DEVELOPING SOFTWARE WITH CLEARCASE

Release 4.1 and later

UNIX Edition

Rational®
the e-development company™

800-023555-000

Developing Software with ClearCase
Document Number 800-023555-000 August 2000
Rational Software Corporation 20 Maguire Road Lexington, Massachusetts 02421

IMPORTANT NOTICE

Copyright Notice

Copyright © 1992, 2000 Rational Software Corporation. All rights reserved.
Copyright 1989, 1991 The Regents of the University of California
Copyright 1984–1991 by Raima Corporation
Copyright 1992 Purdue Research Foundation, West Lafayette, Indiana 47907

Trademarks

Rational, the Rational logo, Atria, ClearCase, ClearCase MultiSite, ClearCase Attache, ClearDDTS, ClearQuest, ClearGuide, PureCoverage, Purify, Quantify, Rational Rose, and SoDA are trademarks or registered trademarks of Rational Software Corporation in the United States and in other countries. All other names are used for identification purposes only and are trademarks or registered trademarks of their respective companies.

Microsoft, MS, ActiveX, BackOffice, Developer Studio, Visual Basic, Visual C++, Visual InterDev, Visual J++, Visual Studio, Win32, Windows, and Windows NT are trademarks or registered trademarks of Microsoft Corporation.

Sun, Solaris, and Java are trademarks or registered trademarks of Sun Microsystems, Inc.

Oracle and Oracle7 are trademarks or registered trademarks of Oracle Corporation.

Sybase and SQL Anywhere are trademarks or registered trademarks of Sybase Corporation.

U.S. Government Rights

Use, duplication, or disclosure by the U.S. Government is subject to restrictions set forth in the applicable Rational License Agreement and in DFARS 227.7202-1(a) and 227.7202-3(a) (1995), DFARS 252.227-7013(c)(1)(ii) (Oct 1988), FAR 12.212(a) 1995, FAR 52.227-19, or FAR 52.227-14, as applicable.

Patent

U.S. Patent Nos. 5,574,898 and 5,649,200 and 5,675,802. Additional patents pending.

Warranty Disclaimer

This document and its associated software may be used as stated in the underlying license agreement, and, except as explicitly stated otherwise in such license agreement, Rational Software Corporation expressly disclaims all other warranties, express or implied, with respect to the media and software product and its documentation, including without limitation, the warranties of merchantability or fitness for a particular purpose or arising from a course of dealing, usage or trade practice.

Technical Acknowledgments

This software and documentation is based in part on BSD Networking Software Release 2, licensed from the Regents of the University of California. We acknowledge the role of the Computer Systems Research Group and the Electrical Engineering and Computer Sciences Department of the University of California at Berkeley and the Other Contributors in its development.

This software and documentation is based in part on software written by Victor A. Abell while at Purdue University. We acknowledge his role in its development.

This product includes software developed by Greg Stein <gstein@lyra.org> for use in the mod_dav module for Apache (http://www.webdav.org/mod_dav/).

Rational®
the e-development company™

Preface

Rational ClearCase is a comprehensive configuration management (CM) system that manages multiple variants of evolving software systems. ClearCase maintains a complete version history of all software development artifacts, including code, requirements, models, scripts, test assets, and directory structures. It performs audited system builds, enforces site-specific development policies, offers multiple developer workspaces, and provides advanced support for parallel development. ClearCase includes Unified Change Management (UCM), an optional, out-of-the-box process for organizing software development teams and their work products.

About This Manual

This manual guides software developers through everyday development tasks using either UCM or customizable features of ClearCase.

Organization

The manual is divided into two parts:

- *Working in UCM*. Read this part only if your development team uses UCM to implement its development process.
- *Working in Base ClearCase*. Read this part only if your development team uses customizable ClearCase features to implement its own development process.

Recommended Reading Paths

Use this manual as a guide during your first few weeks of developing software with ClearCase. We recommend this sequence for proceeding:

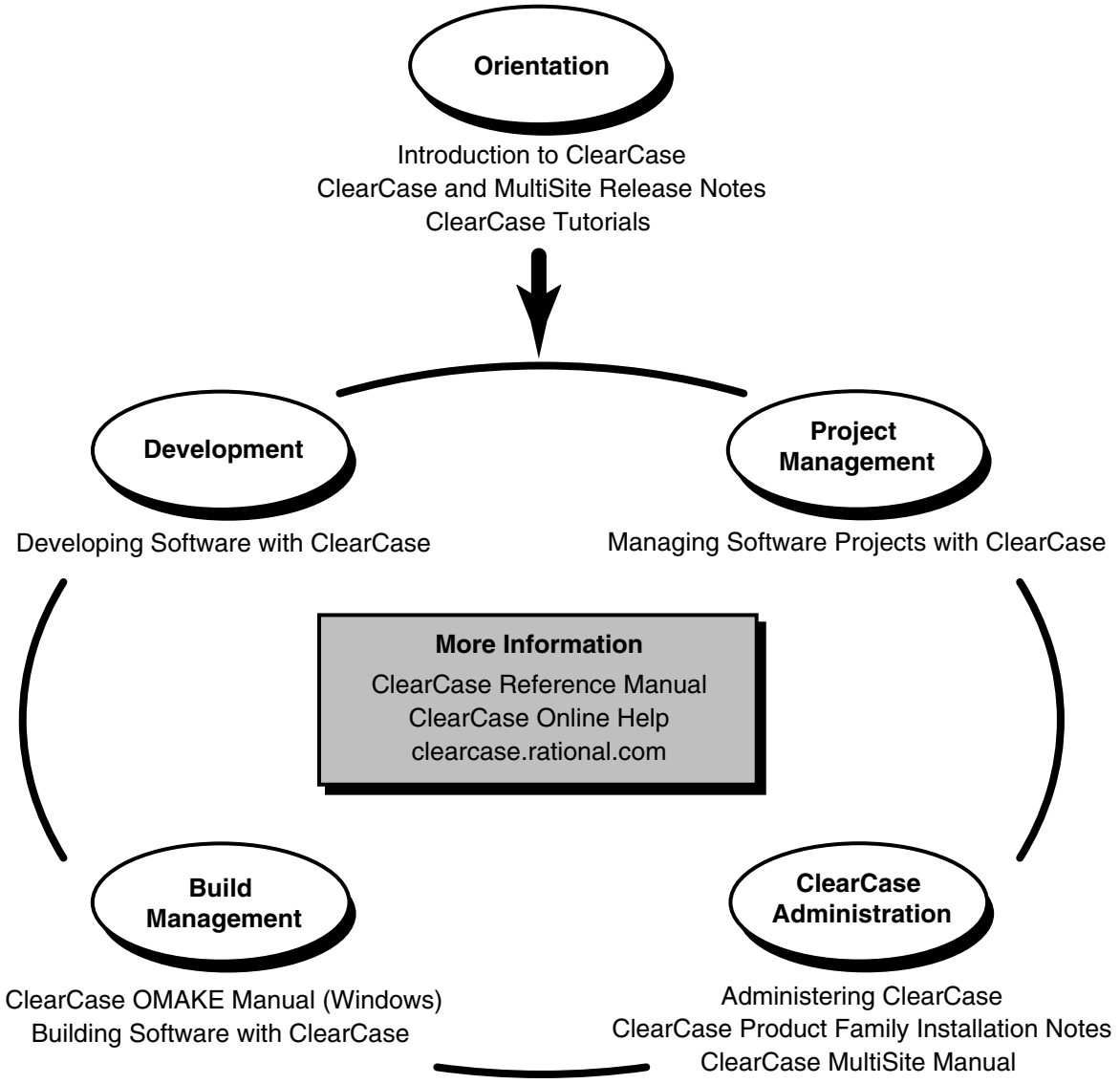
- Skim through this manual to understand at what points in the development cycle you'll need to use ClearCase. Consider skipping the sections titled *Under the Hood*, which describe advanced concepts and technical details.
- Read through this manual carefully and follow the procedures.
- As you become familiar with ClearCase, read the sections titled *Under the Hood*.

Assumptions

This manual assumes:

- ClearCase has been installed and configured on your workstation. For information on installing ClearCase, see *ClearCase Product Family Installation Notes*.
- One or more *versioned object bases* (VOBs), which are the repositories for ClearCase data, exist in your organization.
- If your development team uses ClearCase UCM, your project manager has set up a UCM *project*.
- Your organization has established some other development strategy using base ClearCase as the configuration-management system, and you are familiar with this strategy.

ClearCase Documentation Roadmap



Typographical Conventions

This manual uses the following typographical conventions:

- *ccase-home-dir* represents the directory into which the ClearCase Product Family has been installed. By default, this directory is **/usr/atria** on UNIX and **C:\Program Files\Rational\ClearCase** on Windows.
- *attache-home-dir* represents the directory into which ClearCase Attache has been installed. By default, this directory is **C:\Program Files\Rational\Attache**, except on Windows 3.*x*, where it is **C:\RATIONAL\ATTACHE**.
- **Bold** is used for names the user can enter; for example, all command names, file names, and branch names.
- *Italic* is used for variables, document titles, glossary terms, and emphasis.
- `A monospaced font` is used for examples. Where user input needs to be distinguished from program output, **bold** is used for user input.
- Nonprinting characters are in small caps and appear as follows: **<EOF>**, **<NL>**.
- Key names and key combinations are capitalized and appear as follows: SHIFT, CTRL+G.
- [ ] Brackets enclose optional items in format and syntax descriptions.
- { } Braces enclose a list from which you must choose an item in format and syntax descriptions.
- | A vertical bar separates items in a list of choices.
- ... In a syntax description, an ellipsis indicates you can repeat the preceding item or line one or more times. Otherwise, it can indicate omitted information.

  NOTE: In certain contexts, ClearCase recognizes "**...**" within a pathname as a wildcard, similar to "*" or "?". See the **wildcards_ccase** reference page for more information.
- If a command or option name has a short form, a "medial dot" (·) character indicates the shortest legal abbreviation. For example:

  lsc·heckout

  This means that you can truncate the command name to **lsc** or any of its intermediate spellings (**lsch**, **lsche**, **lschec**, and so on).

Online Documentation

The ClearCase graphical interface includes a Microsoft Windows-like help system.

There are three basic ways to access the online help system: the **Help** menu, the **Help** button, or the F1 key. **Help>Contents** provides access to the complete set of ClearCase online documentation. For help on a particular context, press F1. Use the **Help** button on various dialog boxes to get information specific to that dialog box.

ClearCase also provides access to full "reference pages" (detailed descriptions of ClearCase commands, utilities, and data structures) with the **cleartool man** subcommand. Without any argument, **cleartool man** displays the **cleartool** overview reference page. Specifying a command name as an argument gives information about using the specified command. For example:

`%` **cleartool man** *(display the cleartool overview page)*

`%` **cleartool man man** *(display the cleartool man reference page)*

`%` **cleartool man checkout** *(display the cleartool checkout reference page)*

ClearCase's **–help** command option or **help** command displays individual subcommand syntax. Without any argument, **cleartool help** displays the syntax for all **cleartool** commands. **help checkout** and **checkout –help** are equivalent.

```
% cleartool lsprivate –help
Usage: lsprivate [-tag view-tag] [-invob vob-selector] [-long | -short]
                 [-size] [-age] [-co] [-do] [-other]
```

Additionally, the online *ClearCase Tutorial* provides important information on setting up a user's environment, along with a step-by-step tour through ClearCase's most important features. To start the *ClearCase Tutorial* from the command line, type **hyperhelp cc_tut.hlp**.

Technical Support

If you have any problems with the software or documentation, please contact Rational Technical Support via telephone, fax, or electronic mail as described below. For information regarding support hours, languages spoken, or other support information, click the **Technical Support** link on the Rational Web site at **www.rational.com**.

