Records

Fieldname[length]	Datatype	Description
trackId	*number*	*database id of the track*
userId	*number*	*database id of the user who owns the fix record*
compId	*number*	*database id of the component associated with the fix record*
state[15]	char	Fix Record State
addDate[25]	char	Date Created
lastUpdate[25]	char	Date of Last Update
releaseName[31]	char	Release Name
defectPrefix[31]	char	Defect or Feature Prefix
defectName[31]	char	Defect or Feature Identifier
defectAbstract[127]	char	Defect or Feature Abstract
defectReference[31]	char	Defect or Feature Reference
defectType[7]	char	'defect' or 'feature'
compName[63]	char	Component Name
userLogin[31]	char	Fix Record Owner's CMVC User ID
userName[63]	char	Fix Record Owner's Full Name
userArea[31]	char	Fix Record Owner's Work Area or Department

HistoryView

Defect or Feature State Change History

Fieldname[length]	Datatype	Description
defectId	*number*	*database id of the defect or feature*
userId	*number*	*database id of the user who caused the defect or feature state change*
action[15]	*char*	*action that occurred and caused the state change*
addDate[25]	*char*	*date action occurred*
defectPrefix[31]	*char*	*defect or feature prefix*
defectName[31]	*char*	*defect or feature identifier*
defectReference[31]	*char*	*defect or feature reference*
userLogin[31]	*char*	*CMVC user id of the user performing the actions to the defect or feature*
userName[63]	*char*	*full name of the user performing the actions to the defect or feature*
userArea[31]	*char*	*work area or department of the user performing the actions to the defect or feature*

HostView‡

User Host List Entries

Fieldname[length]	Datatype	Description
userId	*number*	*database id of the user*
name[127]	char	Name of the Client Host
login[31]	char	User's *Login* Name on the Client Host
userLogin[31]	char	User's CMVC User ID
userName[63]	char	User's Full Name
userArea[31]	char	User's Work Area or Department

LevelMemberView‡

Level Members

Fieldname[length]	Datatype	Description
levelId	*number*	*database id of the level*
trackId	*number*	*database id of the track*
levelName[31]	char	Level Name
releaseName[31]	char	Release Name
trackUserLogin[31]	char	Track Owner's CMVC User ID
trackUserName[63]	char	Track Owner's Full Name
trackUserArea[31]	char	Track Owner's Work Area or Department

Fieldname[length]	Datatype	Description
defectPrefix[31]	char	Defect or Feature Prefix
defectName[31]	char	Defect or Feature Identifier
defectReference[31]	char	Defect or Feature Reference
defectAbstract[127]	*char*	*defect or feature abstract*
defectType[7]	*char*	*'defect' or 'feature'*

LevelView‡

Level Properties

Fieldname[length]	Datatype	Description
id	*number*	*database id of the level*
name[31]	char	Level Name
releaseId	*number*	*database id of the release*
userId	*number*	*database id of the user who owns the level*
addDate[25]	char	Date Created
commitDate[25]	char	Date Committed
lastUpdate[25]	char	Date of Last Update
state[15]	char	Level State
type[31]	char	Level Type
releaseName[31]	char	Release Name
userLogin[31]	char	Level Owner's CMVC User ID
userName[63]	char	Level Owner's Full Name
userArea[31]	char	Level Owner's Work Area or Department

NoteView‡

Defect Notes

Fieldname[length]	Datatype	Description
defectId	*number*	*database id of the defect or feature*
userId	*number*	*database id of the user who added remarks*
action[15]	char	Action Occurring when remarks were added
remarks[15 999]	long	Text of Remarks
addDate[25]	char	Date Created
defectPrefix[31]	char	Defect or Feature Prefix
defectName[31]	char	Defect or Feature Identifier
defectReference[31]	char	Defect or Feature Reference
userLogin[31]	char	User's CMVC User ID
userName[63]	char	User's Full Name
userArea[31]	char	User's Work Area or Department

NotifyDownView‡

Notification List Members Including Descendant Members

Fieldname[length]	Datatype	Description
compId	*number*	*database id of the component where notification is controlled*
userId	*number*	*database id of the user who owns the notification interest*
interestName[31]	char	Interest Group Name
compName[63]	*char*	*component name*
childCompName[63]	char	Child Component Name
userAddress[159]	char	User's Sendmail Address
userLogin[31]	char	User's CMVC User ID
userName[63]	char	User's Full Name
userArea[31]	char	User's Work Area or Department

NotifyFastView

Low Level Actions Subscribed to by Users at a Component

Fieldname[length]	Datatype	Description
userId	*number*	*database id of the user who owns the notification interest*
userAddress[159]	*char*	*user's sendmail address*
compId	*number*	*database id of the component where notification is controlled*
action[15]	*char*	*low level action name*

NotifyUpView‡

Notification List Entries Including Those Inherited from Parents

Fieldname[length]	Datatype	Description
compId	*number*	*database id of the component where notification is controlled*
userId	*number*	*database id of the user who owns the notification interest*
interestName[31]	char	Interest Group Name
compName[63]	*char*	*component name*
parentName[63]	char	Parent Component Name
userAddress[159]	char	User's Sendmail Address
userLogin[31]	char	User's CMVC User ID
userName[63]	char	User's Full Name
userArea[31]	char	User's Work Area or Department

NotifyView‡

Notification List Entries

Fieldname[length]	Datatype	Description
compId	*number*	*database id of the component where notification is controlled*
userId	*number*	*database id of the user who owns the notification interest*
interestName[31]	char	Interest Group Name
compName[63]	char	Component Name
userAddress[159]	char	User's Sendmail Address
userLogin[31]	char	User's CMVC User ID
userName[63]	char	User's Full Name
userArea[31]	char	User's Work Area or Department

ReleaseView‡

Release Properties

Fieldname[length]	Datatype	Description
id	*number*	*database id of the release*
name[31]	char	Release Name
compId	*number*	*database id of the component associated with the release*
binding[3]	char	Binding Status (no longer used)
description[127]	char	Release Description
userId	*number*	*database id of the user who owns the release*
addDate[25]	char	Date Created
dropDate[25]	char	Date Deleted
lastUpdate[25]	char	Date of Last Update
relSize	number	(not currently used)
relProcess[31]	char	Release Process Name
track[3]	char	Track Subprocess (‘yes’ or ‘no’)
approve[3]	char	Approval Subprocess (‘yes’ or ‘no’)
fix[3]	char	Fix Subprocess (‘yes’ or ‘no’)
lvl[3]	char	Level Subprocess (‘yes’ or ‘no’)
test[3]	char	Test Subprocess (‘yes’ or ‘no’)
compName[63]	char	Component Name
userLogin[31]	char	Release Owner's CMVC User ID
userName[63]	char	Release Owner's Full Name
userArea[31]	char	Release Owner's Work Area or Department

SizeView‡

Sizing Records

Fieldname[length]	Datatype	Description
defectId	*number*	*database id of the feature or defect*
compId	*number*	*database id of the component associated with the sizing record*
userId	*number*	*database id of the user who owns the sizing record*
addDate[25]	char	Date Created
lastUpdate[25]	char	Date of Last Update
state[7]	char	Sizing Record State
sizing[127]	char	Text of Sizing Information
releaseId	*number*	*database id of the release associated with the sizing record*
featurePrefix[31]	char	Feature or Defect Prefix
featureName[31]	char	Feature or Defect Identifier
featureAbstract[127]	char	Feature or Defect Abstract
featureReference[31]	char	Feature or Defect Reference
compName[63]	char	Component Name
userLogin[31]	char	Size Record Owner's CMVC User ID
userName[63]	char	Size Record Owner's Full Name
userArea[31]	char	Size Record Owner's Work Area or Department
releaseName[31]	char	Release Name

TestView‡

Environment Test Records

Fieldname[length]	Datatype	Description
trackId	*number*	*database id of the track*
envName[31]	char	Environment Name
userId	*number*	*database id of the user who owns the test record*
state[15]	char	Environment Test Record State
addDate[25]	char	Date Created
lastUpdate[25]	char	Date of Last Update
defectPrefix[31]	char	Defect or Feature Prefix
defectName[31]	char	Defect or Feature Identifier
defectReference[31]	char	Defect or Feature Reference
defectAbstract[127]	char	Defect or Feature Abstract
defectType[7]	*char*	*'defect' or 'feature'*
userLogin[31]	char	Test Record Owner's CMVC User ID

Fieldname[length]	Datatype	Description
userName[63]	char	Test Record Owner's Full Name
userArea[31]	char	Test Record Owner's Work Area or Department
releaseName[31]	char	Release Name

TrackView‡

Track Properties

Fieldname[length]	Datatype	Description
id	*number*	*database id of the track*
releaseId	*number*	*database id of the release associated with the track*
defectId	*number*	*database id of the defect or feature*
userId	*number*	*database id of the user who owns the track*
state[15]	char	Track State
target[31]	char	Target for Completion
actual[31]	char	Name of Level where Track is Included
addDate[25]	char	Date Created
lastUpdate[25]	char	Date of Last Update
defectPrefix[31]	char	Defect or Feature Prefix
defectName[31]	char	Defect or Feature Identifier
defectReference[31]	char	Defect or Feature Reference
defectAbstract[127]	char	Defect or Feature Abstract
defectType[7]	*char*	*'defect' or 'feature'*
releaseName[31]	char	Release Name
userLogin[31]	char	Track Owner's CMVC User ID
userName[63]	char	Track Owner's Full Name
userArea[31]	char	Track Owner's Work Area or Department

VerifyView‡

Verification Record Properties

Fieldname[length]	Datatype	Description
defectId	*number*	*database id of the defect or feature*
userId	*number*	*database id of the user who owns the verification record*
type[15]	char	'original' or 'duplicate'
state[15]	char	Verification Record State
addDate[25]	char	Date Created
lastUpdate[25]	char	Date of Last Update
defectPrefix[31]	char	Defect or Feature Prefix

Fieldname[length]	Datatype	Description
defectName[31]	char	Defect or Feature Identifier
defectReference[31]	char	Defect or Feature Reference
defectAbstract[127]	char	Defect or Feature Abstract
userLogin[31]	char	Verification Record Owner's CMVC User ID
userName[63]	char	Verification Record Owner's Full Name
userArea[31]	char	Verification Record Owner's Work Area or Department
compName[63]	*char*	*name of component associated with the defect or feature*
defectType[7]	*char*	*'defect' or 'feature'*

Chapter 7. CMVC Tables for Queries

This chapter shows the field names for all of the possible table queries. The six tables you can query using **Report -view** are marked with a double dagger symbol(‡); the other tables are available only as subselect criteria. The fields are listed in the order in which they appear in the database. Field names in italics may be used as search criteria, but they do not provide data output. Specify values for date fields in the format *yy/mm/dd hh:mm:ss*.

When entering a query, do not abbreviate the field names. Depending on the database used by your installation, some abbreviations can be interpreted incorrectly and give inaccurate information. In most situations, you will receive only an error message if you abbreviate a field name.

For example, if your installation uses the ORACLE database and you replace the field names `userLogin` or `userName` with `user`, you do not receive an error message. The search, however, is ended with no records found.

Note: The INFORMIX data types are listed for the following tables. The ORACLE equivalents are integer for number, varchar for char, and text in table for long. The SYBASE equivalents are integer for number, varchar for char, and text for long. The DB2/6000 equivalents are integer for number, varchar for char, and long varchar for long.

If your family administrator configures the database fields, the following field names may differ from those in your database.

AccessTable

Fieldname[length]	Datatype	Description
compId	*number*	*database id of the component where access is granted or restricted*
userId	*number*	*database id of the user who owns the access authority*
authorityName[31]	*char*	*access authority group name*
authorityType[15]	*char*	*access authority type ('granted' or 'restricted')*

Approvals

Fieldname[length]	Datatype	Description
trackId	*number*	*database id of the track for the approval record*
userId	*number*	*database id of the user who owns the approval record*
state[15]	*char*	*approval record state*
addDate[25]	*char*	*date created*
lastUpdate[25]	*char*	*date of last update*

Approvers

Fieldname[length]	Datatype	Description
releaseId	*number*	*database id of the release*
userId	*number*	*database id of the user*

Authority‡

Fieldname[length]	Datatype	Description
name[31]	char	Access Authority Group Name
action[15]	char	Low Level Action Name

Cfgcomproc‡

Fieldname[length]	Datatype	Description
name[31]	char	Component Process Name
config[15]	char	Subprocess Name

Cfgrelproc‡

Fieldname[length]	Datatype	Description
name[31]	char	Release Process Name
config[15]	char	Subprocess Name

Changes

Fieldname[length]	Datatype	Description
pathId	*number*	*database id of the file's path name*
trackId	*number*	*database id of the track for this file change*
fileId	*number*	*database id of the file*
versionId	*number*	*database id of the file's version SID*
levelId	*number*	*database id of the level where the change is included and converted*
type[8]	*char*	*type of file change*
userId	*number*	*database id of the user who made the file changes*

CompMembers

Fieldname[length]	Datatype	Description
parentId	*number*	*database id of the parent component*
childId	*number*	*database id of the component*
rank	*number*	*number of links or generations between parent and child*
reference	*number*	*number of references to this parent-child-rank combination*

Components

Fieldname[length]	Datatype	Description
id	*number*	*database id of the component*
name[63]	*char*	*component name*
userId	*number*	*database id of the user who owns the component*
description[127]	*char*	*component description*
addDate[25]	*char*	*date created*
dropDate[25]	*char*	*date deleted*
lastUpdate[25]	*char*	*date of last update*
process[31]	*char*	*component process name*
featureDSR[3]	*char*	*feature DSR (design/size/review) subprocess*
featureVerify[3]	*char*	*feature verify subprocess*
defectDSR[3]	*char*	*defect DSR (design/size/review) subprocess*
defectVerify[3]	*char*	*defect verify subprocess (‘yes’ or ‘no’)*

Config‡

Fieldname[length]	Datatype	Description
type[15]	char	Name of Data Type
name[31]	char	Data Name
dflt[3]	char	Default Value (‘yes’ or ‘no’)
value1	number	(not currently used)
value2	number	(not currently used)
description[127]	char	Description of Data Name

Coreqs

Fieldname[length]	Datatype	Description
trackId	*number*	*database id of the track*
groupId	*number*	*database id of the corequisite group to which the track belongs*

Defects

(This table is also used to record features.)

Fieldname[length]	Datatype	Description
id	*number*	*database id of the defect or feature*
type[7]	*char*	*‘defect’ or ‘feature’*
prefix[31]	*char*	*defect or feature prefix*
name[31]	*char*	*defect or feature identifier*

Fieldname[length]	Datatype	Description
compId	*number*	*database id of the component associated with the defect or feature*
envName[31]	*char*	*environment where discovered (when type is 'defect', blank otherwise)*
state[15]	*char*	*defect or feature state*
severity[31]	*char*	*severity level (when type is 'defect')*
abstract[127]	*char*	*defect or feature abstract*
reference[31]	*char*	*defect or feature reference*
answer[31]	*char*	*accept or return answer type*
levelName[31]	*char*	*name of level where discovered (when type is 'defect')*
lastUpdate[25]	*char*	*date of last update*
addDate[25]	*char*	*date created*
assignDate[25]	*char*	*date when reassigned*
responseDate[25]	*char*	*date accepted or returned*
endDate[25]	*char*	*date closed or canceled*
originId	*number*	*database id of the user who opened the defect or feature*
ownerId	*number*	*database id of the defect or feature owner*
age	*number*	*age of defect or feature (dependent on the aging mechanism of the family)*
duplicate[31]	*char*	*defect or feature for which this defect or feature is a duplicate*
releaseId	*number*	*database id of the release where the defect is found (optional)*

Environments

Fieldname[length]	Datatype	Description
name[31]	*char*	*environment name*
releaseId	*number*	*database id of the release*
userId	*number*	*database id of the user*

Files

Fieldname[length]	Datatype	Description
id	*number*	*database id of the file*
compId	*number*	*database id of the component associated with the file*
sourceId	*number*	*database id of the file's source. Defines where it is derived from, and is used to identify common files*
pathId	*number*	*database id of the file's path name*
nuPathId	*number*	*database id of the file's pending path name*

Fieldname[length]	Datatype	Description
versionId	*number*	*database id of the current version of a file*
nuVersionId	*number*	*database id of the committed version of a file*
addDate[25]	*char*	*date created*
nuAddDate[25]	*char*	*pending creation date*
dropDate[25]	*char*	*date deleted*
nuDropDate[25]	*char*	*pending deletion date*
lastUpdate[25]	*char*	*date of last update*
baseName[127]	*char*	*file base name*
type[7]	*char*	*'text', 'binary', or 'long'*
releaseId	*number*	*database id of the release associated with the file*
userLogin[31]	*char*	*CMVC user ID who locked or checked out the file*
fmode[4]	*char*	*file mode*

FilesOut

Fieldname[length]	Datatype	Description
fileId	*number*	*database id of the file*
versionId	*number*	*database id of the file's version*
userId	*number*	*database id of the user who has the file locked*
checkOutDate[25]	*char*	*date file was locked*
newSID[47]	*char*	*file's pending version number on check in*

Fix

Fieldname[length]	Datatype	Description
trackId	*number*	*database id of the track*
userId	*number*	*database id of the user who owns the fix record*
compId	*number*	*database id of the component associated with the fix record*
state[15]	*char*	*fix record state*
addDate[25]	*char*	*date created*
lastUpdate[25]	*char*	*date of last update*

History

Fieldname[length]	Datatype	Description
defectId	*number*	*database id of the defect or feature*
userId	*number*	*database id of the user who caused the defect or feature state change*

Fieldname[length]	Datatype	Description
action[15]	*char*	*action that occurred and caused the state change*
addDate[25]	*char*	*date action occurred*

Hosts

Fieldname[length]	Datatype	Description
userId	*number*	*database id of the user*
name[127]	*char*	*client host name*
login[31]	*char*	*user's login name on client host*
address	*number*	*(not currently used)*

Interest‡

Fieldname[length]	Datatype	Description
name[31]	char	Interest Group Name
action[15]	char	Low Level Action Name

LevelMembers

Fieldname[length]	Datatype	Description
levelId	*number*	*database id of the level*
trackId	*number*	*database id of the track*

Levels

Fieldname[length]	Datatype	Description
id	*number*	*database id of the level*
name[31]	*char*	*level name*
releaseId	*number*	*database id of the release associated with the level*
userId	*number*	*database id of the user who owns the level*
addDate[25]	*char*	*date created*
commitDate[25]	*char*	*date committed*
lastUpdate[25]	*char*	*date of last update*
state[15]	*char*	*level state*
type[31]	*char*	*level type*
levelSize	*number*	*(not currently used)*

Notes

Fieldname[length]	Datatype	Description
defectId	*number*	*database id of the defect or feature*
userId	*number*	*database id of the user who added remarks*
action[15]	*char*	*action occurring when remarks were added*
addDate[25]	*char*	*date remarks added*
remarks[15999]	*long*	*text of remarks*

Notification

Fieldname[length]	Datatype	Description
compId	*number*	*database id of the component where notification is controlled*
userId	*number*	*database id of the user who owns the notification interest*
interestName[31]	*char*	*interest group name*

Path

Fieldname[length]	Datatype	Description
id	*number*	*database id of the file's path name*
name[195]	*char*	*full path name of the file*

Releases

Fieldname[length]	Datatype	Description
id	*number*	*database id of the release*
name[31]	*char*	*release name*
compId	*number*	*database id of the component associated with the release*
binding[3]	*char*	*binding status (no longer used)*
description[127]	*char*	*release description*
userId	*number*	*database id of the user who owns the release*
addDate[25]	*char*	*date created*
dropDate[25]	*char*	*date deleted*
lastUpdate[25]	*char*	*date of last update*
relSize	*number*	*(not currently used)*
process[31]	*char*	*release process name*
track[3]	*char*	*track subprocess (‘yes’ or ‘no’)*
approve[3]	*char*	*approval subprocess (‘yes’ or ‘no’)*
fix[3]	*char*	*fix subprocess (‘yes’ or ‘no’)*
lvl[3]	*char*	*level subprocess (‘yes’ or ‘no’)*
test[3]	*char*	*test subprocess (‘yes’ or ‘no’)*

Sequence

Fieldname[length]	Datatype	Description
name[15]	*char*	*sequence type*
lastSerial	*number*	*database id of the last sequence number assigned for the sequence type*

Sizes

Fieldname[length]	Datatype	Description
defectId	*number*	*database id of the defect or feature*
compId	*number*	*database id of the component associated with the sizing record*
userId	*number*	*database id of the user who owns the sizing record*
addDate[25]	*char*	*date created*
lastUpdate[25]	*char*	*date of last update*
state[7]	*char*	*sizing record state*
sizing[127]	*char*	*text of sizing information*
releaseId	*number*	*database id of the release associated with the sizing record*

Tests

Fieldname[length]	Datatype	Description
trackId	*number*	*database id of the track*
envName[31]	*char*	*environment name*
userId	*number*	*database id of the user who owns the test record*
state[15]	*char*	*environment test record state*
addDate[25]	*char*	*date created*
lastUpdate[25]	*char*	*date of last update*

Tracks

Fieldname[length]	Datatype	Description
id	*number*	*database id of the track*
releaseId	*number*	*database id of the release associated with the track*
defectId	*number*	*database id of the defect or feature*
userId	*number*	*database id of the user who owns the track*
state[15]	*char*	*track state*
target[31]	*char*	*target for completion (such as, a level or a date)*
actual[31]	*char*	*level commit date where track included (actual level name)*

Fieldname[length]	Datatype	Description
addDate[25]	*char*	*date created*
lastUpdate[25]	*char*	*date of last update*

Users‡

Fieldname[length]	Datatype	Description
id	*number*	*database id of the user*
login[31]	char	CMVC ID
name[63]	char	User's Full Name
superUser[3]	char	Superuser Privilege (‘yes’ or ‘no’)
area[31]	char	User's Work Area or Department
address[159]	char	User's Sendmail Address
addDate[25]	char	Date Created
dropDate[25]	char	Date Deleted
lastUpdate[25]	char	Date of Last Update

Versions

Fieldname[length]	Datatype	Description
id	*number*	*database id of the file version*
previousId	*number*	*database id of the file's previous version*
sourceId	*number*	*database id of the file's source. Defines where the file is derived from and is used to identify common files.*
userId	*number*	*database id of the user who created the file version*
rawLines	*number*	*(not currently used)*
codeLines	*number*	*(not currently used)*
commentLines	*number*	*(not currently used)*
remarks[15999]	*long*	*remarks added when the version was created*
changeDate[25]	*char*	*date the version was made*
SID[47]	*char*	*file version number*
verSize	*number*	*(not currently used)*

Verify

Fieldname[length]	Datatype	Description
defectId	*number*	*database id of the defect or feature*
userId	*number*	*database id of the user who owns the verification record*
type[15]	*char*	*‘original’ or ‘duplicate’*
state[15]	*char*	*verification record state*

Fieldname[length]	Datatype	Description
addDate[25]	*char*	*date created*
lastUpdate[25]	*char*	*date of last update*

Chapter 8. Sample Shell Scripts, Tutorial, and Demonstration

Figure 6 lists the sample shell scripts provided with the IBM CMVC products. The list is contained in the **client.samples** file in the **$CMVC_HOME/samples** directory on your client workstation.

Users running these scripts require user and host access to the CMVC family.

Most of the scripts require input parameters and some require that the `CMVC_FAMILY` or `CMVC_RELEASE` environment variables be set. If the user running the script is acting for another user, the `CMVC_BECOME` environment variable must also be set.

Programming convention has been used to show the required, optional, and selective input parameters.

Script Name	Function	Inputs	Environment Variables
allUsers	Lists the addresses of all users in a specified CMVC family.	familyName	[CMVC_BECOME]
cloneDefect	Creates a new defect based on values contained in a specified defect.	defectNumber	CMVC_FAMILY [CMVC_BECOME]
cloneDefFromFea	Creates a new defect based on values contained in a specified feature.	featureNumber	CMVC_FAMILY [CMVC_BECOME]
cloneFeature	Creates a new feature based on values contained in a specified feature.	featureNumber	CMVC_FAMILY [CMVC_BECOME]
cloneFeaFromDef	Creates a new feature based on the values contained in the specified defect.	defectNumber	CMVC_FAMILY [CMVC_BECOME]
cloneRelease	This script has been replaced by the **Release -link** action.	n/a	n/a
compAccess	Lists the explicit access of users in a specified component.	componentName	CMVC_FAMILY [CMVC_BECOME]
compChildren	Lists the direct children of a specified component. Also lists the description and owner of each child component.	componentName	CMVC_FAMILY [CMVC_BECOME]
compOwners	Displays a list of component owners' addresses in a specified family.	familyName	[CMVC_BECOME]
compWalk	Displays the children, and grandchildren of a specified component.	componentName	CMVC_FAMILY [CMVC_BECOME]
defFeaDesc	Displays the full remarks that were entered when the specified defect or feature was created.	{defectNumber | featureNumber}	CMVC_FAMILY [CMVC_BECOME]
defFeaFileChgs	Lists all the files that were changed for a specified defect or feature.	{defectNumber | featureNumber}	CMVC_FAMILY [CMVC_BECOME]