Your Location	Telephone	Facsimile	Electronic Mail
North America	800-433-5444 toll free or 408-863-4000 Cupertino, CA	408-863-4194 Cupertino, CA 781-676-2460 Lexington, MA	**support@rational.com**
Europe, Middle East, and Africa	+31-(0)20-4546-200 Netherlands	+31-(0)20-4546-201 Netherlands	**support@europe.rational.com**
Asia Pacific	61-2-9419-0111 Australia	61-2-9419-0123 Australia	**support@apac.rational.com**

Working in UCM

Contents

Preface iii
About This Manual iii
Organization iii
Recommended Reading Paths iv
Assumptions iv
ClearCase Documentation Roadmap v
Typographical Conventions vi
Online Documentation vii
Technical Support vii

Working in UCM

1. The UCM Workflow 1
Recommended Reading Paths 1
1.1 The UCM Workflow 2
Setting Up Work Areas 2
Finding and Setting Activities 3
Finding Activities 3
Setting Activities 3
Working on Activities 4
Checking Out and Modifying Source Files 4
Viewing Metadata 4
Checking In and Testing Source Files 4
Indicating Your Progress 5
Delivering Activities 5
Preparing Your Work Areas 5
Starting the Deliver Operation 5
Merging 6
Testing 6

Completing the Operation 6
Delivering with MultiSite 6
Rebasing Your Development Work Area 7
1.2 Under the Hood: ClearCase UCM Concepts 7
Projects 8
Components and Baselines 8
Activities 9
Views and Streams 10
Views 11
Streams 11
Development Stream 13
The Integration Stream 13
Delivering, Creating Baselines, and Rebasing 15
1.3 Under the Hood: UCM ClearQuest Concepts 17
The UCM-ClearQuest Schema 17
UCM-Enabled Record Types and ClearCase Activities 17
Queries 18
MyToDoList 19
UCMCustomQuery1 20
Other Queries 21
State Types and State Transitions 21
State Transitions 22

2. Setting Up Work Areas 25
2.1 Adjusting Your umask 26
The CCASE_BLD_UMASK Environmental Variable 26
2.2 Starting the Join Project Wizard 26
To Start the Join Project Wizard 27
2.3 Choosing a Project 27
2.4 Verifying Stream Names 27
Development Stream 27
Integration Stream 28

Under the Hood: Different Ways to Refer to Objects in a Project 28
Setting Activities in Projects that Are Enabled for ClearQuest 29
2.5 Setting Up Your Views 29
Determining the View Types 29
Locations for Snapshot Views 31
Under the Hood: A Snapshot View Storage Directory 31
Locations for Snapshot View Storage Directories 31
Under the Hood: .ccase_svreg 32
View-Tag for Dynamic Views 32
View Storage for Dynamic Views 33
Valid Locations for Dynamic View Storage Directories 33
2.6 Choosing Components to Load into Snapshot Views 34
Refining the List of Source Files 34
Loading Elements 34
2.7 Accessing Your Development View 35
Accessing a Snapshot View 35
Accessing Someone Else's Snapshot View 35
Accessing a Dynamic View 35
To Set a Dynamic View 35
To Mount VOBs 36
2.8 Logging In to a ClearQuest User Database 36

3. Finding and Setting Activities 37
3.1 Finding Activities 38
Finding Activities with MyToDoList and Other ClearQuest Queries 38
Using MyToDoList 38
To Open MyToDoList 39
Creating Your Own ClearQuest Query 39
Finding Activities with Project Explorer 40
Finding Activities with cleartool lsactivity 40
To List the Activity Objects in Your Development Stream 41
Finding Activities with ClearCase Dialog Boxes 41

3.2 Creating Activities 43
Creating Activities in a Project That Is Enabled for ClearQuest 43
To Create Activities from ClearQuest 43
Creating Activities in a Project That Is Not ClearQuest-Enabled 44
To Create Activities in a Project That Is Not ClearQuest-Enabled 44
3.3 Assigning Activities in a Project That Is Enabled for ClearQuest 45
To Make Activities Appear in MyToDoList 46
3.4 Setting Activities 46
Setting Activities from ClearQuest 47
To Set Activities from ClearQuest 47
Setting Activities with cleartool setactivity 48
To Set an Activity with cleartool setactivity 48
Setting Activities from ClearCase Dialog Boxes 49
Unsetting Your View from an Activity 49
To See Which Activity Is Currently Set 50

4. Working on Activities 51
4.1 Checking Out Elements 52
To Check Out Elements 52
Under the Hood: VOB Links 53
Symbolic Links and Hard Links in Dynamic Views 53
Symbolic Links in Snapshot Views 53
Hard Links in Snapshot Views 53
Caution: Losing Data Because of VOB Hard Links 54
4.2 Working with Checkouts 54
Viewing an Element's History 55
To View an Element's History 55
Comparing Versions of Elements 55
To Compare with a Predecessor 55
To Compare with a Version Other Than the Predecessor 55
Comparing with a Change-Set Predecessor 56
To Compare with a Change-Set Predecessor 57
Tracking Checked-Out Versions 57

Viewing an Activity's Change Set 57
To View an Activity's Change Set from ClearQuest 57
To View an Activity's Change Set from Project Explorer 58
To View an Activity's Change Set with cleartool lsactivity 58
Moving Versions to a Different Activity 59
To Move Versions with ClearCase 59
To Move Versions with ClearQuest 59
4.3 Canceling Checkouts 60
Under the Hood: Canceling Checkouts 60
Canceling Directory Checkouts 61
4.4 Checking In Elements 62
To Check In Elements 62
Snapshot View: Checking In VOB Links 63
4.5 Testing Your Work 63
4.6 Indicating Your Progress 64
Modifying Information in ClearQuest 64
To Modify ClearQuest Information 64
Closing Activities 65
To Close a ClearQuest Activity 65
Using Schema-Specific Actions 66
Reassigning Activities 67
To Change an Activity's Owner 67
Deleting Activities 68
To Delete a ClearCase Activity 68
To Delete a ClearQuest Activity 69

5. Delivering Activities 71
5.1 Preparing Your Work Areas 72
Finding, Comparing, and Checking In Work from Your Development View 72
To Find, Compare, and Check In Your Work 72
Updating Your Integration View 73
To Update a Snapshot View and Resolve Hijacked Files 73

5.2 Starting the deliver Operation 74
From the Command Line 74
From the Project Explorer 75
5.3 Selecting Activities 75
Activity Dependencies in the deliver Operation 76
5.4 Merging Versions 77
Under the Hood: Concurrent Deliver Operations 78
Under the Hood: Integration Activities and Baselines 79
Using Diff Merge to Resolve Differences 80
To Resolve Differences 80
Under the Hood: How ClearCase Merges Files and Directories 81
Handling a Binary File in a deliver Operation 82
Stopping the deliver Operation 82
5.5 Testing Your Work 83
Checking Out Versions 83
Under the Hood: Checking Out Versions from Snapshot Views 84
Undoing a deliver Operation 84
To Undo a deliver Operation 84
5.6 Completing the deliver Operation 85
To Complete a deliver Operation 85
What Happens When You Complete a deliver Operation 85
5.7 MultiSite: Posting Work to Deliver 85
Posting Work To Be Delivered 86

6. Rebasing Your Work Area 87
6.1 Preparing Your Development View 88
To Prepare Your Development View 88
6.2 Starting the rebase Operation 89
Rebasing Your Development Work Area 90
Merging Versions 91

6.3 Testing Your Development Work Area 91
Checking Out Elements 92
Undoing a rebase Operation 92
To Undo a rebase Operation 92
6.4 Completing the rebase Operation 93

7. Other Development Tasks 95
7.1 Adding Files and Directories to Source Control 95
Under the Hood: What Happens When You Add a File or Directory to Source Control 96
Importing Files 97
7.2 Moving, Removing, and Renaming Elements 98
Moving and Removing Elements 98
To Move an Element Within a VOB 99
To Move an Element to Another VOB 99
To Remove an Element Name from a Directory 99
Other Methods for Removing Elements 99
Renaming Elements 100
To Rename an Element 100
7.3 Accessing Elements Not Loaded into a Snapshot View 101
Listing All Elements in the VOB Namespace 101
7.4 Adjusting the Scope of a View 102
To Change Which Elements Are Loaded into a Snapshot View 102
To Add or Modify Load Rules When Editing the Config Spec 103
To Add Load Rules with update –add_loadrules 103
Mounting or Unmounting VOBs for Dynamic Views 104
To Mount VOBs 104
To Unmount VOBs 105
7.5 Moving Work from a Development Stream to Another Project 105
To Move Work from a Development Stream to Another Project 105

7.6 Moving Views 106
Changing the Physical Location of a Snapshot View 107
To Find the Location of the View Storage Directory 107
Update After Moving 107
Moving a View Storage Directory 107
7.7 Accessing Views and VOBs Across Platform Types 108
Creating Views Across Platform Types 108
Snapshot View Characteristics and Operating-System Type 108
Accessing Views Across Platform Types 109
Accessing UNIX Snapshot Views from Windows Hosts 109
Accessing Windows Snapshot Views from UNIX Hosts 109
Accessing UNIX Dynamic Views from Windows Hosts 110
Accessing Windows Dynamic Views from UNIX Hosts 110
Accessing VOBs Across Platform Types 110
Developing Software Across Platform Types 111
7.8 Regenerating the .view.dat File 112
To Regenerate the .view.dat File 112
7.9 Regenerating the .ccase_svreg file 112

A. Working in a Snapshot View While Disconnected from the Network 113
A.1 Setting Up a View for Your Hardware Configuration 114
Under the Hood: Location of the View Storage Directory in Disconnected-Use Configurations 115
A.2 Preparing the View 116
A.3 Disconnecting the View 116
A.4 Working in the View 116
Hijacking a File 117
To Hijack a File 117
Finding Modified Files While Disconnected 117
A.5 Connecting to the Network 118
A.6 Using the Update Tool 118

Determining How to Handle Hijacked Files....118
To Find Hijacked Files....119
To Compare a Hijacked File to the Version in the VOB....119
Checking Out a Hijacked File....120
You May Be Prompted to Merge....120
To Merge with the Latest Version....121
Undoing a Hijack....121
Under the Hood: How ClearCase Determines Whether a File is Hijacked....122
Other Ways to Handle Hijacked Files....122
Updating the View....122

Index....123

Figures

Figure 1 Elements, Components, and Baselines 9
Figure 2 Activity Object 10
Figure 3 A Stream's Configuration 12
Figure 4 Many Development Streams 13
Figure 5 Integration Stream 14
Figure 6 Delivering to the Integration Stream 15
Figure 7 Creating a Baseline 16
Figure 8 Rebasing Work Areas 16
Figure 9 Activities and UCM-Enabled Records 18
Figure 10 UCM-Enabled Queries in ClearQuest 19
Figure 11 Example of ClearQuest State Transitions 23
Figure 12 List of Projects 27
Figure 13 Choose View Type 30
Figure 14 Linked Activities in ClearQuest MyToDoList 39
Figure 15 Activities in Project Explorer 40
Figure 16 Activities in the Check Out Dialog Box (Project That Is Enabled for ClearQuest)42
Figure 17 Select Your View 48
Figure 18 Compare with Change Set Predecessor 56
Figure 19 Select Your Stream 74
Figure 20 Undelivered Activities 75
Figure 21 Activity Dependencies 77
Figure 22 Deliver Progress 78
Figure 23 Diff Merge Window 80
Figure 24 ClearCase Merge Algorithm 82
Figure 25 Rebase Stream Dialog Box 89
Figure 26 Development Work Area After Rebasing 90
Figure 27 Creating an Element 97
Figure 28 View on a Laptop 114
Figure 29 View On a Removable Storage Device 114

Figure 30 Copy the View 115
Figure 31 Hijacked Files in the Update Window 119
Figure 32 Hijacked Version May Not Be the Latest Version 120

1 The UCM Workflow

Unified Change Management (UCM) structures the efforts of your software development team into a defined, repeatable process. This chapter describes the UCM workflow and how it affects your work as a developer.

Recommended Reading Paths

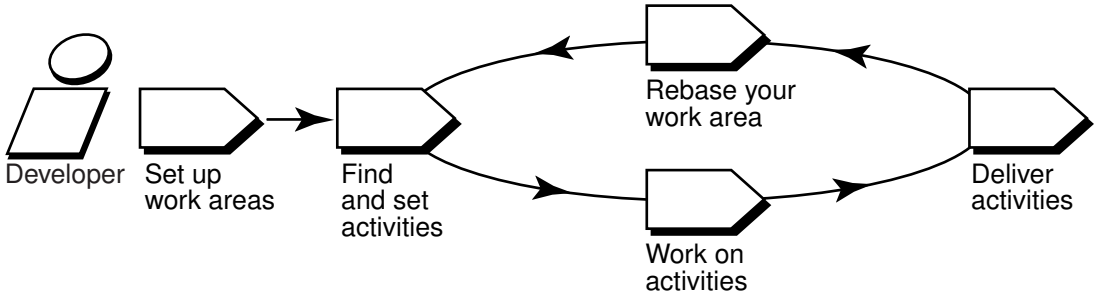
Read this chapter first. Then, if you want to start working immediately, use online help to learn as you go. Or, if you prefer a more structured approach, use the remainder of *Working in UCM* as a guide through the development cycle.To start ClearCase online help, type this command:

hyperhelp cc_main.hlp

Until you are familiar with using Rational ClearCase, consider skipping the sections titled *Under the Hood*, which describe advanced concepts and technical details.