Figure 6 (Part 1 of 4). Sample Shell Scripts

Script Name	Function	Inputs	Environment Variables
defectReport	Generates a global defect report showing tracks, test records, approval records and fix records.		CMVC_FAMILY [CMVC_BECOME]
defectStats	Displays total active defect statistics on a defect owner area basis.	ownerArea	CMVC_FAMILY [CMVC_BECOME]
defectsInLevel	Lists all defects for a specified level.	levelName	CMVC_FAMILY [CMVC_BECOME]
defectsInState	Lists the defect number of all defects that are in a specified state.	stateName	CMVC_FAMILY [CMVC_BECOME]
defectsWithRef	Displays the full details of defects that contain the specified reference field value.	reference	CMVC_FAMILY [CMVC_BECOME]
featureReport	Generates a global feature report showing tracks, test records, size records and fix records.		CMVC_FAMILY [CMVC_BECOME]
featureStats	Displays total active feature statistics on a feature owner area basis.	ownerArea	CMVC_FAMILY [CMVC_BECOME]
featuresInLevel	Lists all features contained in a specified level.	levelName	CMVC_FAMILY [CMVC_BECOME]
featuresInState	Lists the feature number of all features that are in a specified state.	stateName	CMVC_FAMILY [CMVC_BECOME]
fileCheckIn	Checks files into the CMVC family. When common files are encountered, the script requests the releases for which the file should remain common.	filePathName [releaseName]	CMVC_FAMILY [CMVC_RELEASE] [CMVC_BECOME]
fileCompare	Gets a file, locks it, and compares it with a previous copy of the file in the same directory.	fileName [releaseName] [relativePathName]	CMVC_FAMILY [CMVC_RELEASE] [CMVC_BECOME]
fileHistory	Lists all defect and feature numbers and abstracts that caused a change to a specified file in a specified release.	fileName [releaseName]	CMVC_FAMILY [CMVC_RELEASE] [CMVC_BECOME]
fileInfo	Displays information for a specified file.	fileName	CMVC_FAMILY [CMVC_RELEASE] [CMVC_BECOME]
filesInComp	Lists all files related to a specified component.	componentName	CMVC_FAMILY [CMVC_BECOME]
filesInRelease	Lists all files related to a specified release.	releaseName	CMVC_FAMILY [CMVC_BECOME]
filesLockedBy	Lists all files that a specified user has locked.	userLogin	CMVC_FAMILY [CMVC_BECOME]
findFilePath	Finds and lists all files that match a partial file path name.	filePathName	CMVC_FAMILY [CMVC_BECOME]

Figure 6 (Part 2 of 4). Sample Shell Scripts

Script Name	Function	Inputs	Environment Variables
findUser	Finds user information based on part of the user's name. A fuzzy search is performed.	userName	CMVC_FAMILY [CMVC_BECOME]
getCompFiles	*Extracts* all the files associated with a specific component. The files are placed in a directory that represents the release name to which the version of the file is associated. This directory is created relative to the relativePathName parameter.	componentName relativePathName [committed]	CMVC_FAMILY [CMVC_BECOME]
getTrackFiles	Extracts all the files associated with a specific track and places them in the path specified by the relativePathName parameter.	releaseName defectNumber relativePathName	CMVC_FAMILY [CMVC_BECOME]
giveAccessAuth	Lists the users who have the authority to give other users access to a specified component.	componentName	CMVC_FAMILY [CMVC_BECOME]
levelFileChgs	Lists the files that were changed for a specified level.	levelName	CMVC_FAMILY [CMVC_BECOME]
levelHasDefFea	Lists the name of the levels that contain a specified defect or feature.	{defectNumber | featureNumber}	CMVC_FAMILY [CMVC_BECOME]
levelMembers	Lists the defect and feature members of a specified level for a specified release.	levelName [releaseName]	CMVC_FAMILY [CMVC_RELEASE] [CMVC_BECOME]
managerReport	Generates a manager's report based on the specified areas or departments of interest.	areaName ...	CMVC_FAMILY [CMVC_BECOME]
mktutorial	Populates a CMVC family for tutorial purposes. Refer to the file **tutorial.text** in the **$CMVC_HOME/samples** directory for set up instructions, tutorial exercises, and other related information		CMVC_FAMILY
newestDefect	Displays the number of the most recent defect that was entered in the system.		CMVC_FAMILY [CMVC_BECOME]
newestFeature	Displays the number of the most recent feature that was entered in the system.		CMVC_FAMILY [CMVC_BECOME]
parentComponent	Lists the parent component of a specified component.	componentName	CMVC_FAMILY [CMVC_BECOME]
reassignWork	Re-assigns all current work and objects owned by userLogin1 to userLogin2.	userLogin1 userLogin2	CMVC_FAMILY [CMVC_BECOME]
releaseOwners	Displays a list of addresses of all release owners in a specified CMVC family.	familyName	[CMVC_BECOME]
reopenDefect	Reopens a previously canceled or returned defect.	defectNumber	CMVC_FAMILY [CMVC_BECOME]
reopenFeature	Reopens a previously canceled or returned feature.	featureNumber	CMVC_FAMILY [CMVC_BECOME]
sendMailTo	Sends a message to the addresses read through stdin.	messagefile subject	

Figure 6 (Part 3 of 4). Sample Shell Scripts

Script Name	Function	Inputs	Environment Variables
showAllVersions	Lists the version numbers, release names and path names for the specified file.	fileName releaseName	CMVC_FAMILY [CMVC_BECOME]
showConfig	Lists the valid values pertaining to a specified configurable type.	configType	CMVC_FAMILY [CMVC_BECOME]
trackStats	Generates a track activity statistics report on a user area basis.	userArea	CMVC_FAMILY [CMVC_BECOME]
tracksInCommit	Lists the tracks that are in the commit state for a specified release.	releaseName	CMVC_FAMILY [CMVC_BECOME]
tracksInDevLvl	Lists the tracks that are in the integrate state and are associated with at least one development level for the specified release.	releaseName	CMVC_FAMILY [CMVC_BECOME]
tracksInFix	Lists all the tracks that are in the fix state for a given release.	releaseName	CMVC_FAMILY [CMVC_BECOME]
tracksInInt	Lists the tracks that are in the integrate state for a specified release.	releaseName	CMVC_FAMILY [CMVC_BECOME]
tracksInProdLvl	Lists the tracks that are included in a production level and are in the integrate state for a specified release.	releaseName	CMVC_FAMILY [CMVC_BECOME]
tracksInTest	Lists the tracks that are in the test state for a specified release.	releaseName	CMVC_FAMILY [CMVC_BECOME]
userAccess	Lists the explicit access of a specified user for the specified component and its descendant components.	userLogin componentName	CMVC_FAMILY [CMVC_BECOME]
userReport	Generates a user's report based on the specified user login.	userLogin	CMVC_FAMILY [CMVC_BECOME]
whoHasFile	Lists who has a specified file checked out.	fileName	CMVC_FAMILY CMVC_RELEASE [CMVC_BECOME]

Figure 6 (Part 4 of 4). Sample Shell Scripts

CMVC Tutorial

You can use the tutorial provided with the CMVC product to become familiar with CMVC's version control, problem tracking, and change control capabilities. The **$CMVC_HOME/samples** directory in the CMVC client contains the following files:

Filename	Description
mktutorial	A shell script that must be run from a CMVC client to initialize the tutorial.
tutorial.text	An instruction file containing start-up instructions and tutorial exercises.
tutorial.cmvcrc	A resource file for the CMVC GUI that contains tutorial-based tasks.

CMVC Demonstration

You can also use the demonstration provided with the CMVC product to become familiar with CMVC's version control, problem tracking, and change control capabilities. The CMVC demonstration differs from the tutorial in two aspects:

- The files are available on the CMVC server instead of the client
- There are no specific exercises to complete.

The following files for the demo are located in the **$CMVC_HOME/samples** on the CMVC server:

Filename	Description
mkdemo1.tar	A tar (archive) file containing all the required files and a script to populate a CMVC demonstration family.
README.demo1	A readme file containing instructions for setting up the demonstration family.

Chapter 9. Supported Keywords

The PVCS Version Manager keywords supported by the CMVC programs are listed in Figure 7.

Keyword	Description
$Archive$	The full path name of the Version Manager archive.
$Author$	The VCSID of the author of the revision. When expanded this will always be the name of the CMVC family.
$Date$	The check-in date of the revision.
$Header$	The archive name, revision number, revision date and author ID.
Log	Cumulative check-in messages.
$Modtime$	The time of the last modification.
$Revision$	The revision number.
$Workfile$	The file name stored in the Version Manager archive. The CMVC server expands this keyword to the composite: file name, component name, release name.

Note: The Log keyword differs from the other keywords in that its expansion text is not enclosed by dollar signs. Version Manager inserts the change description entered at check-in after the line that contains the Log keyword. The change descriptions accumulate in the workfile in reverse chronological order.

Figure 7. PVCS Version Manager Keywords

The SCCS keywords supported by the CMVC programs are listed in Figure 8.

Keyword	Description
%M%	The module name: the value of the m header flag in the SCCS file. Since the CMVC server defines numeric names when creating the SCCS files, expansion of this keyword will display the name of the file that the CMVC server defined.
%I%	The SID (%R%. %L. B%. %S%) of the g-file.
%R%	The SCCS Release.
%L%	The SCCS Level.
%B%	The SCCS Branch.
%S%	The SCCS Sequence.
%D%	The date of the current SCCS get (YY/MM/DD).
%H%	The date of the current SCCS get (MM/DD/YY).
%T%	The time of the current SCCS get (HH:MM:SS)
%E%	The date the newest applied delta was created (YY/MM/DD).
%G%	The date the newest applied delta was created (MM/DD/YY).
%U%	The time the newest applied delta was created (HH:MM:SS)
%F%	The SCCS file name. Since the CMVC server defines numeric names when creating the SCCS files, expansion of this keyword will display the names of the SCCS file that the CMVC server defined (for example, /u/<familyName>/vc/0/0/0/1/s.22).

Figure 8 (Part 1 of 2). SCCS Keywords

Keyword	Description
%P%	Full path name of the SCCS file. Since the CMVC server defines numeric names when creating the SCCS files, expansion of this keyword will display the name of the SCCS file that the CMVC server defined (for example, /u/<familyName>/vc/0/0/0/1/s.22).
%C%	The current line number. This keyword is for identifying messages output by the program. It is not intended to be used on every line to provide sequence numbers.
%A%	A shorthand notation for constructing what command strings for program files. Its value is the key letters: %A% = %Z%%Y% %M% %I%%Z%
%W%	A shorthand notation for constructing what command strings for program files. Its value is the characters and key letters: %W% = %Z%%M%<horizontal-tab>%I% The CMVC server expands this keyword to the composite value: path name, component name, release name **Note:** This keyword is not supported by the CMVC for Sun systems server code.
%Z%	The 4-character string @(#) recognized by the what command.

Figure 8 (Part 2 of 2). SCCS Keywords

Chapter 10. State Diagrams

This chapter contains the CMVC state diagrams which illustrate the movement of defects, features and tracks through the different states.

Defect and Feature State Diagram When Subprocesses Are Configured

Figure 9 shows the state transitions for features and defects when the related component processes include both the design, size, review (DSR) and verify subprocesses.

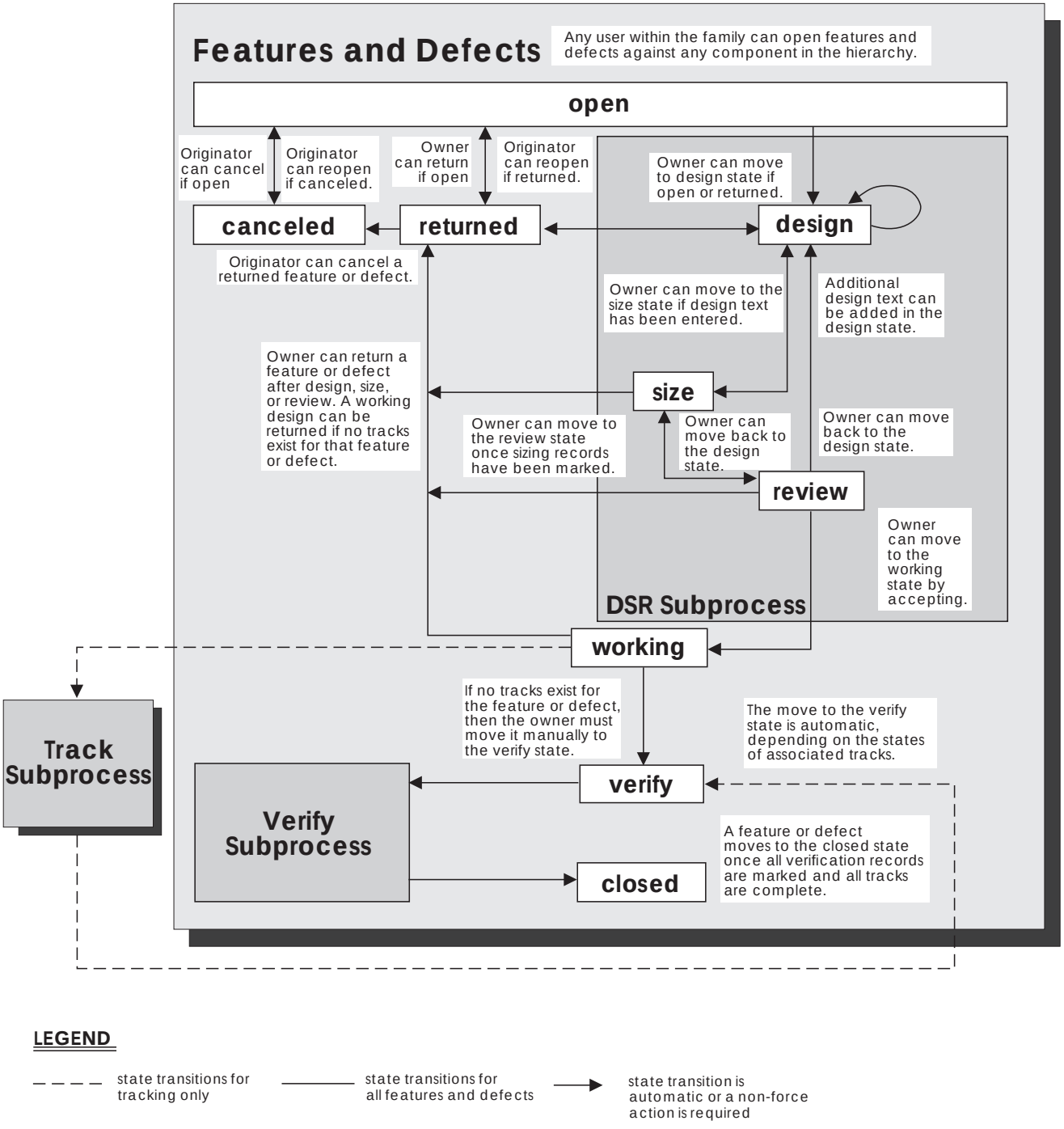

Figure 9. Defect and Feature State Diagram When Subprocesses Are Configured

Track and Level State Diagram When All Subprocesses Are Configured

Figure 10 shows the state transitions for a track that is associated with a release whose process includes the track, approval, fix, level, and test subprocesses.

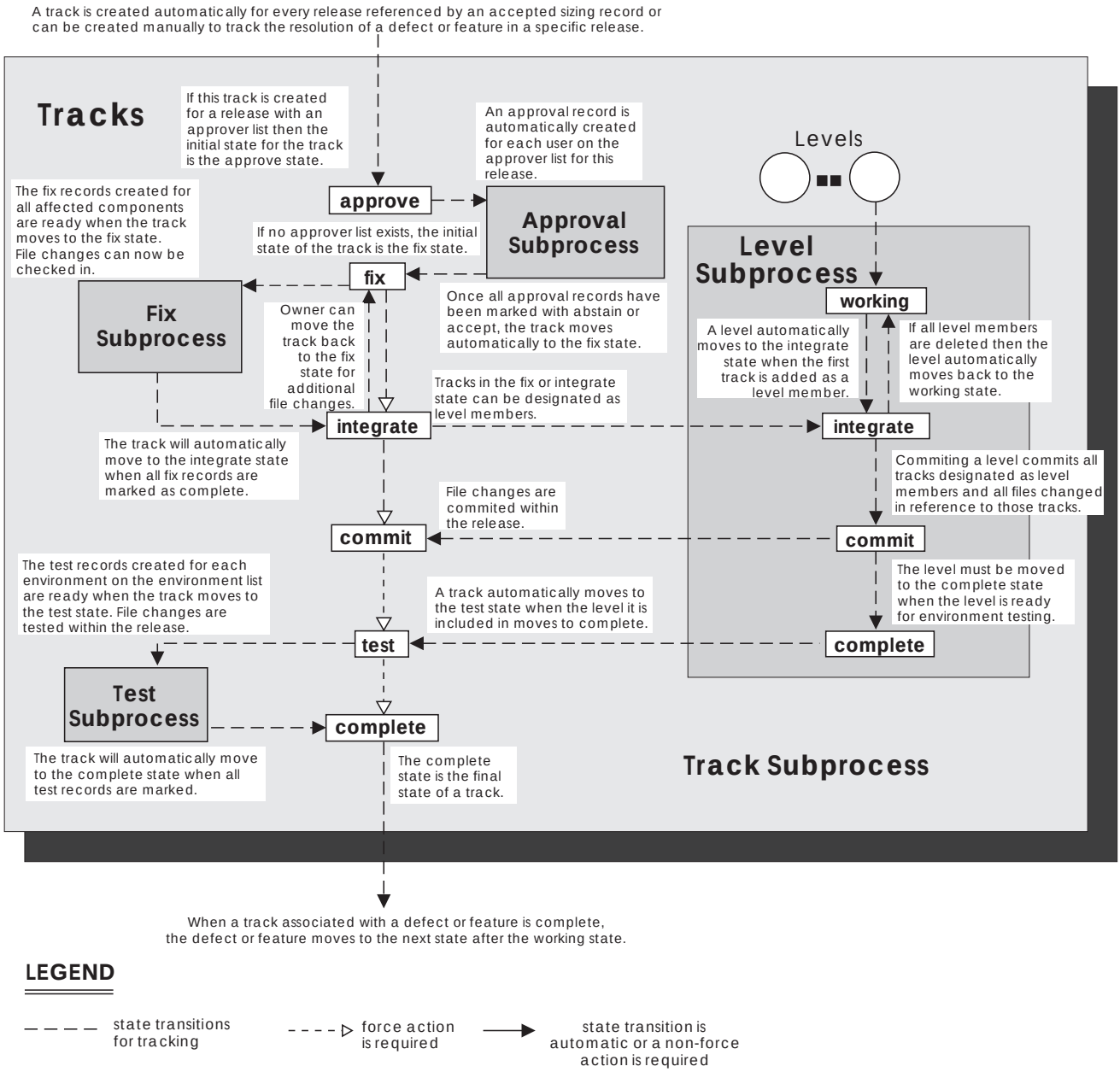

Figure 10. Track and Level State Diagram When All Subprocesses Are Configured

Track and Level State Diagram When Subprocesses Are Not Configured

Figure 11 shows the state transitions for a track that is associated with a release whose process does not include any subprocesses.

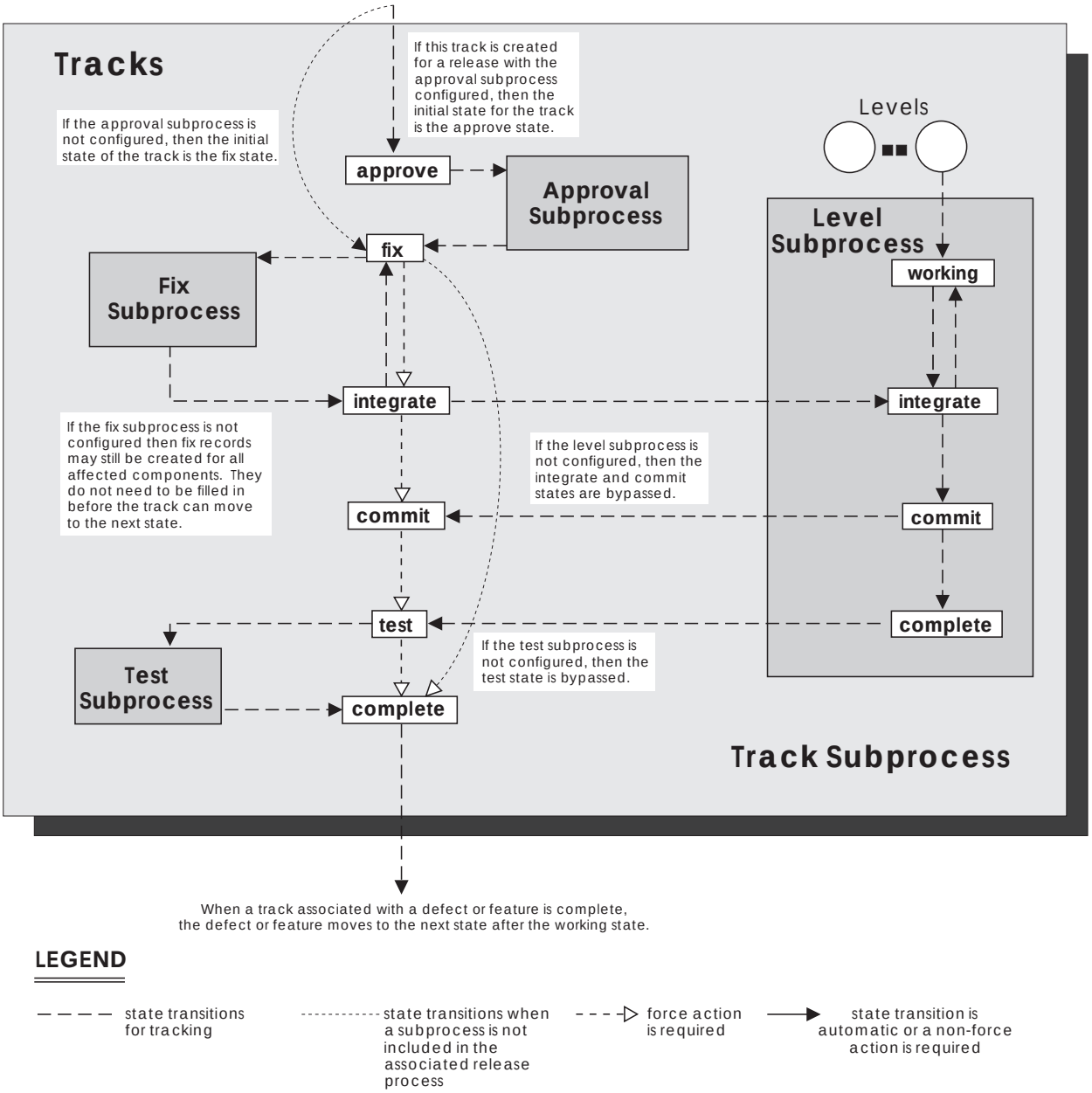

Figure 11. Track and Level State Diagram When Subprocesses Are Not Configured

CMVC State Diagram

Figure 12 shows the state transitions for defects, features, tracks, and levels including transitions when subprocesses are not configured for the release and component associated with the tracks, levels, defects, and features.

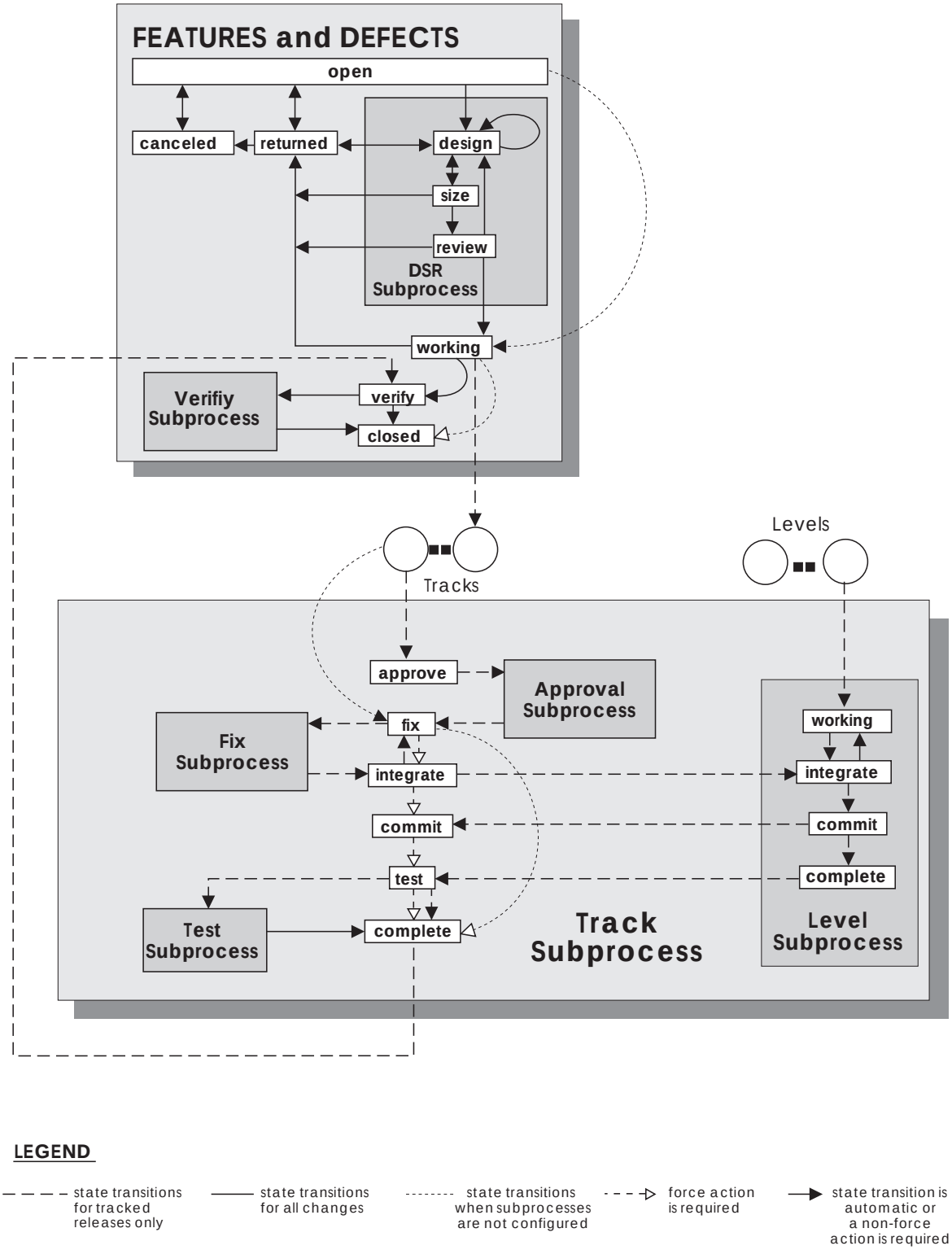

Figure 12. CMVC State Diagram

Part 2. The Message Interface

Part 2 describes how the message-integrated CMVC uses the Broadcast Message Server on SDE WorkBench/6000, HP SoftBench, and HP SoftBench for Sun tools to communicate with other development tools.