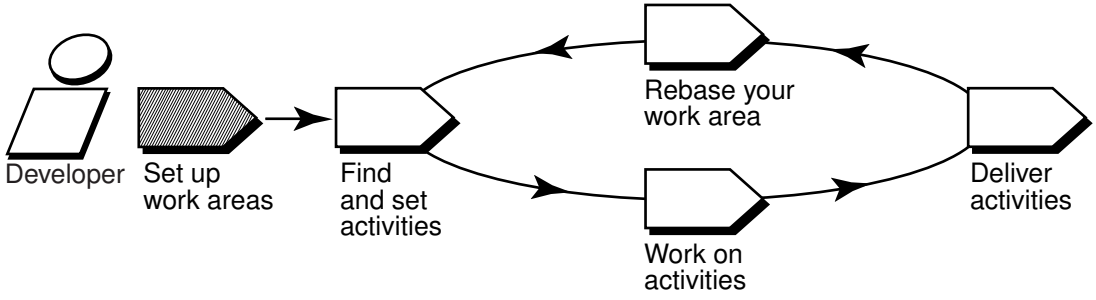
1.1 The UCM Workflow

When your project uses UCM, your work as a software or information developer follows a cycle.



To access your project's source files, you set up work areas. As you modify source files, you use activities to organize and identify your work. Other developers on the project do not see your work until you deliver your activities to a shared work area. Periodically, the project manager incorporates activities in the shared work area into baselines, which are sets of activities that represent a significant change. Then, you synchronize (or rebase) your work area with the activities in the new baseline.

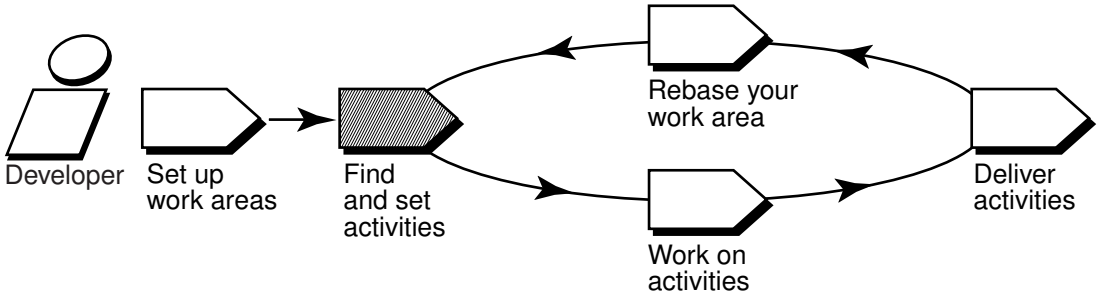
Setting Up Work Areas



To contribute work to a UCM project, you must set up two work areas: a development area for working on activities in isolation and an integration area for testing the activities you deliver.

Each work area consists of a view, which provides a directory tree of source files for you to modify, and a stream, a ClearCase administrative object that configures the view and keeps track of the activities you create and deliver. For more information, see *Views and Streams* on page 10.

Finding and Setting Activities



After you set up your work areas, find any activities that your project manager or other team members have assigned to you; if necessary, you can add new ones.

Finding Activities

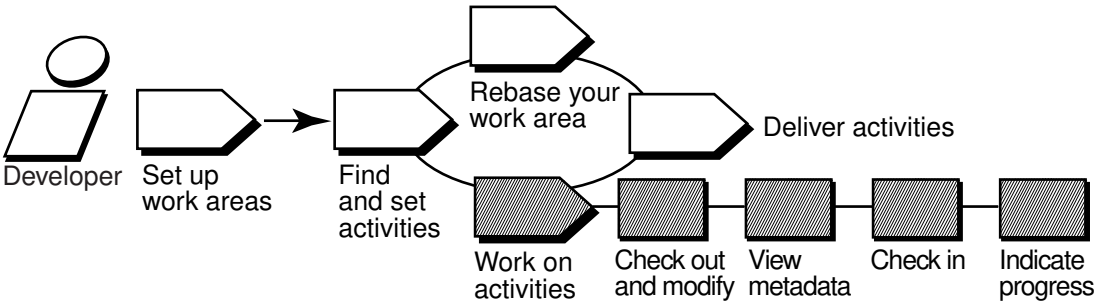
The **cleartool lsactivity** command shows the activities that are in your work area. If your project uses Rational ClearQuest as its change-request management system, you can use ClearQuest queries such as MyToDoList to find activities that are assigned to you but are not yet in your work area. For more information, see *UCM-Enabled Record Types and ClearCase Activities* on page 17 and *Queries* on page 18.

When you're ready to work on an activity, you must set it in your development view.

Setting Activities

Setting your view to an activity instructs ClearCase to assign any versions of source files you create to the activity's change set. You must set your view to an activity before you can check out and modify source files.

Working on Activities



To work on activities, you check out and modify source files and test your modifications. You may also need to enter information in your change-request management system to communicate your progress.

Checking Out and Modifying Source Files

Checking out a file makes it writable in your view. Then you can use any editing tool to modify it.

Viewing Metadata

Metadata is information that ClearCase and ClearQuest keep about your activities and source files. While working with checkouts, you may want use ClearCase and ClearQuest to view the following types of metadata:

- The revision history of an element
- A comparison between versions of an element
- A list of versions that are checked out to your work area

Checking In and Testing Source Files

When you want to keep a record of a file's current state, check it in. Any work you check in from your development view is not available to other team members until you deliver it.

Build and test the work in your development view, especially after you rebase the stream and before you deliver activities, to reduce the amount of merging needed when you deliver your work.

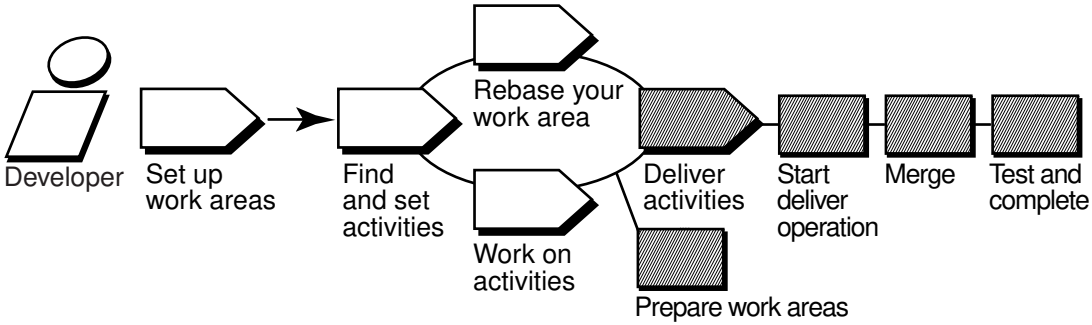
Indicating Your Progress

If your project is enabled for ClearQuest, you can indicate your progress on an activity in several ways:

- Modifying information in ClearQuest
- Closing an activity
- Using schema-specific actions

You may need to do reassign or delete an activity, regardless of whether your project is enabled for ClearQuest. For more information, see *Indicating Your Progress* on page 64. When you're ready to make your work available to the rest of the project, deliver it.

Delivering Activities



When you're ready to make one or more activities available to the project team, prepare your work areas. Then start the deliver operation.

Preparing Your Work Areas

Do the following to prepare your work areas:

- If your project manager has created a new recommended baseline since you last rebased, rebase your development area.
- Find, compare, and check in the work you want to deliver.

Starting the Deliver Operation

After preparing your work areas, start the deliver operation.

Merging

As part of the deliver operation, ClearCase merges the work in your development stream with the work in the integration stream. It completes trivial merges for you. If it encounters merge conflicts, it prompts you to resolve the conflicts.

Testing

Your integration view contains the merge results. To make sure that your delivered work is compatible with the work in the integration stream, build and test the files in the integration view.

In addition to building and testing, you may need to do the following:

- Edit the checked-out versions to resolve build errors.
- Check out and edit additional files.

  NOTE: We recommend that you do not check in any of your changes until you are ready to complete the operation. Checking in complicates efforts to undo the deliver operation.
- Update your integration view to see the work other developers have delivered since you started the deliver operation.

Completing the Operation

When you are satisfied with your test builds, complete the delivery. ClearCase checks in your modifications and changes the state of your stream.

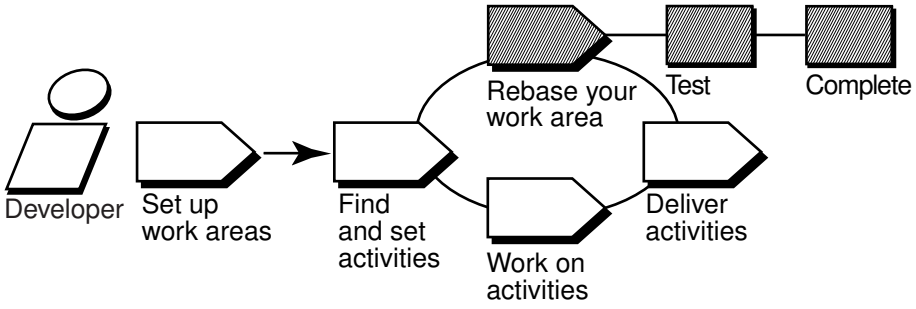
Delivering with MultiSite

If your project uses MultiSite to share source data with developers in other geographical locations, you may use a different method for delivering activities. If a different site is responsible for controlling your project's source data, you do the following:

- Post a list of activities that are ready to be delivered.
- Notify the project manager at the site that controls your project's source data.

Then the project manager at the other site merges your activities to the integration stream and tests your work.

Rebasing Your Development Work Area



The project manager organizes delivered activities into baselines. Usually, baselines go through a cycle of testing and fixing bugs until they reach a satisfactory level of stability. When a baseline reaches this level, your project manager designates it as a recommended baseline.

To work with the set of versions in the recommended baseline, you rebase your development work area. To minimize the amount of merging necessary when you deliver activities, rebase your development work area with each new recommended baseline as it becomes available.

After you rebase, be sure to build and test the source files in your development view to verify that your undelivered activities build successfully with the versions in the baseline.

1.2 Under the Hood: ClearCase UCM Concepts

This section describes fundamental ClearCase UCM concepts:

- Projects
- Components and baselines
- Activities
- Views and streams
- Delivering, creating baselines, and rebasing

Reading this section isn't necessary to start working on a UCM project, but it does provide a basis for understanding how ClearCase implements UCM and may be helpful if you need to diagnose problems.

Projects

In ClearCase, *project* refers to an object that defines a set of development policies and a set of configurations used in a significant development effort. Your organization may create a project for each product it develops, for a group of products, for a subset of a product's functionality, or for a product release.

A project's policies govern how developers access and modify sets of source files and directories (called components). To record and configure the development work that proceeds on components, projects use the following ClearCase objects:

- Baselines
- An integration stream
- Development streams
- Activities

Because ClearCase supports *parallel development*, different projects can work concurrently with different versions of the same set of source files.

ClearCase stores projects in a data repository called a *PVOB* (project versioned object base).

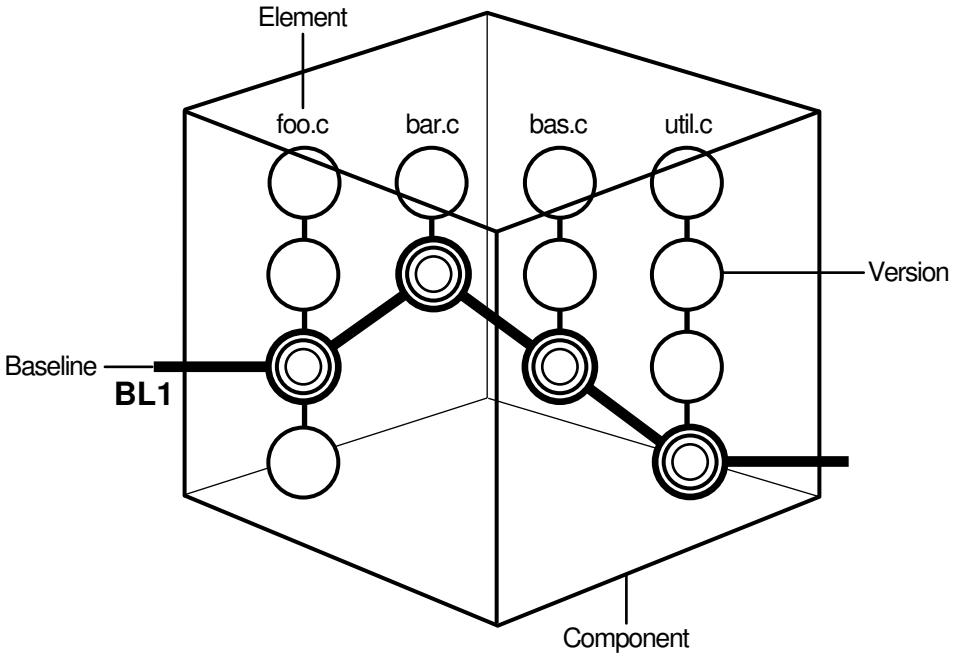
Components and Baselines

Your project manager organizes files and directories that are under source control into components. Usually, components can be built into functional units of software.