Use the information in this part if you are designing or developing new development tools to integrate and work with message-integrated CMVC and other SDE WorkBench/6000 or SoftBench tools.

Chapter 11. The Broadcast Message Server

The foundation of SDE WorkBench/6000 or SoftBench tools is the Broadcast Message Server (BMS). Each tool encapsulated within or fully integrated with SDE WorkBench/6000 or SoftBench tools must create and send messages in common formats to describe its activities. Messages are sent by the tools when actions are taken from their graphical user interfaces. The BMS then notifies other tools of these actions so that they can perform tasks dependent on these actions. For example, Figure 13 shows how message-integrated CMVC receives a request from other SDE WorkBench/6000 or SoftBench tools and notifies them of the actions it takes in response to the request.

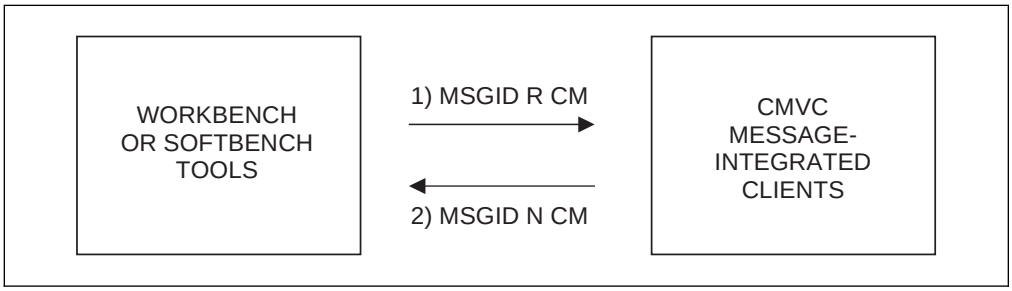

Figure 13. How the Tools Communicate

For example, a user selects a CMVC action from one of the SDE WorkBench/6000 or SoftBench tools, such as `Check In`... from the Development Manager. The Development Manager sends a request with MSGID for a Check-in operation using tool class CM. Message-integrated CMVC services the request and responds with a notification message, also with MSGID for the CM message class.

You can use the Message Monitor tool on SDE WorkBench/6000 or SoftBench tools to display all messages being broadcast between the tools. This tool helps you to develop and debug new tools that can work with message-integrated CMVC and other SDE WorkBench/6000 or SoftBench tools. Figure 14 shows the main window of the SDE WorkBench/6000 Message Monitor.

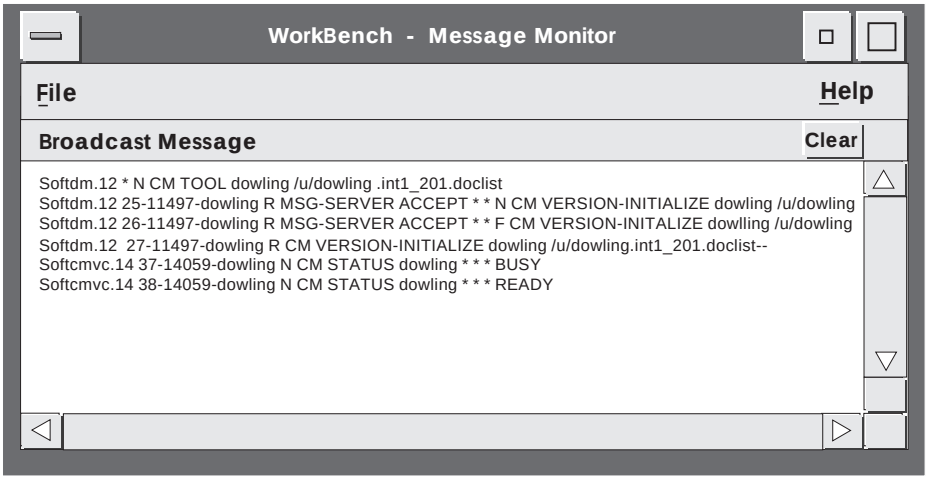

Figure 14. SDE WorkBench/6000 Message Monitor Window

Messages can also be sent directly from the File menu of the Message Monitor window. For a complete description of the message monitor and its use, see your SDE WorkBench/6000 or SoftBench tools documentation.

Scope

The *scope* of a BMS message is related to the *tool context*. Specifically, the term scope applies to certain fields in the message *context* (host, dir, file fields) that the Execution Manager uses when determining whether a particular tool can service a particular request. These fields are denoted by an asterisk (*). The scope can have one of the following values:

NET A tool with a NET scope handles all requests sent by a tool of the same class, regardless of the *message context*. For example, when you are using the Program Editor, the editor can service any request to edit a file located in any directory on any host in the network. This means that the Execution Manager does not need to match any of the three message context fields when it determines whether a particular tool session can handle a particular request for action. The tool context of a NET scope tool is:

```
* * *
```

HOST A tool with a HOST scope handles all requests for action from a tool of the same class, provided that the context host for the new request matches the context host for the current tool context. This means that the Execution Manager must match only the host part of the message context fields when it determines whether a particular tool session can handle a particular request for action. The tool context of a HOST scope tool is:

```
host * *
```

The message-integrated CMVC tool is a HOST scope tool. This means that the Tool Manager only starts one session of the message-integrated CMVC graphical user interface (*GUI*) per machine host. When you start WorkBench or SoftBench tools, the default context host is your local system.

DIR A tool with a DIR scope handles all requests for action from a tool of the same class, provided that the context host and the context directory for the new request match the context host and the context directory for the current tool context. This means that the Execution Manager must match both the HOST and DIR parts of the message context when it determines which tool session can handle a particular request for action. The tool context of a DIR scope tool is:

```
host dir *
```

The Development Manager is a DIR scope tool. This means that the Execution Manager only starts one session of the Development Manager graphical user interface per context directory. The directory from which you start the message-integrated tool is the default context directory.

FILE A tool with a FILE scope only handles requests for a tool of the same class, to service a file of the same host, directory, and file context as the current tool context. The tool context of a FILE scope tool is:

```
host dir file
```

The Program Debugger is a FILE scope tool. This means that Execution Manager must start a new session of the Program Debugger to handle a request for action sent to any file other than one that started an active tool session.

Format of BMS Messages

The Broadcast Message Server messages have the following format:

```
  0         1          2           3           4        5    6   7     8
sender message-id message-type tool-class message-name host dir file  data
```

Each of these fields is described in the following sections. The combination of fields 5, 6, and 7 (host, dir, and file) is the context. A message can contain one or more data fields starting with the field position 8. Field numbering begins with zero(0).

Message Field Delimiters

Message fields are delimited by white space: spaces, tabs, or new lines. When a tool is parsing a message, it can collect all the data fields at the end of a message and treat them as one logical field. The last data field can also be a NULL terminated string that can contain white-space characters.

Message Field Values

The characters * and - have special meanings when they are values in a message field.

***** An asterisk in a message field indicates that the field is not applicable to this message. CMVC does not accept this character in the pathname, component, or revision fields. For example, an asterisk in field 7 (file) means that the file portion of the path is not used in the message.

- A dash in the message field by the message sender indicates that the default value, if any, should be provided and used by the message receiver. The receiver should select the default. The actual value should be returned in the Notify or Failure response message. CMVC does not accept this value in the dir or file fields.

Message fields left off the end of a message should be treated as a dash (-) by the receiver.

Sender

The sender field contains the unique identifier for a running SDE WorkBench/6000 or SoftBench tool and is created by the Broadcast Message Server. It is made up of the X class name (the name of the application defaults file) and a connection-id number assigned by the *message server* when the tool is started. The message-integrated CMVC identifier is **Softcmvc**. If the sender is `Softcmvc.10`, the **softcmvc** tool with a message server connection-id of `10` is sending the message.

Message ID

The message-ID field contains a unique identifier for a message and any responses to that message. The WorkBench/6000 or SoftBench platform generates this ID. It is unique for each Request message and for each Notify or Failure message which is not a response to a Request message. A message-id usually has the form:

```
MessageNumber-ProcessID-HostName
```

The SDE Integrator/6000 or HP SoftBench Encapsulator provides a mechanism for encapsulated tools to conform to this standard.

Message Type

The three types of messages are Request messages (R), Notify messages (N), and Failure messages (F).

Request Messages

A Request Message (R) is a request by a tool to perform an action. If another tool is registered to receive the message, it must send either a Notify or Failure message to indicate the status of that action.

Notify Messages

A Notify Message (N) is a notification by a tool to other tools of the successful completion of a significant event.

The data fields should correspond to the fields in the Request message, except where a user has changed them in a dialog box. Other information may be added to the end of the Notify message. Any - placeholders in the Request message have the replacement values substituted for the - in the Notify response, where possible.

Note: Not all Notify messages have a matching Request message.

Failure Messages

A Failure Message (F) is a notification by a tool to other tools of the failure of actions that were requested by either a Request message or through an application's GUI.

The data fields of the Failure message correspond to the Request message, except where a user has changed them in a dialog box. Any - placeholders in the Request message have the replacement values substituted for the - in the Failure response, where possible. An error-message string is added to the end of the message to indicate the reason the action failed.

Tool Class

The tool-class field indicates the class of tool that responds to the message. Message-integrated CMVC is a tool of the Configuration Management (CM) class.

Message Name

The message-name is the name of the message. It indicates the action associated with the message, such as VERSION-CHECK-OUT.

Message Context (Host Dir File)

The context is the network specification of a file. It is made up of the host, dir, and file. dir is always an *absolute path name* of a directory on the host. file is either a path name relative to dir or an absolute path name on the host.

A context is important in two ways: it describes the location of data by forming a path specification to a particular file on a network, and it relates to the scope of a tool.

When a Request message is received by the SDE WorkBench/6000 or SoftBench platform for a tool class and context and no tool is registered for that combination, a new instance of the tool is started by the Execution Manager.

Data

Data fields contain additional information specific to the action of the message. Any number of data fields can follow the context fields.

There is a unique form of data field called the options data field. Message-integrated CMVC ignores any value in this field.

The last data field in a message can have a NULL terminated string with embedded blanks. This is the only field that can contain white space characters.

A string that is the last data field in a Request message is not included in the corresponding Notify or Failure message.

General BMS Request Messages

This section describes the Request messages that all tools integrated with SDE WorkBench/6000 or SoftBench tools must register to receive. Tools encapsulated using the IBM SDE Integrator/6000 or HP SoftBench Encapsulator automatically register to receive all of these Request messages.

Request messages can refer to one or all tools. To refer to all tools, set fields 3, 5, 6, and 7 to -. To refer to a particular tool, see field 3, which indicates the tool class, and fields 5, 6, and 7, which indicate the context.

Tools respond to a Request message (except for a STOP message) with either a Notify or a Failure message.

STOP

The STOP message requests that a tool close all its windows and dialog boxes and exit. If a tool is **BUSY**, the STOP will not be performed until the tool is no longer **BUSY**.

```
2      3         4      5  6  7       8
R  tool-class STOP     context    options
R      -      STOP     -  -  -
```

tool-class
The tool-class that receives the message. For example, CMVC is a tool of the **CM** Class and Development Manager is of the **DM** class.

A - in this field refers to all SDE WorkBench/6000 or SoftBench tools.

context The current default host, dir, and file.

Hyphens (-) in these fields refer to all tools.

options CMVC ignores any value in this field.

START

The START message requests that a tool start on the specified context. The Request message is directed to the SDE WorkBench/6000 or SoftBench platform.

```
2      3          4     5  6  7         8              9
R  tool-class  START  context   execution-host  options
```

Message-integrated CMVC is a tool with a HOST scope. It can handle requests for the CM tool class providing that the data resides on the given host. It sends a Failure message if the context host is not the same as the execution-host.

tool-class
The tool-class that receives the message.

context The current default host, dir, and file.

execution-host
If the execution-host is specified and a tool of the CM tool class is not already running in the specified context, the Tool Manager starts CMVC on the specified host.

options CMVC ignores any value in this field.

SET-CONTEXT

The SET-CONTEXT message is broadcast to change the default context of message-integrated CMVC to the newhost, newdir, and newfile values contained in the message.

A SET-CONTEXT Notify message indicates that the CM tool context has changed.

```
2      3             4        5  6  7     8        9       10       11
R tool-class   SET-CONTEXT  context  newhost newdir  newfile options
```

tool-class
The tool-class that receives the message. For example, CMVC is an application of the **CM** class.

context The current default host, dir, and file when a Request message is received or the previous state of the context when a Notify message is sent.

newhost The new host name requested.

newdir The new directory name requested.

newfile The new file name requested.

options CMVC ignores any value in this field.

STATUS

The STATUS message requests that a tool return its current status. For a complete description of this message, refer to “STATUS” on page 86.

```
2      3          4      5  6  7      8
R  tool-class  STATUS  context  filetypes
```

tool-class
The name of the tool-class that receives the message. For example, CMVC is an application of the **CM** class.

context Indicates the current default host, dir, and file.

file-types * or a list of | separated tokens representing the type of files on which the tool operates.

ICONIFY

The ICONIFY message is broadcast to iconify all windows of a tool. A Failure message is sent by tools that do not have a GUI.

```
2      3          4       5  6  7      8
R  tool-class  ICONIFY   context   options
R      -       ICONIFY   -  -  -
```

tool-class
The tool-class that receives the message. For example, the CMVC tool is an application of the **CM** class.

A hyphen (-) can be used to indicate all tools.

context The current default host, dir, and file.

A hyphen (-) can be used to indicate all tools.

options CMVC ignores any value in this field.

NORMALIZE

The NORMALIZE message requests that a tool return its windows to a normal size (the opposite of iconify).

A Failure message is sent by tools that do not have a GUI.

```
2      3           4       5  6  7     8
R  tool-class  NORMALIZE   context  options
R      -       NORMALIZE   -  -  -
```

tool-class
The name tool-class that receives the message. For example, CMVC is an application of the **CM** class.

A hyphen (-) can be used to indicate all tools.

context The current default host, dir, and file.

A hyphen (-) can be used to indicate all tools.

options CMVC ignores any value in this field.

General BMS Notify Messages

This section describes the Notify messages that every tool sends.

FILE-MODIFIED

A tool broadcasts the FILE-MODIFIED message to notify other SDE WorkBench/6000 or SoftBench tools that it has modified a file.

```
2      3             4          5  6  7      8
N  tool-class  FILE-MODIFIED  context  options
```

tool-class
The tool-class that receives the message. For example, CMVC is an application of the **CM** class.

context The file (or files) that has been modified.

The file in the context can be a non-directory file, the literal string *, or a directory path name that ends with /*. If the file is * or a directory path name that ends in /*, CMVC has created, deleted, or modified one or more items in the directory or subdirectory. This form is used when CMVC does not determine exactly which files have been modified, such as, when a large number of files have been modified, or after a LEVEL-EXTRACT or RELEASE-EXTRACT operation.

A file without a * indicates that a non-directory file has been created, deleted, or had its permissions or contents modified.

options CMVC ignores any value in this field.

STATUS

The STATUS Notify message tells the other SDE WorkBench/6000 or SoftBench tools of the status of the broadcasting tool. Tool Manager uses this information to maintain its status display and Execution Manager uses these messages to keep track of the tools that are registered to receive messages.

```
2      3          4       5  6  7        8          9
N  tool-class  STATUS   context   file-types  status
```

tool-class
The tool-class that receives the message.

context The current default host, dir, and file.

file-types Either the string * or a list of separated tokens from column 3 of the **softtypes** file representing the softtypes that the tool uses.

status The status is one of:

READY When the tool has first started and registered for all applicable messages or when the tool completes an action and is no longer **BUSY**. The Execution Manager can send any messages that caused it to start the tool or that may have arrived during startup.

BUSY Indicates a period of activity that prevents new messages from receiving actions from the tool.

STARTING
Broadcast by SDE WorkBench/6000 or SoftBench tools when the Execution Manager starts a new tool.

NOTRUNNING

Broadcast by SDE WorkBench/6000 or SoftBench tools when a tool disconnects from the platform.

DUPLICATE-TOOL

Broadcast by message-integrated CMVC when it detects a status message from a duplicate copy of itself.

Chapter 12. The CM Class Messages

Message-integrated CMVC is a tool of the Configuration Management (CM) class. This chapter describes the general SDE WorkBench/6000 and SoftBench CM class messages supported by message-integrated CMVC.

If you are designing or integrating a tool to work with CMVC in the SDE WorkBench/6000 or SoftBench environment, you must use this message interface. The three message formats (R, N, and F) for each message are shown and the values for each field are described.

For information about the dialog boxes that appear from the message-integrated CMVC GUI when CMVC receives Request messages, refer to the book *IBM CMVC User's Guide*.

CMVC Context Mapping

CMVC uses context mapping to relate the files, pathnames, and directories in Development Manager to the CMVC releases and components in the CMVC server.

All SDE WorkBench/6000 or SoftBench tools communicate file names using the context fields of BMS messages. These are the host, dir, and file fields. A context specifies the name of a file in a particular directory on a particular host.

Message-integrated CMVC, a CM tool, stores the versioned file in a separate area from the *working file*, that is, in the CMVC server. The working file is in one of the user's directories. The context for the message interface only refers to the working file, (or the client copy of the file). Message-integrated CMVC converts the directory and file parts of the working file context to an absolute path name (file name) that can be understood by the CMVC server. The following example shows how message-integrated CMVC converts the context of three files to absolute pathnames:

```
     Context                                  Absolute Path name
host: dir  file

harris: /home/harris/src  foo.c              /home/harris/src/foo.c
nadon: /home/nadon  ../../gui/cmvc/main.c    /gui/cmvc/main.c
gaitanos: /src/test/cmvc  /ui/test/cmvc.c    /ui/test/cmvc.c
```

Message-integrated CMVC then maps this absolute path name to an associated CMVC path name, release and component on the CMVC server according to the context mapping rules. Each mapping that you set in the mapping table consists of a directory prefix, a release, and a component. The context mapping rules are:

1. If a directory prefix value can be mapped to a valid prefix of the lead portion of the absolute path name, CMVC uses that value as the directory prefix and also uses the associated values for the release and component.

In addition, the CMVC path name is set to:

```
absolute path name - directory prefix
```

For example, if the absolute path name of a file is `/home/nadon/ui/filel.c`, and you specify the following mapping:

```
directory prefix: /home/nadon
release: rel1.2
component: CompB
```

The result of the mapping is as follow:

```
directory prefix      CMVC pathname     release      component

 /home/nadon           ui/file1.c       rel1.2         CompB
```

2. If the above condition is not met, then the directory prefix is set to the directory portion of the absolute path name of the file. The CMVC path name is set to the base name of the file. The values for release and component are set to null.

   For example, if the absolute pathname is `/home/nadon/ui/file1.c`, the following applies:

```
directory prefix   CMVC pathname         release      component

 /home/nadon/ui      file1.c             (null)        (null)
```

   Note: In the two cases described above, the source directory or destination directory (whichever applies to the particular action) is set to the value of the directory prefix.

You can set up to 100 context mappings in message-integrated CMVC. For more information on setting the context mapping, refer to “SET-MAPPING” on page 104.

General BMS Messages for the CM Class Tools

These messages are common to many SDE WorkBench/6000 or SoftBench tools of the CM class. Message-integrated CMVC supports all of these messages.

VERSION-INITIALIZE

Create a new versioned file on the CMVC server.

```
2   3          4          5  6  7    8       9
R  CM VERSION-INITIALIZE  context options comment
N  CM VERSION-INITIALIZE  context options comment
F  CM VERSION-INITIALIZE  context options error-message
```

The VERSION-INITIALIZE message is broadcast to request that a new file be created on the CMVC server. The context file becomes the initial version. A copy of the file remains on the workstation.

A FILE-MODIFIED Notify message is broadcast with the regular VERSION-INITIALIZE Notify message when the operation succeeds.

The context file in the working directory is read-only after it is created on the server.

A Failure message is broadcast if the file does not exist or if the versioned file cannot be created.

context Specifies the working file that becomes the versioned file.

options CMVC ignores any value in this field. If the options field is followed by a comment, a - should be specified in the Request message.

comment This is a description of the file in the Request message.

The comment can contain up to 15 999 characters and can include imbedded spaces and new line characters. If the comment is the literal string -, CMVC prompts the user for a comment before performing the initialize operation.

The Notify message returns the comment supplied in the Request message, or as entered or changed in the Remarks field of the dialog box.

The Failure message removes the comment supplied in the Request message and replaces it with an error message.

error-message The Failure message uses this field to specify the reason the request failed.

VERSION-CHECK-OUT

Check out a file version.

```
2  3          4          5  6  7  8     9       10        11
R CM VERSION-CHECK-OUT context rev options keyword
N CM VERSION-CHECK-OUT context rev options keyword
F CM VERSION-CHECK-OUT context rev options keyword error-message
```

The VERSION-CHECK-OUT message is broadcast to transfer a file from the CMVC server to the location specified in the message context. A working version of the file is copied from the CMVC server. You can lock the versioned file, or make the file writable using the keyword field.

If the file is modified in any way (including its permissions), a FILE-MODIFIED Notify message is broadcast with that file in the context field.

If the versioned file does not exist or if the user is not authorized to check out the file, the request to check out the file fails. The Failure message includes the reason for the failure in the error-message field.

context Specifies where the working file is placed.

rev The name of the revision to be checked out.

rev is only used when the value in the keyword field is **CO**. If rev is specified when keyword is **LOCK** or **CO-LOCK**, a Failure message is sent.

rev cannot contain embedded spaces because it must appear as a single field. If rev is - or missing, the latest version is checked out. If it is not - or missing, rev must be a string that is understood by the CMVC server, for example, 1.1 or 1.2.1.1. If the CMVC server cannot recognize the value, the Request will fail.

If a tool does not use the rev field, it should pass - to CMVC in this field.

A tool should be able to store and reuse the rev string passed back by CMVC without knowing or understanding the contents of the string. CMVC defines the contents of the string. This is often the string that was passed back in a previous VERSION-CHECK-IN Notify message.

The Notify message contains the actual string used by CMVC to retrieve the versioned file.

options CMVC ignores any value in this field. If the options field is followed by a keyword, a - should be specified in the Request message.

keyword One of **CO-LOCK** (default), **CO**, or **LOCK**.

If keyword is **CO-LOCK**, the string -, or missing, CMVC updates the context file from the CMVC server. If the file does not already exist, it is created. If it is read-only, the file is updated and renamed to .filename. For example, if the file name is `foo.c`, the renamed file is `.foo.c`.

If keyword is the string -, or missing, CMVC returns the keyword **CO-LOCK** in the Notify or Failure message.

After a **CO-LOCK** operation, the context file is writable and locked. This prevents other users from locking or checking out that file version. If the file is already locked, the VERSION-CHECK-OUT fails.

If keyword is **CO**, the CMVC tool updates the context file from the CMVC server. If the file does not already exist, it is created. If the file is read-only, it is updated and renamed to .filename. For example, if the file name is `foo.c`, the renamed file is `.foo.c`.

The file is read-only after a successful **CO** operation.

If the keyword is **LOCK**, the CMVC server marks the context file as locked to prevent another user from checking or locking that version of the file. The file is not updated. If the file is already locked, the VERSION-CHECK-OUT fails.

error-message The Failure message uses this field to specify the reason the request failed.

VERSION-CHECK-IN

Check the working file into the CMVC server.

```
2  3        4          5  6  7   8     9      10        11
R CM VERSION-CHECK-IN  context  rev options keyword comment
N CM VERSION-CHECK-IN  context  rev options keyword comment
F CM VERSION-CHECK-IN  context  rev options keyword error-message
```

The VERSION-CHECK-IN message is broadcast to store the working context file in the CMVC server.

Message-integrated CMVC creates the next version of the versioned file from the working file. The versioned file is unlocked or made read-only depending on the value in the keyword field.

The request fails if the file does not exist or cannot be checked in as requested (the reason appears in the error-message field).

If the file has been modified in any way, including its access authorities, a FILE-MODIFIED Notify message is broadcast with the message context of that file.

context Specifies where the working file is.

rev Set to - or missing in the Request message.

After a successful CHECK-IN operation, the Notify message will place a string into this field. This string, if passed in the rev field of a future VERSION-CHECK-OUT message, will cause this newly checked-in file to be retrieved. A tool should be able to store and reuse the version string passed by CMVC without knowing or understanding the contents of the string.

CMVC defines the contents of the version string. It cannot contain imbedded spaces.

options CMVC ignores any value in this field. If the options field is followed by a keyword, a - should be specified in the Request message.

keyword One of **CO** (default) or **CANCEL**.

If keyword is the string -, missing, or **CO**, message-integrated CMVC checks the context file into the CMVC server and releases all locks. A copy of the checked-in file remains on the workstation and is read-only.

If keyword is **CANCEL**, the file is not checked in and the file lock is released. The contents of the working file will remain the same, but the mode changes to read-only.

comment This field is ignored if the keyword is **CANCEL**.

In the Request message, the comment describes the changes made to the file. CMVC associates this field with the version on the CMVC server.

The comment can contain up to 15 999 characters and can include imbedded spaces and new line characters. If the comment is the literal string -, CMVC prompts the user for a comment before performing the check-in operation.

The Notify message returns the comment supplied in the Request message, or as entered or changed in the Remarks field of the dialog box.

The Failure message removes the comment supplied in the Request message and replaces it with an error message.

error-message The Failure message uses this field to specify the reason the request failed.