Files and directories in a component are called elements, and each checked-in revision of an element is called a version.

To keep track of different configurations of versions, your project manager creates a baseline, which records one version of each element in a component. In effect, a baseline is a version of a component (Figure 1).

Figure 1 Elements, Components, and Baselines



ClearCase stores elements (and their versions) in data repositories called *versioned object bases* (VOBs); it stores component and baseline definitions in PVOBs.

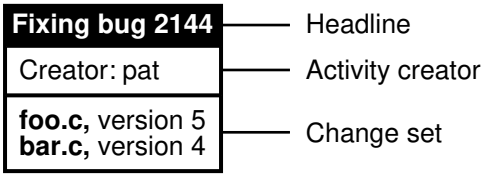
Activities

To create a version in a ClearCase UCM project, you must first assign it to an activity. An activity is a ClearCase object that identifies the versions created to complete a development task. For example, the versions you create to fix a defect that is stored in your change request management system may constitute an activity. Your organization determines the scope of its activities.

The ClearCase activity object includes a text headline, which describes the task, the user ID of the activity creator, and a change set, which identifies all versions that you create or modify while working on the task (Figure 2). (For more information on an activity's headline, see *Under the Hood: Different Ways to Refer to Objects in a Project* on page 28.)

NOTE: In a project that is enabled for ClearQuest, ClearQuest includes a field to describe the activity owner. This activity owner and the ClearCase activity creator are two different data points; the former is stored in ClearQuest and the latter in ClearCase.

Figure 2 Activity Object



An activity object belongs to a single stream and cannot be moved from one stream to another. (If you assign one or more versions to the wrong activity or if you create new activities to better represent your work, you can assign the versions to a different activity. For more information, see *Moving Versions to a Different Activity* on page 59.)

When you deliver an activity, ClearCase merges the versions in the activity's change set to the integration stream, but it does not move the activity object to the integration stream. Instead, it creates an integration activity to identify the versions created as a result of the merge. For more information, see *Under the Hood: Integration Activities and Baselines* on page 79.

In a project that is enabled for ClearQuest, ClearCase links its activity objects to UCM-enabled records in ClearQuest. For more information, see *UCM-Enabled Record Types and ClearCase Activities* on page 17.

Views and Streams

To create new versions of elements in a component, you use a view; to keep track of the versions you create, you use a stream.

Views

A view provides a directory tree of one version of each file in one or more components. In the view, you modify source files, compile them into object modules for testing purposes, format them into documents, and so on. ClearCase offers two kinds of views:

- *Snapshot views*, which copy files from VOBs to your computer.
- *Dynamic views*, which provide immediate, transparent access to the data in VOBs. (Dynamic views are not available on all platforms. For more information, see the ClearCase online help.)

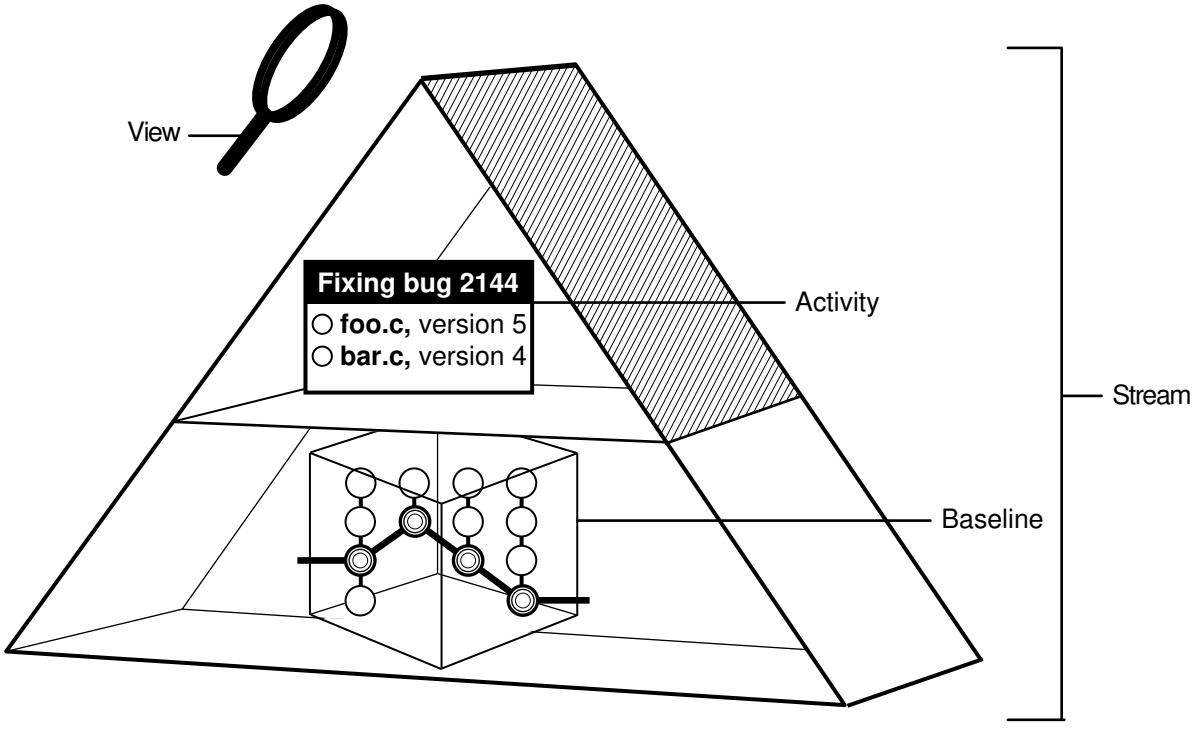
An associated stream determines which versions of elements are in the view.

Streams

A stream is a long-lived ClearCase object. It is a member of a single UCM project and is a mechanism for creating and recording configurations. A stream identifies the exact set of versions currently available for you to view, modify, or build.

UCM uses baselines and activities to encapsulate a stream's configuration. When you create a stream, its original configuration is the same as a baseline (that is, it identifies a single version of each element in a component). When you create new versions of elements, you assign the new versions to one or more activities. Hence, a stream's configuration is a given baseline plus one or more activities (Figure 3).

Figure 3 A Stream's Configuration



The following actions modify a stream's configuration:

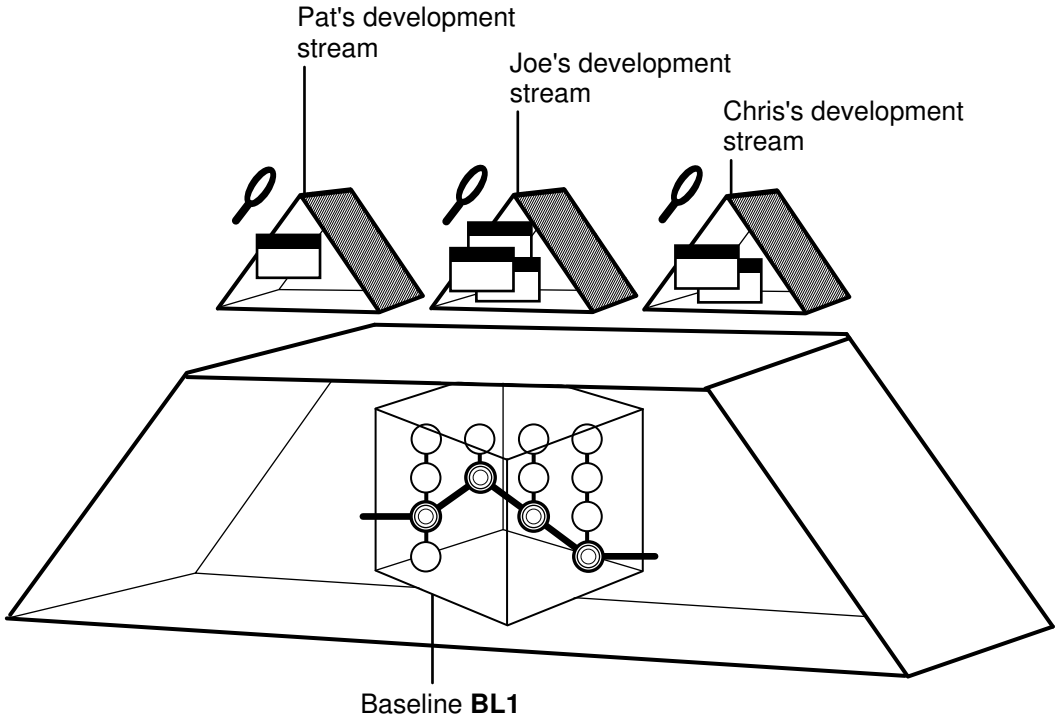
- Checking in versions from an associated view. (A stream may be associated with several views.)
- Rebasing, which replaces the baseline in the stream's configuration with a more recent one.
- Delivering activities, which changes the configuration of the integration stream by adding activities that were previously available only to the contributing development stream. (Delivering activities does not modify the development stream.)

A project includes two kinds of streams: development streams and integration streams.

Development Stream

Typically, each project includes many development streams (one for each developer on the project), all of which start from a baseline and evolve separately as developers add activities (Figure 4.)

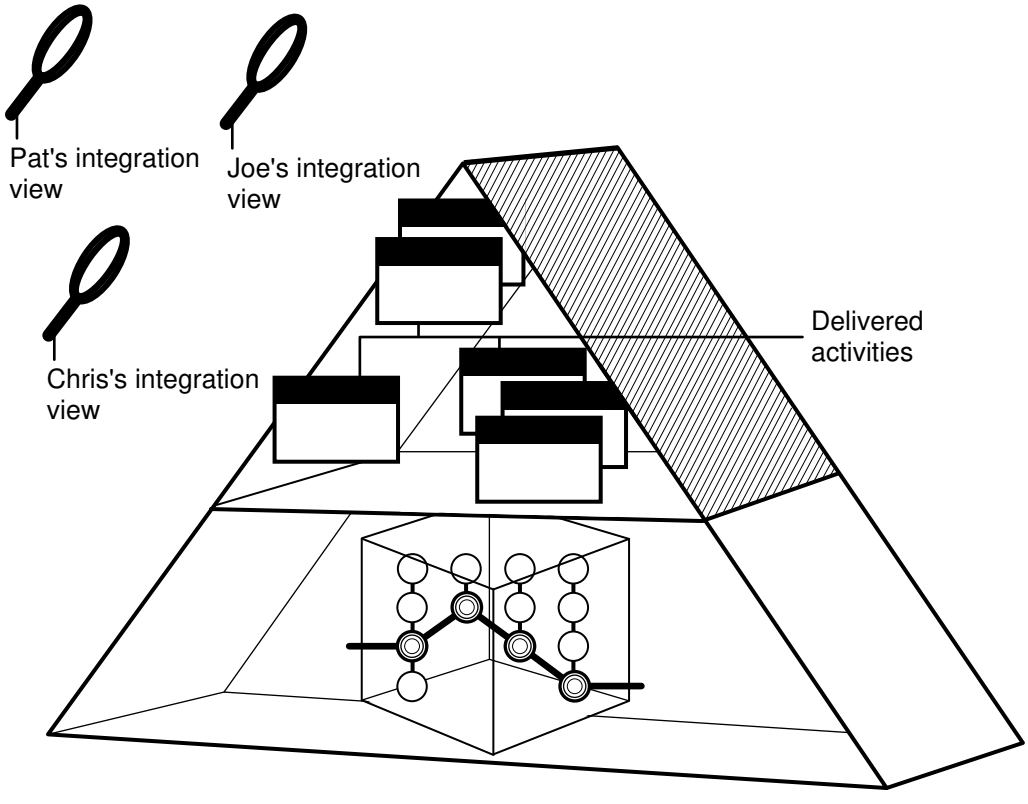
Figure 4 Many Development Streams



The Integration Stream

Projects always include a single integration stream to collect the work team members deliver from their development streams (Figure 5).

Figure 5 Integration Stream

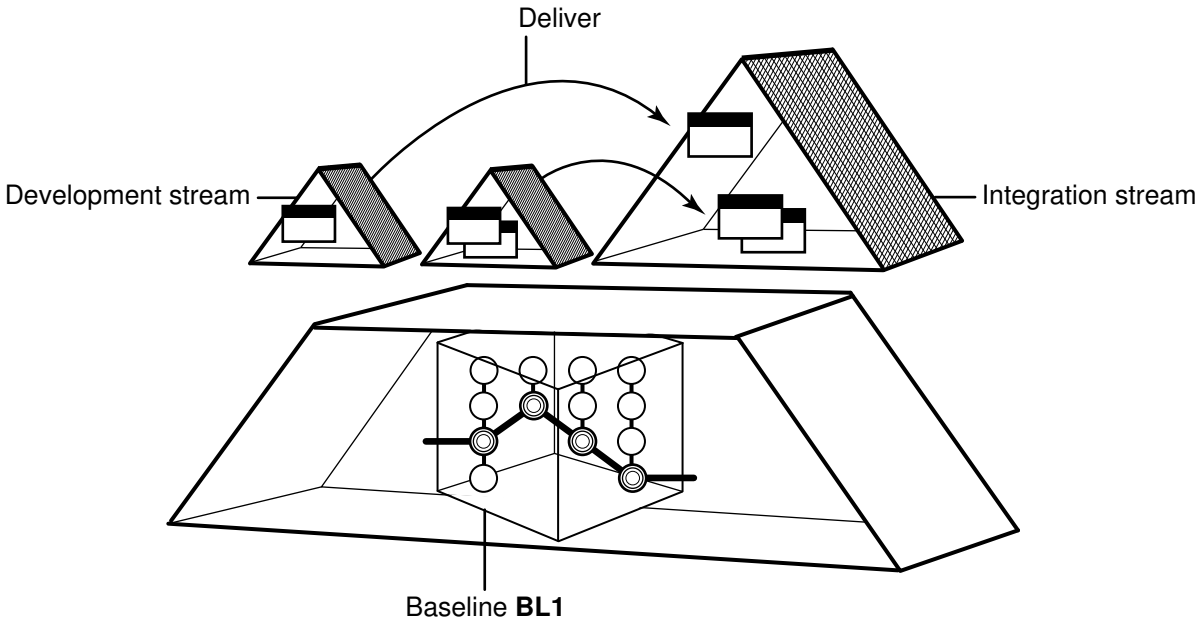


To facilitate delivering activities, each team member associates a separate integration view with the project's integration stream. The integration views show the baseline and all delivered activities.

Delivering, Creating Baselines, and Rebasing

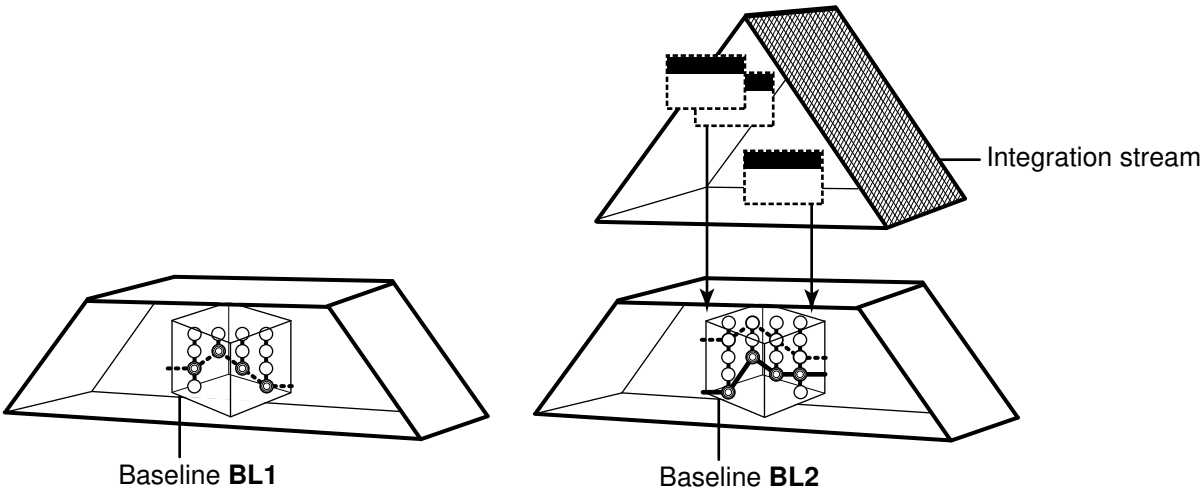
Developers on a project deliver activities from their development streams to a single integration stream, which identifies a shared set of versions to be used for projectwide building and testing (Figure 6).

Figure 6 Delivering to the Integration Stream



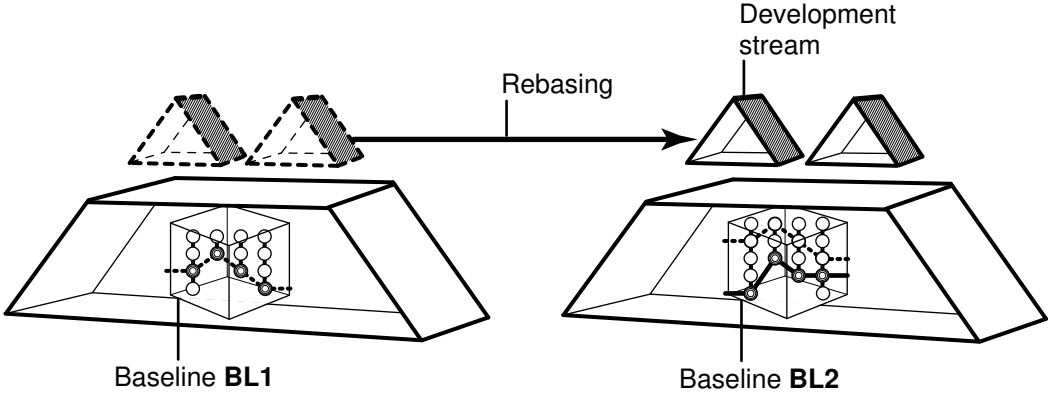
Project managers collect the activities delivered to the integration stream into baselines (Figure 7.) They characterize specific baselines that include stable and significant changes as *recommended*.

Figure 7 Creating a Baseline



Developers rebase their development work areas to see the new work identified by the recommended baselines. After rebasing, a development work area shows the versions specified by the baseline, plus any of your undelivered activities (Figure 8.)

Figure 8 Rebasing Work Areas



1.3 Under the Hood: UCM ClearQuest Concepts

Rational ClearQuest is a change-request management system that can integrate with UCM projects to provide extended functionality. This section describes fundamental concepts for the UCM-ClearQuest integration:

- ClearQuest schema
- UCM-enabled record types and ClearCase activities
- Queries
- State types and state transitions

Reading this section isn't necessary to start working on a UCM project that is enabled for ClearQuest, but it does provide a basis for understanding how ClearQuest implements the integration and may be helpful if you need to diagnose problems.

The UCM-ClearQuest Schema

ClearQuest stores change-requests as records in a database. A *schema* defines the types of records in the database, the states available to each type of record, and other attributes of the database.

ClearQuest supplies two UCM-enabled schemas, Unified Change Management and Enterprise; your project manager can set up a custom UCM-enabled schema. To support the UCM-ClearQuest integration, a ClearQuest database must use one of these UCM-enabled schemas.

ClearQuest stores all schemas in a schema repository, which is a master database of schemas. The schema repository does not contain any user-owned data or change request data. Change request data is stored in a user database.

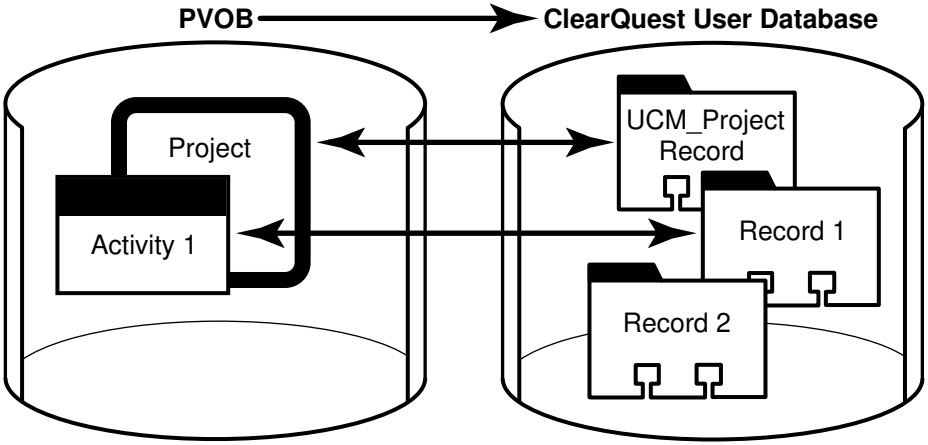
UCM-Enabled Record Types and ClearCase Activities

A UCM-enabled record type is a template for ClearQuest records. It includes definitions for a set of fields that store information about ClearCase activities. Your ClearQuest database may include different UCM-enabled record types for different purposes (such as tracking defects and enhancement requests), and it may include other record types that are not UCM enabled.

In a project that uses the UCM-ClearQuest integration, records based on a UCM-enabled record type can be linked with ClearCase activity objects (Figure 9). This link enables ClearQuest to

display information about the ClearCase activity (such as its change-set, its stream, and whether it is currently set in any view). It also enables policies governing when you can deliver an activity in ClearCase and when you can close an activity in ClearQuest. Because of the close association between linked UCM-enabled records and ClearCase activities, the UCM documentation usually refers to both linked entities as *activities*.

Figure 9 Activities and UCM-Enabled Records



At any point in a project, your ClearQuest database may contain UCM-enabled records that are not linked to a ClearCase activity object. For example, a newly created record may not be linked to a ClearCase activity. You must explicitly complete an action (for example, by clicking **Action>WorkOn** in ClearQuest) to link a UCM-enabled record to a ClearCase activity.

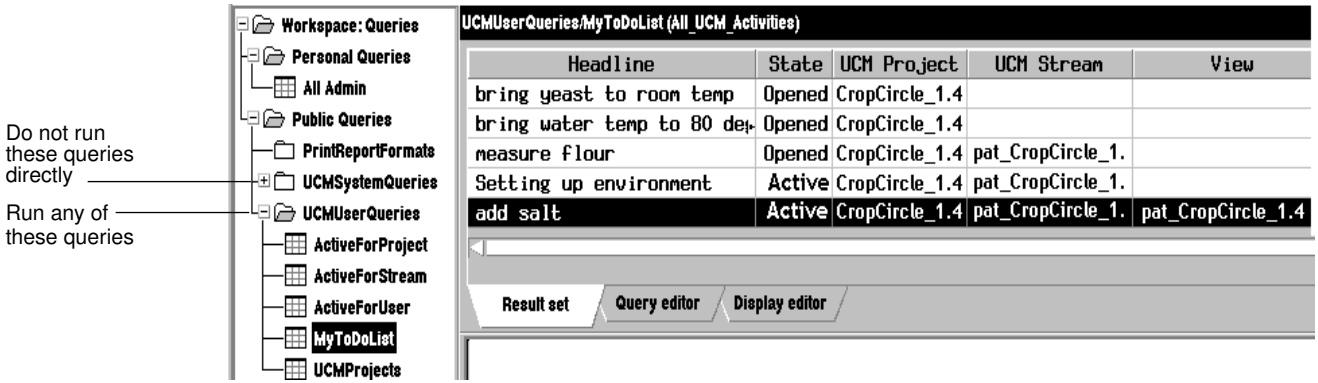
Each ClearCase activity in a UCM project that is enabled for ClearQuest must be linked to a ClearQuest record. You cannot create a ClearCase activity object without linking it to a UCM-enabled record in a ClearQuest database.

Queries

Queries are the vehicle for navigating through the ClearQuest database. Before viewing or modifying records, you must query the database to find the records you are interested in. For example, you may want to see only activities that are assigned to you or only those activities that are associated with your project, or you may want to see activities created before or after a particular date.

A UCM-enabled schema includes a set of queries that you can use to find UCM-enabled records. ClearQuest places the queries into two categories in the ClearQuest Public Queries folder (Figure 10).

Figure 10 UCM-Enabled Queries in ClearQuest



You can run queries in the UCMUserQueries folder and modify them when necessary. You can save any public query you modify in your Private Queries folder.

Do not run queries in the UCMSystemQueries folder from ClearQuest; they are intended to be run by the ClearQuest and ClearCase system only.

This section describes the following queries:

- MyToDoList
- UCMCustomQuery1
- Other queries

In addition to the queries included in the database schema, you and your project manager can create other queries.

MyToDoList

The MyToDoList query finds UCM-enabled records that all of match the following criteria:

- They are assigned to you.
- They are in a state whose state type is Ready or Active.

You or your project manager can modify this query.