VERSION-UPDATE-DIR

Update the versioned files in the context directory with any newer versions.

```
2  3         4           5  6  7     8      9         10
R  CM VERSION-UPDATE-DIR  context keyword options
N  CM VERSION-UPDATE-DIR  context keyword options
F  CM VERSION-UPDATE-DIR  context keyword options error-message
```

The VERSION-UPDATE-DIR message is broadcast to update the versioned files associated with the current release and mapped to the directory specified by the context directory and the context file with any newer versions that may exist on the CMVC server. The current release determined by the context mapping rules is the release on which the query/update operation is performed.

The Request message copies (extracts) into the context directory all the files on the CMVC server that *map* to the context directory and are missing or read-only.

Files that are already extracted but not checked out (files that are read-only) are updated.

A file that already exists in the context directory and is writable is not updated because any user changes to the file would be lost.

Updated files are extracted and are read-only. Message-integrated CMVC broadcasts a FILE-MODIFIED Notify message for each file. It also broadcasts a VERSION-CHECK-OUT Notify message for each file that is successfully extracted.

A VERSION-CHECK-OUT Failure message is broadcast for each file that is not successfully extracted.

If no files match the results of the query to the server, a VERSION-UPDATE-DIR Notify message is broadcast indicating the successful completion of the operation.

The VERSION-UPDATE-DIR message only succeeds if all files mapped to the context directory and its subdirectories are updated successfully. The error-message field of the Failure message contains all the errors associated with each VERSION-CHECK-OUT failure.

context The specification of the directory to update.

The file part of the context is the literal string * or a subdirectory path ending in /*.

keyword One of **CURRENT** (default), **RECURSIVE**.

If keyword is missing, the string - , or **CURRENT** is specified only the specific context directory gets updated.

If keyword has the value **RECURSIVE** the update is performed *recursively* on the context directory and all its subdirectories.

options CMVC ignores any value in this field.

error-message The Failure message uses this field to specify that the request failed.

VERSION-COMPARE-REVS

Compare two versions of the context file.

```
2  3          4           5  6  7   8   9       10         11        12
R CM VERSION-COMPARE-REVS context rev1 rev2 result-file options
N CM VERSION-COMPARE-REVS context rev1 rev2 result-file options result
F CM VERSION-COMPARE-REVS context rev1 rev2 result-file options error-message
```

The VERSION-COMPARE-REVS message is broadcast to compare two versions of the versioned file to which the context file is mapped.

Message-integrated CMVC can use any compare command that produces no output when the two files are identical. The commands **diff**, **bdiff**, or **sdiff** with a **-s** flag are all acceptable.

If the comparison is performed successfully, a Notify message is broadcast with an additional field (field 12) appended to the end. If the 2 files are identical, this field contains the string **IDENTICAL**. If they are not, it contains the string **DIFFERENT**.

If **sdiff** is used as the compare command with no **-s** flag, there will not be an **IDENTICAL** Notify message because an output file is produced.

If the result-file is successfully written, CMVC broadcasts a FILE-MODIFIED Notify message for that file.

context The file to be compared.

rev1 The name of the first version to compare. If the string is the literal - or missing, the current version is used in the comparison. Otherwise, a string representing a valid revision of a *CMVC file* on the CMVC server must be used.

rev2 The name of the second version to compare. If the string is the literal - or missing, the context file is used in the comparison. If file is a relative path, it together with the host and dir fields of the context specify the workstation file to be used in the comparison. If file is an absolute path, it alone specifies the workstation file to be used in the comparison.

Otherwise, a string representing a valid revision of a CMVC file on the CMVC server must be used.

result-file The name of the file with the results of the comparison. If this is * , - , or missing, the results are displayed interactively on the screen. If result-file is a relative path, it together with the host and dir fields of the context specify the location of the comparison output. If result-file is an absolute path, it alone specifies the location of the comparison output.

If result-file is missing or the string * in the Request message, a - is returned in the Notify or Failure message.

If the result-file is specified, it will be created regardless of whether the files are identical or different. If the result-file is read-only, the Request will fail. If the result-file is successfully written, a FILE-MODIFIED Notify message is broadcast for it.

options CMVC ignores any value in this field.

result If the results of the comparison show that the two files have no differences, the value is **IDENTICAL**.

If there are any differences in the files, the value is **DIFFERENT**.

error-message The Failure message uses this field to specify the reason the request failed.

VERSION-LIST-DIR

List all of the versioned files in the CMVC server that are contained in the directory specified by the context directory and the context file.

```
2   3         4        5  6  7     8          9       10         11
R  CM VERSION-LIST-DIR context result-file keywords options
N  CM VERSION-LIST-DIR context result-file keywords options
F  CM VERSION-LIST-DIR context result-file keywords options error-message
```

The VERSION-LIST-DIR message is broadcast to list all the versioned files from the CMVC server associated with the current release that map to the directory specified by the context directory and the context file. The current release as determined by the context mapping rules is the release on which the query is performed.

The output returned by this message is returned in a CMVC - Information window unless an output file (result-file) is specified.

context The specification of the directory to list. The file component of the context must be the literal * or a subdirectory path ending in /*.

result-file The name of the file where the directory list is placed. If this is missing, *, or - , the results are displayed interactively on the screen. If result-file is a relative path, it together with the host and the dir components of the context specifies the location of the list output. If result-file is an absolute path, it alone specifies the location of the list output.

If result-file is missing or the string * in the Request message, a - is returned in the Notify or Failure message.

If the result-file is read-only, the Request will fail. If the result-file is successfully written, a FILE-MODIFIED Notify message is broadcast for it.

keyword One of **CURRENT** (default) or **RECURSIVE**.

If keyword is missing, the string - , or **CURRENT**, the files listed are those in the specified context directory. **CURRENT** will be returned in the Notify message.

If keyword has the value **RECURSIVE** the list is performed recursively on the context directory and all its subdirectories.

options CMVC ignores any value in this field.

error-message The Failure message uses this field to specify the reason the request failed.

VERSION-SHOW-HISTORY

Show the history of the versioned file that corresponds to the context file.

```
2  3           4           5  6  7      8         9        10
R  CM VERSION-SHOW-HISTORY context result-file options
N  CM VERSION-SHOW-HISTORY context result-file options
F  CM VERSION-SHOW-HISTORY context result-file options error-message
```

The VERSION-SHOW-HISTORY message is broadcast to get a history of the versioned file that maps to the context file. The history contains all relevant information, such as, who created the version, the comments this person provided, and the date of creation of each version.

The output returned by this message is returned in a CMVC - Information window unless an output file (result-file) is specified.

If the result-file is successfully written, message-integrated CMVC broadcasts a FILE-MODIFIED Notify message for that file.

context The specification of the file whose history you want to see.

result-file The name of the file with the history information. If this is *, -, or missing, the results are displayed interactively on the screen. If result-file is a relative path, it together with the host and dir fields of the context specify the location of the history output. If result-file is an absolute path, it alone specifies the location of the history output.

If result-file is * or missing in the Request message, a - is substituted in the Notify or Failure message.

If the result-file is read-only, the request will fail. If the result-file is successfully written, a FILE-MODIFIED Notify message is broadcast for it.

options CMVC ignores any value in this field.

error-message The Failure message uses this field to specify the reason the request failed.

Chapter 13. Tool-Specific CM Messages

The previous chapters describe the way that message-integrated CMVC implements the general BMS messages and general CM class messages. This chapter describes the additional version control CM messages specific to message-integrated CMVC. You can use these messages to integrate a new tool with the CMVC and the SDE WorkBench/6000 or SoftBench tools.

VERSION-LINK

Link a version of a file in one release on the CMVC server to another release.

```
2  3       4         5  6  7     8     9   10          11
R  CM VERSION-LINK   context release rev options
N  CM VERSION-LINK   context release rev options
F  CM VERSION-LINK   context release rev options error-message
```

The VERSION-LINK request message is broadcast to link a version of the context file in one release to another release on the CMVC server. The Link Files dialog box appears.

The file is created in the new release upon successful operation.

A Failure message is issued if there is no context file on the CMVC server or if the versioned file cannot be linked.

context The file to be linked.

release The context file links to this new release.

rev The revision that is the base for the link. It cannot contain embedded spaces because it must appear as a single field to SDE WorkBench/6000 or SoftBench tools. If rev is - or missing, the latest or (right) version is linked.

If it is not -, rev must be a string that is understood by CMVC. Otherwise, the request will fail.

If the requester does not use this field, a - should be passed to CMVC. After a successful VERSION-LINK operation, the Notify message contains the actual rev string used by CMVC.

options CMVC ignores any value in this field.

VERSION-UNDO

Remove the changes made to the current version of a file on the CMVC server.

```
2   3    4          5  6  7    8         9
R  CM VERSION-UNDO   context options
N  CM VERSION-UNDO   context options
F  CM VERSION-UNDO   context options error-message
```

The VERSION-UNDO request message is broadcast to undo all the changes made to the current version of the context file. The Undo Files dialog box appears. The changes are undone for the versioned file on the CMVC server.

A Failure message is broadcast if the file does not exist or if the changes cannot be undone.

context The file whose changes must be undone.

options CMVC ignores any value in this field.

error-message The Failure message uses this field to specify the reason the request failed.

VERSION-RECREATE

Re-create a previously deleted file on the CMVC server.

```
2   3        4          5  6  7    8         9
R  CM VERSION-RECREATE  context options
N  CM VERSION-RECREATE  context options
F  CM VERSION-RECREATE  context options error-message
```

The VERSION-RECREATE Request message is broadcast to re-create a previously deleted context file on the CMVC server. The Re-create Files dialog box appears.

The request fails if the file does not exist or if it cannot be recreated.

context The file to be recreated.

options CMVC ignores any value in this field.

error-message The Failure message uses this field to specify the reason the request failed.

VERSION-DELETE

Delete the versioned file on the CMVC server. The file can be recreated.

```
2  3        4         5  6  7    8          9
R  CM  VERSION-DELETE  context  options
N  CM  VERSION-DELETE  context  options
F  CM  VERSION-DELETE  context  options  error-message
```

The VERSION-DELETE Request message is broadcast to delete the context file. This file can be recreated. The Delete Files dialog box appears.

The request fails if the file does not exist or if it cannot be deleted.

context The file to be deleted.

options CMVC ignores any value in this field.

error-message The Failure message uses this field to specify the reason the request failed.

VERSION-DESTROY

Destroy a CMVC versioned file. All file and version information is removed from the CMVC server database tables.

If the file is part of a release whose process includes the track subprocess, you can only *destroy* it if it has been deleted previously.

```
2   3        4         5  6  7    8         9
R  CM VERSION-DESTROY  context options
N  CM VERSION-DESTROY  context options
F  CM VERSION-DESTROY  context options error-message
```

The VERSION-DESTROY Request message is broadcast to remove the context file from the CMVC server. The Destroy Files dialog box appears.

The request fails if the file does not exist or if it cannot be destroyed.

context The file to be destroyed.

options CMVC ignores any value in this field.

error-message The Failure message uses this field to specify the reason the request failed.

VERSION-MODIFY-NAME

The VERSION-MODIFY-NAME Request message is broadcast to change the path name of the CMVC file on the server. The Modify Path Name dialog box appears.

```
2  3            4        5  6  7    8       9         10
R CM VERSION-MODIFY-NAME context pathname options
N CM VERSION-MODIFY-NAME context pathname options
F CM VERSION-MODIFY-NAME context pathname options error-message
```

The request fails if the file does not exist or if the path name cannot be modified.

context The path name of this file will be modified.

pathname The new path name for the file on the CMVC server.

If pathname is missing or the string -, CMVC prompts you for the new value.

If pathname is the string *, the request will fail.

options CMVC ignores any value in this field.

error-message The Failure message uses this field to specify the reason the request failed.

VERSION-MODIFY-COMP

Modify the component that manages the file on the CMVC server.

```
2  3            4          5  6  7     8        9        10
R  CM VERSION-MODIFY-COMP  context component options
N  CM VERSION-MODIFY-COMP  context component options
F  CM VERSION-MODIFY-COMP  context component options error-message
```

The VERSION-MODIFY-COMP Request message is broadcast to change the component that manages a CMVC file. The Modify Component dialog box appears.

context The file whose component is modified.

component The name of the new component that manages the file on the CMVC server.

If component is missing or the string -, CMVC prompts you for the new value.

If component is the string *, the request will fail.

options CMVC ignores any value in this field.

error-message The Failure message uses this field to specify the reason the request failed.

RELEASE-EXTRACT

Extract a release from the CMVC server and place it in the directory to which the context maps.

```
2   3         4          5  6  7     8         9          10
R   CM   RELEASE-EXTRACT  context  keywords  options
N   CM   RELEASE-EXTRACT  context  keywords  options
F   CM   RELEASE-EXTRACT  context  keywords  options error-message
```

The RELEASE-EXTRACT Request message is broadcast to extract (copy) the files currently associated with the release on the CMVC server to the directory. The directory and release fields of the Extract Release dialog box are determined by the context mapping rules. The Extract Releases dialog box appears. (For more information on context mapping, see “CMVC Context Mapping” on page 89).