UCMCustomQuery1

The UCMCustomQuery1 query finds UCM-enabled records that match any of the following criteria:

- In a state whose state type is **Ready** and match all of the following criteria:
  - Are assigned to you
  - Have not yet been assigned to a UCM project
  - Have not yet been worked on

- In a state whose state type is Ready or Active and match all of the following criteria:
  - Are assigned to you
  - Have been assigned to the UCM project associated with the current view
  - Have not yet been worked on

- In a state whose state type is Active and match all of the following criteria:
  - Are assigned to you
  - Have been assigned to the UCM project associated with the current view
  - Have been worked on already in the stream associated with the current view

The ClearCase **checkin** and **checkout** dialog boxes present the activities that UCMCustomQuery1 finds. Although you can also see this query from ClearQuest, it is not designed to be run outside the context of a ClearCase view.

Your project manager can modify this query.

Other Queries

A UCM-enabled schema also supplies the queries described in Table 1. You can run any of these queries.

Table 1 Other Queries in the Unified Change Management Schema

Query	Description
ActiveForProject	For one or more specified projects, selects all activities in an active state type.
ActiveForStream	For one or more specified streams, selects all activities in an active state type.
ActiveForUser	For one or more specified developers, selects all assigned activities in an active state type.
UCMProjects	Selects all UCM-project records in ClearQuest user database.

State Types and State Transitions

Change requests move through a pattern, or life cycle, from submission through resolution. In ClearQuest, each stage in this life cycle is called a *state*, and each movement from one state to another is called a *state transition*.

As with record types and records, a state type is a template that defines actions and other attributes associated with a state. The states in a UCM-enabled schema must be based on one of the following state types:

- Waiting
- Ready
- Active
- Complete

Your project manager may give the states in your UCM-enabled schema different names and may create multiple, different states based on the same state type. For example, your UCM-enabled schema may contain the states Scheduled and Deferred, both of which are based on the Active state type but have different associated actions and meanings.

State Transitions

Schemas include rules for changing records from one state type to the next. In a UCM-enabled schema, records must follow these state-transition rules (Figure 11):

- **Waiting**. Creating an activity places it in a state based on this state type to indicate, for example, that the work is waiting for someone to resolve dependencies and that it is not ready to be scheduled. Activities in this state do not appear in MyToDoList; they may or may not contain a value in the owner field and may or may not contain a value in the UCM project field. The schema in Figure 11 bases its Submitted and Postpone states on the Waiting state type.

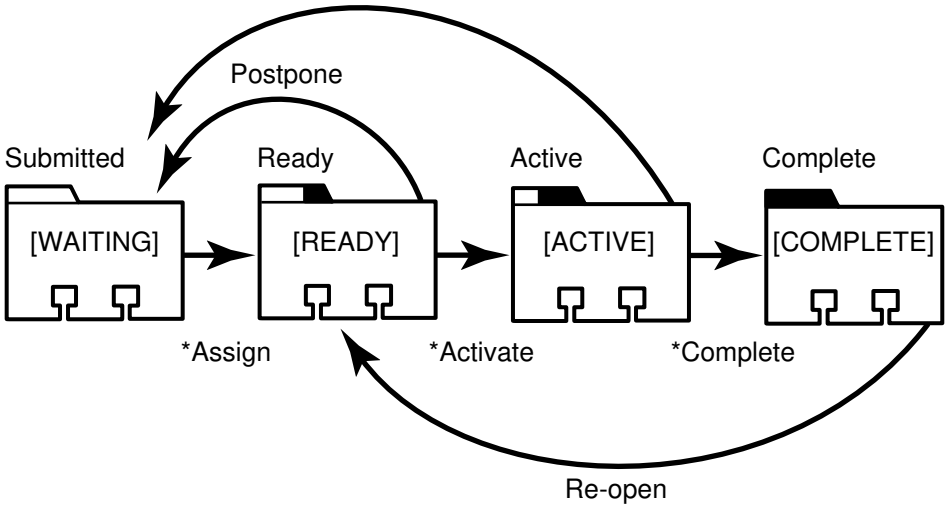
- **Ready**. Assigning an activity usually places it in a state based on this state type to indicate that the work is pending. Activities in this appear in the activity owner's MyToDoList, but are not part of the owner's stream. The schema in Figure 11 bases its Ready and Re-open states on the Ready state type.

- **Active**. Setting your view to an activity usually places the activity in a state based on this state type to indicate that the work is in progress. Setting your view to an activity also links the ClearQuest UCM-enabled record to the ClearCase activity. After the activity is linked, you can change the ClearQuest record's owner but you cannot change the ClearCase stream that contains the activity. For more information, see *Reassigning Activities* on page 67.

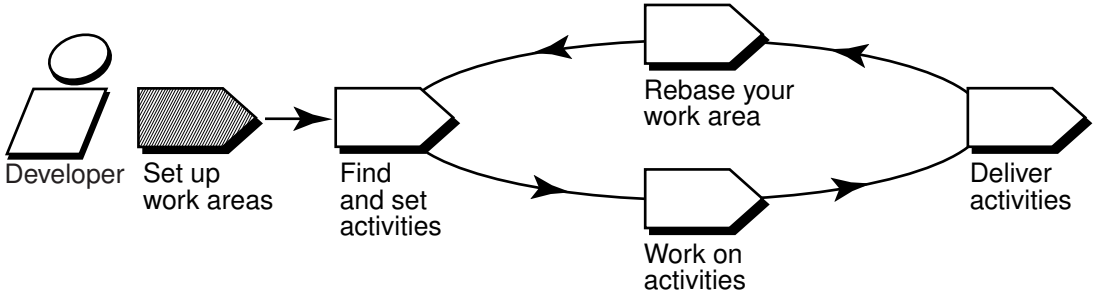
  The **ucm_stream** field in a UCM-enabled record displays the stream to which an activity is linked, and the **ucm_view** field indicates whether an activity is currently set in a view. The schema in Figure 11 bases its Active state on the Active state type.

- **Complete**. When you have completed and delivered an activity, you change it to a state based on the Complete state type. Activities in this state do not appear in MyToDoList, but do remain a permanent part of your stream. The schema in Figure 11 bases its Complete state on the Complete state type

Figure 11 Example of ClearQuest State Transitions



2 Setting Up Work Areas



Before setting up work areas, consider adjusting your **umask(1)** setting to determine the level of access others have to your work. (See *Adjusting Your umask* on page 26 for more information.)

Then, complete the following tasks to set up your work areas:

- Starting the Join Project Wizard
- Choosing a project
- Verifying stream names
- Setting up views
- Choosing components to load into snapshot views
- Accessing your development view
- Logging in to a ClearQuest user database (only for projects that use the UCM-ClearQuest integration)

NOTE: If you prefer to use the command line, you can complete the tasks in this chapter by using arguments for the **cleartool mkstream** and **cleartool mkview** commands. For more information, see the corresponding reference pages in *ClearCase Reference Manual* and *Under the Hood: Different Ways to Refer to Objects in a Project* on page 28.

2.1 Adjusting Your umask

Your **umask(1)** setting at the time you join a project affects how accessible your views are to others. For example:

- A umask of **002** is appropriate for a view that you share with other users in the same group. Members of your group can create and modify view-private data; those outside your group can read view-private data, but cannot modify it. To completely exclude nongroup members, set your umask to **007**.
- A umask of **022** produces a view in which only you can write data, but anyone can read data.
- A umask of **077** is appropriate for a completely private view. No other user can read or write view-private data.

Change your umask in the standard way. For example, enter this command from a shell:

umask 022

For more information, refer to a **umask(1)** man page.

The CCASE_BLD_UMASK Environmental Variable

You can also use the **CCASE_BLD_UMASK** environmental variable (EV) to set the **umask(1)** value for files created from a **clearmake** build script. It may be advisable to have this EV be more permissive than your standard umask—for example, **CCASE_BLD_UMASK** = 2 where umask is 22.

For more information on ClearCase EVs, see the **env_ccase** reference page in *ClearCase Reference Manual*.

2.2 Starting the Join Project Wizard

The Join Project Wizard assists you in each step of joining a project. After adjusting your umask setting, start the Join Project Wizard and use this chapter to complete the steps.

To Start the Join Project Wizard

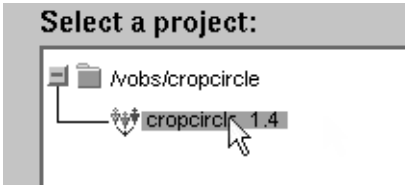
Type the following command:

clearjoinproj

2.3 Choosing a Project

As its first step, the Join Project Wizard prompts you to choose a project. Navigate the list of projects and click the one you want to join (Figure 12).

Figure 12 List of Projects



To see a description of a project, select it and click **Properties**.

2.4 Verifying Stream Names

After you choose a project, the Join Project Wizard presents the two streams that ClearCase uses to keep track of your work on this project.

Development Stream

The Join Project Wizard gives your development stream a title based on the following convention: *userID_project-name*. For example, Pat's development stream for the 1.4 release of the Cropcircle project is titled **pat_1.4_cropcircle**.

Although you can change the title of your development stream on this step of the Join Project Wizard, we recommend that you do not, unless your project manager has requested it. Following consistent conventions facilitates project management.

Integration Stream

Your project manager creates the integration stream as part of creating the project. The Join Project Wizard presents the title of the project's integration stream for informational purposes only.

Under the Hood: Different Ways to Refer to Objects in a Project

Project objects, such as streams and activities, have two types of identifiers:

- Titles or headlines provide a way to refer to the object that is easy to remember and meaningful to you. Titles (which apply to projects, streams, and folders) and headlines (which apply to *activities*) do not need to be unique among all objects in a PVOB.
- Object-selectors provide an identifier that is unique among all objects in all PVOBs in your organization. An object-selector uses the following syntax:

  [*object-type*:] *alphanumeric-string* [***@/****PVOB-tag*]

  The alphanumeric string must be unique to the current PVOB; ClearCase provides one if you do not. For example, the selector ClearCase provides for the **pat_1.4_cropcircle** stream is **stream: pat_1.4_cropcircle_02442@/projects**.

  @/*PVOB-tag* is not required for **mkactivity** and **setactivity** as long as you issue those commands from a view attached to the stream in which you want to make or set the activity.

When you use ClearCase and ClearQuest GUIs, you need to know only an object's title or headline. For example, when the Project Explorer lists the streams and activities for a given project, it displays the stream titles and the activity headlines. These identifiers may not be unique, but are usually sufficient for the context in which they are displayed.

When you use the command-line interface (CLI), which gives you greater flexibility, you must use the object-selector to refer to streams and activities. If you plan to work primarily from the command line, we recommend that you supply meaningful alphanumeric strings when you create and work on objects, rather than use the ClearCase default string, which is unique, but may not be as easy to remember.

Setting Activities in Projects that Are Enabled for ClearQuest

When using the command line to set activities in projects that are enabled for ClearQuest, you must use the record ID of the associated ClearQuest record (for example, SAMPL123456) instead of the ClearCase object-selector.

2.5 Setting Up Your Views

In the next steps of the Join Project Wizard, you set up your development view and integration view by specifying the following information:

- Which type of view to create for your development view and integration view
- Locations for snapshot views
- View tags for dynamic views
- Locations for dynamic view storage directories

We recommend that you accept the defaults the Join Project Wizard presents until you are familiar with UCM. For example, the default naming conventions for your views are intended to distinguish your development view from your integration view.

Determining the View Types

As described in *Views* on page 11, you can use either a snapshot view or a dynamic view to create a directory tree of source files. Your project manager determines the default view types that the Join Project Wizard presents.

Work in a snapshot view when any of these conditions is true:

- Your workstation does not support dynamic views.
- You want to work with source files under ClearCase control when you are disconnected from the network that hosts the VOBs.
- You want to simplify accessing a view from a workstation that is not a ClearCase host.
- Your development project doesn't use the ClearCase *build auditing* and *build avoidance* features.

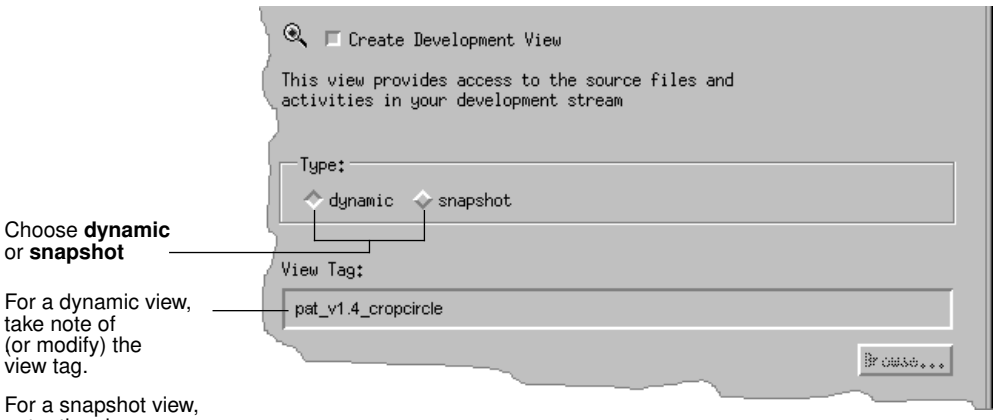
Work in a dynamic view when any of these conditions is true:

- Your development project uses build auditing and build avoidance.
- You want to access elements in VOBs without copying them to your workstation.