A Failure message will be issued if the release does not exist. After a successful RELEASE-EXTRACT operation, a FILE-MODIFIED Notify message is sent with a context directory of the base directory to which the extraction was performed.

context The specification of the directory into which the files are extracted. The file component of the context must be the literal string * or a subdirectory path name ending in /*.

keywords One of **CURRENT** (default), **COMMITTED**, or **DATE**.

If keyword is missing, - , or **CURRENT**, the directory and subdirectories get updated with the current version of files within the release on the CMVC server. The keyword **CURRENT** is returned in the Notify or Failure message.

If keyword has the value **COMMITTED**, the directory and all its subdirectories are updated with the committed version of the files within the release on the CMVC server.

If keyword has the value **DATE**, the directory and all its subdirectories are updated with versions of the files changed within the release on the CMVC server since the date supplied by the user in the Extract Release dialog box. This dialog box appears with the Date button pushed in when the Request is received.

options CMVC ignores any value in this field.

error-message The Failure message uses this field to specify the reason the request failed.

LEVEL-EXTRACT

Extract a level from the CMVC server and place it in the directory to which the context maps.

```
2  3       4       5  6  7   8      9      10         11
R CM LEVEL-EXTRACT context level keywords options
N CM LEVEL-EXTRACT context level keywords options
F CM LEVEL-EXTRACT context level keywords options error-message
```

The LEVEL-EXTRACT Request message is broadcast to extract (copy) the files that are currently associated with the specified level for the release on the CMVC server to the directory. The directory and release fields of the Extract Levels dialog box are determined by the context mapping rules. The Extract Levels dialog box appears. (For more information on context mapping, see “CMVC Context Mapping” on page 89).

A Failure message is issued if the release does not exist.

After a successful LEVEL-EXTRACT operation, a FILE-MODIFIED Notify message is sent with a context directory of the base directory to which the extraction was performed.

context The specification of the directory into which the files are extracted. The file component of the context must be the literal string * or a subdirectory path name ending in /*.

level The name of the level containing the files on the CMVC server to be extracted.

keywords One of **DELTA** (default) or **FULL**.

If keyword is missing, the - , or **DELTA**, the directory and subdirectories are updated with the version of files that have changed since the last committed level within the release and that are included as part of the level on the CMVC server. The keyword **DELTA** is returned in the Notify or Failure message.

If keyword is missing or -, **DELTA** is returned in the Notify or Failure message.

If keyword has the value **FULL**, the directory and all its subdirectories are updated with the version of the files within the specified release and the level on the CMVC server.

options CMVC ignores any value in this field.

error-message The Failure message uses this field to specify the reason the request failed.

SET-MAPPING

Set the context mapping based on the directory specified by the context directory and the context file.

```
2  3     4          5       6        7        8        9
R CM SET-MAPPING context release component options
N CM SET-MAPPING context release component options
F CM SET-MAPPING context release component options error-message
```

The SET-MAPPING Request message is broadcast to specify the context mapping that is saved in the user database. The directory used in the mapping table is resolved from the context directory and context file. The Set Context Mapping dialog box appears. If you have specified the release and component, you can accept or change the prefilled values. If you have not specified the release and component, type in the values for these fields. If you want to remove a context mapping, leave the release and component fields blank. (For additional information on context mapping, see “CMVC Context Mapping” on page 89).

context The specification of the directory to map. The file component of the context must be the literal string * or a subdirectory path name ending in /*.

release The name of the release to be mapped to the resolved context directory and context file.

If the release is missing or the string -, the CM tool prompts for the new value.

If the release is the string *, a Failure message is returned.

component The name of the component to be mapped to the resolved context directory and context file.

If component is missing or the string -, the CM tool prompts for the new value.

If component is the string *, a Failure message is returned.

options CMVC ignores any value in this field.

error-message The Failure message uses this field to specify the reason the request failed.

VERSION-FILES-LOCKED

List the files locked or checked out in a release.

```
2   3           4          5  6  7     8          9        10
R  CM VERSION-FILES-LOCKED context options result-file
N  CM VERSION-FILES-LOCKED context options result-file
F  CM VERSION-FILES-LOCKED context options result-file error-message
```

The VERSION-FILES-LOCKED Request message is broadcast to list the files that are currently locked or checked out on the CMVC server and are associated with the current release. The output returned by this message is returned in a CMVC - Information window unless an output file (result-file) is specified. A failure message is issued if the release does not exist.

context The specification of the directory to list. The file component of context must be the literal string * or a subdirectory path name ending in /*.

options CMVC ignores any value in this field. If the options field is followed by a result-file, a - should be specified in the Request message.

result-file The name of the file in which to place the file list. If the string is -, *, or missing, the results will be displayed interactively on the screen. If result-file is a relative path, it, together with the host and dir components of context, specify where to place the output of the comparison. If result-file is an absolute path, it alone specifies where to place the output of the operation.

If result-file is the string * or missing in the Request message, the string - will be substituted in the corresponding Notify or Failure message.

If the result-file is read-only, the request will fail. If the result-file is successfully written, a FILE-MODIFIED Notify message is broadcast for it.

error-message The Failure message uses this field to specify the reason the request failed.

WINDOW

The WINDOW Request message is broadcast to open the appropriate main window of the message-integrated CMVC GUI. For a brief description of the tasks that you can complete from each window, refer to the *IBM CMVC User's Guide*.

```
2  3    4    5  6  7     8         9        10
R  CM WINDOW context window-name options
N  CM WINDOW context window-name options
F  CM WINDOW context window-name options error-message
```

context The tool context of CMVC. The host portion of the context is the host that the tool was started on. The directory and file portions of the context are *.

window-name The name of the main window of the CMVC GUI to open. The following is a list of the valid window-name entries and the CMVC window that appears:

If the window-name is missing, the string -, or **softcmvc**, the main Tasks window appears. The string **softcmvc** is returned in the Notify or Failure message.

If the window-name is **access**, the CMVC - Access Lists window appears.

If the window-name is **approval**, the CMVC - Approval Records window appears.

If the window-name is **approver**, the CMVC - Approver Lists window appears.

If the window-name is **change**, the CMVC - Change History window appears.

If the window-name is **component**, the CMVC - Components window appears.

If the window-name is **CompTreeView**, the CMVC - Component Tree window appears.

If the window-name is **defect**, the CMVC - Defects window appears.

If the window-name is **feature**, the CMVC - Features window appears.

If the window-name is **environment**, the CMVC - Environment Lists window appears.

If the window-name is **file**, the CMVC - Files window appears.

If the window-name is **FileTreeView**, the CMVC - File Change History window appears.

If the window-name is **fix**, the CMVC - Fix Records window appears.

If the window-name is **host**, the CMVC - Host Lists window appears.

If the window-name is **level**, the CMVC - Levels window appears.

If the window-name is **LevelTreeView**, the CMVC - Level Change History window appears.

If the window-name is **level_member**, the CMVC - Level Members window appears.

If the window-name is **notification**, the CMVC - Notification Lists window appears.

If the window-name is **release**, the CMVC - Releases window appears.

If the window-name is **sizing**, the CMVC - Sizing Records window appears.

If the window-name is **test**, the CMVC - Test Records window appears.

If the window-name is **track**, the CMVC - Tracks window appears.

If the window-name is **user**, the CMVC - Users window appears.

If the window-name is **verification**, the CMVC - Verification Records window appears.

options CMVC ignores any value in this field.

error-message The Failure message uses this field to specify the reason the request failed.

Glossary

Glossary terms are defined as they are used in this manual. If you cannot find the term for which you are looking, refer to the *IBM Dictionary of Computing*, SC20-1699.

A

absolute path name. A directory or a file expressed as a sequence of directories followed by a file name beginning from the root directory.

access list. A CMVC object that controls access to development data. A list of user ID-authority group pairs attached to a component, designating users and the corresponding authority access they are being granted for all objects managed by this component or any of its descendants. It also contains the user ID-authority group pairs designating users who are restricted from performing actions at a specific component.

action. A task performed by the CMVC server and requested by a CMVC client. A CMVC action corresponds to issuing one CMVC command.

approver. A user who approves changes within a specific release.

approver list. A list of user IDs attached to a release, representing the users who must approve file changes required to resolve a defect or implement a feature in that release.

authority. The right to access development objects and perform CMVC commands. See also *access list, base authority, explicit authority, implicit authority, restricted authority* and *superuser privilege*.

B

base authority. The set of actions granted to a user whenever a user ID is created within a CMVC family.

base file name. The name assigned to the file outside of the CMVC server environment, excluding any directory names.

Broadcast Message Server. A facility that coordinates the SDE WorkBench/6000 or HP SoftBench tools. Messages from tools are sent to the Broadcast Message Server which routes 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 an individual release can be integrated with problem tracking by specifying a process for that release that includes the track 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. All components in each CMVC family, with the exception of the root component, must be created in reference to an existing component. The existing component is referred to as the parent component, while the new component becomes known as the child component. A parent component can have more than one child component. See also *component*.

client. A workstation that requests services from another workstation.

CMVC file. A file that is stored by the CMVC server and retrieved by a path name. See also *file, common file, shared file*.

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 contained in 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 simplifies project management, organizes project data into structured groups, and controls configuration management properties. Component owners can control access to development data (see *access list*) and configure notification about CMVC actions (see *notification list*). Components exist in a parent-child hierarchy, with descendent components inheriting access and notification information from ancestor components.

configuration management. The process of identifying, managing, and controlling software modules as they change over time.

context. A description of a data file or directory in the form *host dir file*. That is the host machine, working directory, and file. See also *tool context* and *message context*.

current directory. The directory which is displayed in the directory list.

D

defect. A CMVC object used to formally report a problem. The user who opens a defect is the defect originator.

delete. Deleting a development object, such as, a file or a user ID, within CMVC. Certain objects can be deleted only if certain criteria are met. Most objects that are deleted can be re-created.

destroy. The only CMVC development object that can be destroyed in CMVC is a file. Destroying a file removes the file record from the database on the CMVC server. Though a destroyed file cannot be re-created, it will appear as part of an extracted level.

directory file list. A list of files and sub-directories of the current working directory displayed in the Development Manager main window.

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 per environment.

explicit authority. The ability to perform an action against a CMVC object because you have been granted the authority to perform that action.

extract. A CMVC action you can perform on a file, level, or release. A file extraction results in the specified file being copied to the client workstation. A level extraction and release extraction result in copying the files associated with the level or release to a designated workstation.

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 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. For example, source code, executable programs, documentation, or test cases. See also *common file, shared file*.

fix record. A status record that is associated with a track and is used to monitor the phases of change within each component that is affected by a defect or feature for a specific release.

G

GUI. The OSF/Motif**-based CMVC graphical user interface program.

H

host. Host node, host computer, or host system.

host list. A list associated with each CMVC user ID which indicates the client hosts that can access the CMVC server and act on behalf of the CMVC user. The list is used by the CMVC server to authenticate the identity of a CMVC client upon receipt of a CMVC command. Each entry consists of a login, a CMVC user ID, and a host name.

I

implicit authority. The ability to perform an action against a CMVC object without being granted explicit authority. This authority is implicitly granted due to object ownership. Contrast with *explicit authority* and *base authority*.

inheritance. The passing of configuration management properties from parent component to child component. The configuration management properties that are inherited are access and notification. Inheritance within a component hierarchy is cumulative.

L

level. A collection of tracks which represent a set of changed files within a release.

level member. A track that has been added to a level.

lock. Prevent editing access to a file stored within the CMVC development environment so that only one user can make changes to a given file at one time.

login. Operating system user identification.

M

map. The process of reassigning the meaning of an object.

message context. The Broadcast Message Server uses a context field to pass file name information between the various SDE WorkBench/6000 or HP SoftBench tools. The context is made up of the host, dir and file.

message server. The facility that coordinates the SDE WorkBench/6000 or HP SoftBench tools. It receives messages from tools and routes them to other tools.

N

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 they created the object or because they were assigned ownership of that object.

P

parent component. See *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 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*.

R

recursive. A recursive program calls itself or is called by a subroutine which it calls.

relative path name. The name of a directory or a file expressed as a sequence of directories followed by a file name, beginning from the current directory.

release. A CMVC object defined by a user to group all 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.

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.

S

scope. A parameter in the TOOL statement for each SDE WorkBench/6000 or HP SoftBench tool. It defines the fields in the message context used by the Execution Manager to determine whether a particular tool can service a particular request.

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*.

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. A user who is granted superuser privilege. Superuser 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 being used by the CMVC server.

T

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.

tool. An SDE WorkBench/6000 or HP SoftBench application.

tool context. The range of data for which a tool is registered to receive requests and perform actions.

track. A CMVC object created to monitor the progress of changes within a release to resolve a specific defect or implement a specific feature.

U

user. A person with an active user ID and access to one or more CMVC families.

V

verification record. A status record which must be marked by the originator of a defect or a feature before the defect or feature can move to the closed state. This allows the originator 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 alternate and temporary representation of data from one or more tables.

W

working file. The currently checked-out version of a CMVC file.

Printed in U.S.A.