Because dynamic views can show changes in its associated stream at all times without requiring an update, we recommend that you use dynamic views for your integration views whenever they are available.

The two view types behave slightly differently (for example the way you access a snapshot view is different from the way you access a dynamic view), so remember the type of view you create (Figure 13).

Figure 13 Choose View Type



For a detailed comparison of snapshot views and dynamic views, see the **view** reference page in *ClearCase Reference Manual*.

Locations for Snapshot Views

If you create a snapshot view, you must provide a view location. ClearCase loads, or copies, versions of source files from VOBs into the snapshot view at the location you specify.

When choosing a location, consider these constraints:

- The view's root directory must be located on a disk with enough space for the files loaded into the view and any other files you add. Files and directories that you add to the view but do not add to source control are called view-private files and view-private directories.
- Your organization may restrict where you can create a view. For example, you may be required to use a disk that is part of a data-backup scheme.
- If you use the view on several hosts, make sure the location can be accessed by all those hosts; that is, choose a disk partition that is exported.

Under the Hood: A Snapshot View Storage Directory

Every snapshot view has a *view storage directory* in addition to the directory tree of source files that it loads from VOBs. ClearCase uses the snapshot view storage directory to keep track of such information as which files are loaded into your view and which versions are checked out to it. The view storage directory is for ClearCase administrative purposes only. Do not modify anything in it.

For every 1,000 elements loaded into the view, ClearCase uses about 400 KB of disk space for the view storage directory.

Locations for Snapshot View Storage Directories

Usually, your ClearCase administrator sets up a storage location, which is a directory on a ClearCase server host on UNIX or Windows, and by default ClearCase locates snapshot view storage directories in the storage location. If your ClearCase administrator sets up more than one storage location, ClearCase selects one of these locations as the default when you create a view.

You can override the default and locate the view storage directory under the root directory of the snapshot view. If you choose to locate the view storage directory under the root directory of the view, be aware of the following recommendations:

- Do not choose this configuration if you use the view when disconnected from the network. You can corrupt the data in the view storage directory if you disconnect it from the network while the view's **view_server** process is running.
- Make sure the view storage directory is accessible to any data backup schemes your organization institutes.

If your ClearCase administrator does not set up storage locations, ClearCase locates the view storage directory under the root directory of the snapshot view.

NOTE: If you plan to work while disconnected from the network, your administrator must set up storage locations.

Under the Hood: .ccase_svreg

When you create a snapshot view, ClearCase creates or modifies the file **.ccase_svreg** in your home directory. Do not remove or relocate this file; some ClearCase operations require it.

If you inadvertently delete or corrupt this file, see *Regenerating the .ccase_svreg file* on page 112.

View-Tag for Dynamic Views

If you choose to create a dynamic view, take note of the view-tag. You must use the view-tag to start and access the view. The wizard presents the default view-tag based on the following conventions:

- For a development view, the view name is the same as the stream name.
- For an integration view, the view name is *user-ID_integration-stream-name*

If you change the name of the view, we recommend choosing a name that indicates the view's owner and associated stream.

View Storage for Dynamic Views

The first time you create a dynamic view, the wizard prompts you to provide a pathname for the view storage directory. The location you choose becomes the default for other dynamic views that you create. ClearCase uses this directory to keep track of which versions are checked out to your view and to store view-private objects. The view storage directory is for ClearCase administrative purposes only. Do not modify anything in it.

The size of the view storage directory depends on the following factors:

- Whether you use the **clearmake** *build auditing* and *build avoidance* features
- The size and number of view-private files

For more information, refer to the **clearmake** and **view** reference pages in *ClearCase Reference Manual*.

Valid Locations for Dynamic View Storage Directories

Consider the following restrictions when choosing a dynamic view storage directory location:

- The directory must be located on a ClearCase host. View processes (specifically, **view_server** processes) run on the machine that physically stores the view storage directory, and only ClearCase hosts can run view processes.
- To maintain data integrity, the view storage directory must remain connected to the network. For example, do not locate the view storage directory on a removable storage device.
- If you locate the view storage directory on a laptop and then disconnect the laptop from the network, all of the following restrictions apply:
  - You cannot use the dynamic view.
  - Team members who try to start your view from their hosts will receive error messages from ClearCase.
  - Any **clearmake** process that attempts to wink in a derived object from your view will spend some amount of time trying to contact your view. If it cannot contact your view, it will not consider derived objects in your view as *winkin* candidates for 60 minutes. (You can change the amount of time by setting the **CCASE_DNVW_RETRY** environmental variable.) For more information, see the **clearmake** reference page.

- If you use the view on several hosts, make sure that the location can be accessed by all those hosts; that is, choose a disk partition that is exported.
- If your ClearCase administrator sets up storage locations (which are directories on ClearCase server hosts), you can locate your dynamic view storage directory in a storage location (with **mkview –stgloc**). However, for best performance, we recommend that you locate dynamic view storage directories on your local host.

We recommend that you make the view storage directory accessible to any data backup schemes your organization institutes.

2.6 Choosing Components to Load into Snapshot Views

If you choose to create any snapshot views, the last step of the Join Project Wizard prompts you to choose the components to load.

Refining the List of Source Files

To save disk space, to reduce the time needed for the initial loading operation, and to reduce time needed for rebase operations, clear the check box for any components that you do not need to complete your work

After the Join Project Wizard creates your snapshot views, you can further refine the list of elements in the views by modifying their *load rules*. For more information, see *To Change Which Elements Are Loaded into a Snapshot View* on page 102.

Loading Elements

If your development view is a snapshot view, select **Load the development snapshot view now** on this step of the wizard. When you finish the wizard, ClearCase loads your development view with the elements you selected.

2.7 Accessing Your Development View

After creating your development view, the Join Project Wizard offers to open your development view in a shell or **xclearcase**, a GUI from which you can browse files and directories.

To access your development view without the help of the Join Project Wizard or to access someone else's view, refer to the following sections.

Accessing a Snapshot View

Recall that when you create the view, ClearCase loads one version of each element in the project's *baselines* into your view. To access the files loaded into a view, change to the root directory of the view.

For example, when creating the view you provide this pathname:

~/pat_v1.4_cropcircle

The view's files are located in the **~/pat_v1.4_cropcircle** directory. (See *Locations for Snapshot Views* on page 31 for more information.)

Accessing Someone Else's Snapshot View

You can access someone else's snapshot view as you would access any other directory on another workstation. Assuming that you can access the other workstation and that the directory's owner has set up the proper permissions, use the **cd** command to access the view.

Accessing a Dynamic View

To access source files from a dynamic view, you must set a view and mount VOBs.

To Set a Dynamic View

From a shell, enter the following command:

cleartool setview *view-tag*

You determine the view-tag when you create the view. (See Figure 13 on page 30 for an illustration of providing the view-tag in the Join Project Wizard.)

For more information on setting a view, see the **setview** reference page in *ClearCase Reference Manual* or enter **cleartool man –graphical setview** in a shell.

To Mount VOBs

Type this command:

cleartool mount *VOB-tag*

Usually, ClearCase mounts VOBs that were created with a public VOB-tag when you start or reboot your workstation. If public VOBs do not mount, type **cleartool mount –all** to mount them.

VOBs remain mounted until you reboot your workstation or unmount them with the **cleartool umount** command. For more information on mounting VOBs, see the **mount** reference page in *ClearCase Reference Manual*.

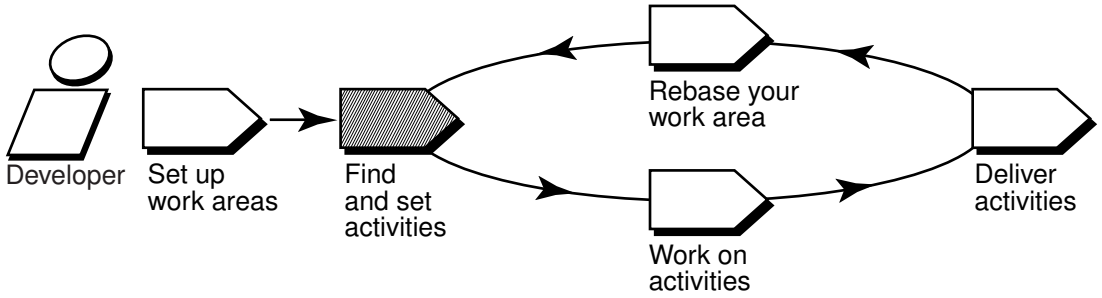
2.8 Logging In to a ClearQuest User Database

In a project that is enabled for ClearQuest, the first time you perform a ClearCase operation that involves the UCM-ClearQuest integration, such as setting your view to an activity, ClearQuest prompts you to enter your ClearQuest user ID and password.

ClearQuest keeps its own database of user IDs and passwords. (ClearCase uses your standard network user ID and password.) Make sure your ClearQuest administrator has added a user ID and password to the ClearQuest user database for you.

When ClearQuest prompts you, be sure to enter your ClearQuest user ID in this dialog box; ClearQuest stores your login information so it doesn't need to prompt you again during ClearCase operations.

3 Finding and Setting Activities



Before you start work on the project, find any *activities* that your project manager or other team members have already assigned to you; if necessary, add new ones to indicate the scope and amount of work scheduled for the project. This chapter describes the following tasks:

- Finding activities
- Creating activities
- Assigning activities in a project that is enabled for ClearQuest
- Setting activities

NOTE: Both ClearCase and ClearQuest provide a Web interface through which you can complete some of the tasks described in this and subsequent chapters. (You cannot set an activity or issue other ClearCase commands from the ClearQuest Web interface.) Ask your system administrator how to access the Web interfaces. Then use the Web interface's online help for more information.

3.1 Finding Activities

UCM provides several locations from which to find activities:

- **MyToDoList** and other ClearQuest queries (only for projects that are enabled for ClearQuest)
- Project Explorer
- **cleartool lsactivity**
- ClearCase dialog boxes

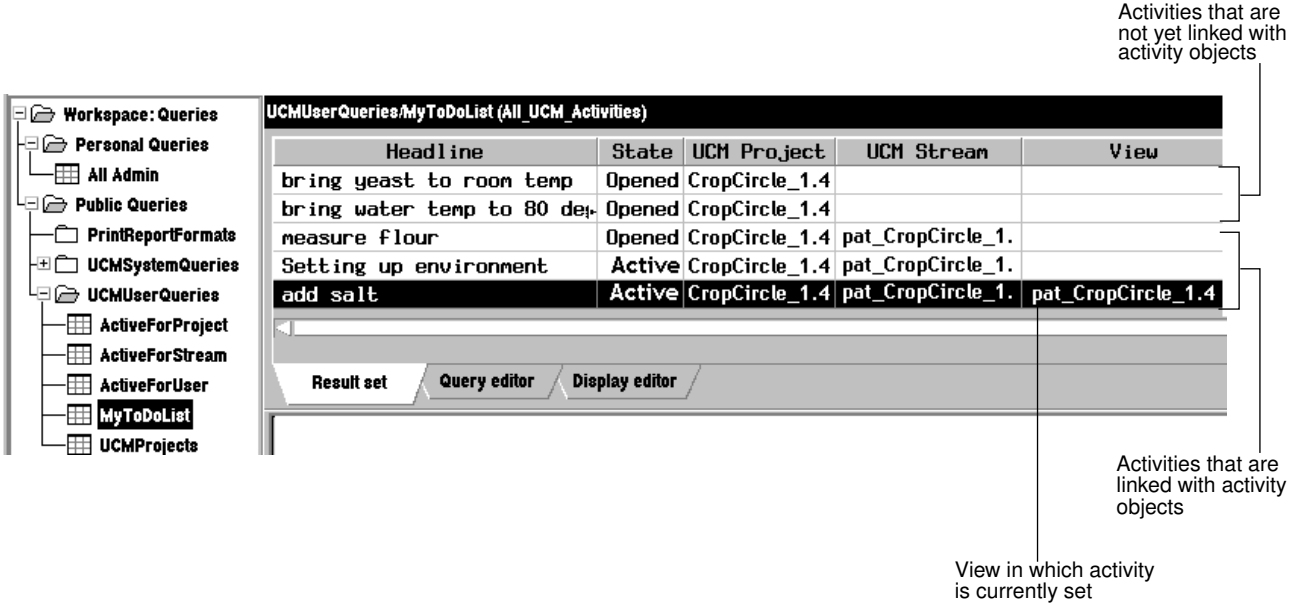
Finding Activities with MyToDoList and Other ClearQuest Queries

If your project is enabled for the UCM-ClearQuest integration, use ClearQuest to find activities.

Using MyToDoList

The MyToDoList query finds all UCM-enabled records that are assigned to you in ClearQuest, even if they have not been linked with activity objects or if they have been linked with activity objects in someone else's stream (Figure 14). For more information, see *Queries* on page 18 and *UCM-Enabled Record Types and ClearCase Activities* on page 17.

Figure 14 Linked Activities in ClearQuest MyToDoList



To Open MyToDoList

1. Enter the following command:

   clearquest

2. In ClearQuest, in the left pane, click **Workspace>Public Queries>UCMUserQueries**. Then run the **MyToDoList** query.

   ClearQuest displays the query results in the Query results pane.

3. To arrange the list by information displayed in a column, click the **Display Editor** tab and change the values in the **Sort** and **Sort Order** columns. Then rerun the **MyToDoList** query.

4. To see more information about an item on your to do list, select it. ClearQuest displays the record's details in the Record form pane.

Creating Your Own ClearQuest Query

If none of the existing queries suits your needs, you can create a new one. When creating a query to find UCM-enabled records, note the following:

- To search across all UCM-enabled record types, in the **Choose a Record** dialog box, select **All_Ucm_Activities**. This selection limits your query to fields common to all UCM-enabled record types.
- You cannot use a ClearQuest query to find ClearCase change-set information.

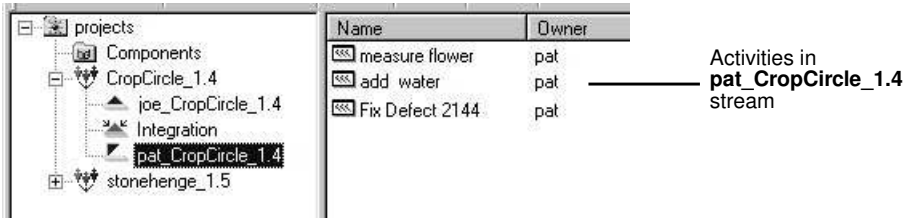
For more information on creating a query, refer to the ClearQuest online help.

Finding Activities with Project Explorer

The Project Explorer arranges activities hierarchically by stream and by project. It shows only the ClearCase activity objects that are in the currently selected stream.

In a project that is enabled for ClearQuest, each activity in a stream is linked to a UCM-enabled record in ClearQuest (Figure 15). For more information, see *UCM-Enabled Record Types and ClearCase Activities* on page 17.

Figure 15 Activities in Project Explorer



To start the Project Explorer, enter the following command:

clearprojexp

Finding Activities with cleartool lsactivity

You can use **cleartool lsactivity** to find activity objects in streams. This section describes how to find activity objects in your development stream. For information on finding other activity objects with this command, see the **lsactivity** reference page.

To List the Activity Objects in Your Development Stream

1. Access your development view. (For more information, see *Accessing Your Development View* on page 35.)

2. Enter the following command:

   cleartool lsactivity –cview

For an example, see Table 2.

Table 2 Using the lsactivity command

```
1  % cleartool setview pat_1.4_cropcircle
2  % cd /guivob
3  % cleartool lsactivity –cview
4  05–Aug.09:14:17  set_up_directories  pat "set up directories"
5  06–Aug.14:17:19  activity0863.3821  chris "fix copyright strings"
```

Table 2 shows that there are currently two activities in the stream attached to **pat_1.4_cropcircle**: `set_up_directories` and `activity0863.3821`. The string `05–Aug.09:14:17` in line 4 indicates the activity's creation date and time, the strings `set_up_directories` provides the activity' s name used in the activity-selector, the string `pat` is the activity creator, and the string `"set up directories"` is the activity headline. For more information, see *Under the Hood: Different Ways to Refer to Objects in a Project* on page 28 and the **lsactivity** reference page in *ClearCase Reference Manual*.

In a project that is enabled for ClearQuest, each activity displayed is linked to a UCM-enabled record in ClearQuest. For more information, see *UCM-Enabled Record Types and ClearCase Activities* on page 17.

Finding Activities with ClearCase Dialog Boxes

ClearCase provides a GUI, the File Browser, from which you can browse the VOB namespace and issue **checkout** and **checkin** commands for the elements in your view. When you issue a **checkout** or **checkin** command from the File Browser, ClearCase opens a dialog box that prompts you to enter a version creation comment and to set an activity (Figure 16.).

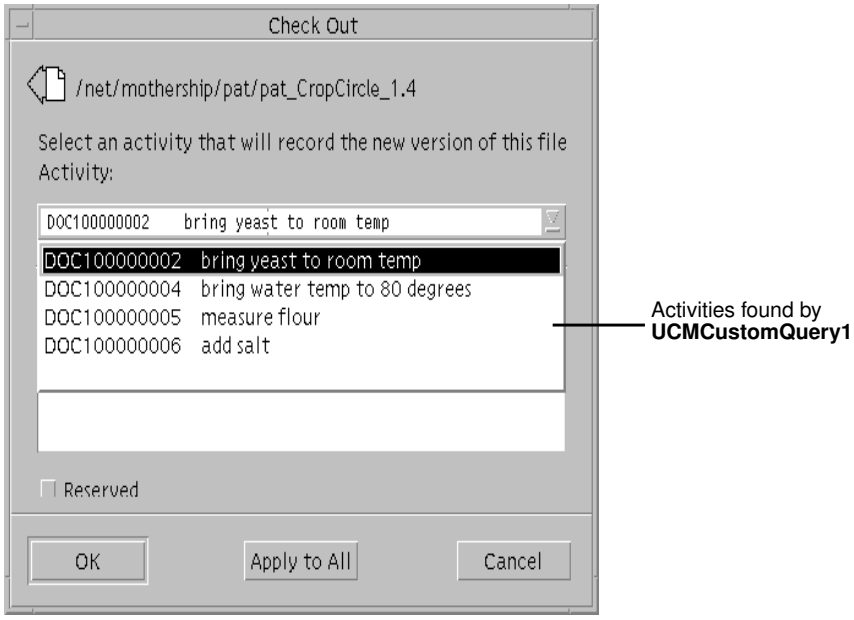
In a project that is enabled for ClearQuest, the dialog boxes display the names of activities found by the UCMCustomQuery1 query. (For more information, see *UCMCustomQuery1* on page 20.)

In projects that do not use the UCM-ClearQuest integration, the dialog boxes show the names of activity objects in the stream that is associated with the view from which you issued the command.

To start the File Browser, enter the following command:

xclearcase

Figure 16 Activities in the Check Out Dialog Box (Project That Is Enabled for ClearQuest)



3.2 Creating Activities

Your organization may have policies regarding the scope of an activity. For example, these policies may require that all activities refer to a change request in your change request management system. The process for creating activities differs, depending on whether your project uses the UCM-ClearQuest integration.

This section describes the following tasks:

- Creating activities in a project that is enabled for ClearQuest
- Creating activities in a project that is not enabled for ClearQuest

Creating Activities in a Project That Is Enabled for ClearQuest

In a project that is enabled for ClearQuest, *activity* usually refers to two objects that are linked together: a ClearCase activity object and a UCM-enabled record in a ClearQuest database. The process of creating an activity in a project that is enabled for ClearQuest entails the creation of two objects in a specific order:

1. You create ClearQuest records (based on a UCM-enabled record type). You can create ClearQuest records only from ClearQuest.

2. When you set a view to an activity (which you can do by clicking **Actions>WorkOn** for a ClearQuest record), ClearCase creates an activity object in the stream that is associated with the view and links the activity object to the UCM-enabled record in ClearQuest. You do not create ClearCase activity objects directly.

To Create Activities from ClearQuest

NOTE: You can create and use a template of default values for the ClearQuest Submit form. This can save time and be helpful when you are entering multiple, related records. For information on using the template, refer to ClearQuest online help.

1. Enter the following command:

   clearquest

2. In ClearQuest, click **Actions>New**.

3. In the **Choose a Record Type** dialog box, select an appropriate UCM-enabled record type. (Your project leader sets up the record types that are in your ClearQuest database.) Then click **OK**.

   ClearQuest highlights in red the fields that are required and the fields that contain invalid values. If a field contains an invalid value, you can right-click the field and then click **Error Message** to see any message your project manager may have included. Your project manager can include explanations for each field. If you have questions about what type of information a field requires, right-click the field and then click **Help** on the shortcut menu.

4. Complete the required fields in the ClearQuest Submit form.

5. In the **UCM Project** list, select your project. (Your project manager determines the location for this list when she creates the corresponding record type.) Then click **OK**.

After you create an activity from ClearQuest, you must complete an additional procedure to make the activity appear in MyToDoList. (See *To Make Activities Appear in MyToDoList* on page 46.)

Creating Activities in a Project That Is Not ClearQuest-Enabled

In UCM projects that do not use the UCM-ClearQuest integration, you create activity objects directly in a specific stream.When you add an activity, it remains in your stream unless you use **rmactivity** to remove it.

To Create Activities in a Project That Is Not ClearQuest-Enabled

This procedure describes adding an activity to your development stream. For information on adding activities to a different stream, see the **mkactivity** reference page in *ClearCase Reference Manual*.

1. Access your development view. (For more information, see *Accessing Your Development View* on page 35.)

2. Enter the following command:

   cleartool mkactivity [–nset] –headline *description-of-activity activity-selector*

Use the **mkactivity** options as follows:

–nset
: Prevents ClearCase from setting your view to the new activity.

–headline
: Provide a name that ClearCase GUI tools use to identify the activity. Create a headline that describes the scope and purpose of the activity. If the headline contains space characters, enclose it in double quotes.

activity-selector
: Provide an alphanumeric string to use when issuing **cleartool** commands. If you do not provide an activity-selector, **mkactivity** creates one.

For example, use the following command to create an activity for changing copyright strings:

**cleartool mkactivity –headline "Changing Copyright Strings" **
changing_copyright_strings

For more information, see the **mkactivity** reference page in *ClearCase Reference Manual*.

3.3 Assigning Activities in a Project That Is Enabled for ClearQuest

The concept of assigning an activity applies only to UCM-enabled records. In general, *assigning* an activity means making a person responsible for some portion of an activity's development. An activity assignment may change several times during a development cycle. For example, a project manager may take initial responsibility for an activity. When she sets the schedule for a given project, she assigns the activity to a developer on the project. After a developer begins work on an activity, the activity assignment usually doesn't change.

ClearQuest records include fields to keep track of an activity's current owner. Your organization may restrict who can change the value in the owner field.

NOTE: ClearCase activity objects exist in a given stream, and they record the activity creator, but these data points cannot be changed and are not intended to express the concept of assignment.

To Make Activities Appear in MyToDoList

By default, the MyToDoList query finds records that are both in a Ready or Active state type and assigned to you. To make an activity appear in MyToDoList, you must change its state type and owner.

1. In ClearQuest, run a query to find an activity. Then click the activity in the Query editor pane.
2. In the Record form pane, click **Actions** and choose a command that moves the activity to a state based on the Ready state type.
3. In the **Owner** list, select your user ID.
4. To see the activity in MyToDoList, in the left pane, click **Workspace>Public Queries >UCMUserQueries**. Then run the MyToDoList query.

3.4 Setting Activities

To start working on an activity, you must set your development view to the activity.ClearCase assigns all work in your view to the currently set activity, adding each version you create in your view to the activity's change set.

You can set activities from the following locations:

- MyToDoList or the results of any other ClearQuest query (only for projects that are enabled for ClearQuest)
- From a shell with **cleartool setactivity**
- Dialog boxes that open during checkout and checkin operations

This section also describes the following tasks:

- Unsetting an activity
- Seeing which activity is currently set in the view

Setting Activities from ClearQuest

The first time you set an activity in a project that is enabled for ClearQuest, you must set it from ClearQuest as described in this section. Because you cannot create ClearCase activity objects directly in these projects, you must use ClearQuest to create the activity object. For more information, see *Creating Activities in a Project That Is Enabled for ClearQuest* on page 43.

After ClearCase creates the activity object in your stream, you can use **cleartool setactivity** (as described in *To Set an Activity with cleartool setactivity* on page 48) if you need to set the activity subsequently.

To Set Activities from ClearQuest

1. In ClearQuest, do **one** of the following:

   - Display MyToDoList by entering the following command:

     clearquest

     Then in the left pane, click **Workspace>Public Queries>UCMUserQueries** and run the MyToDoList query.

   - Use another query to find the activity you want to set.

2. In the Query results pane, click the activity you want to set.

3. In the Record form pane, if it isn't already selected, select your project in the **UCM Project** list. (Your project manager determines the location for this list when she creates the corresponding record type.)

4. Click **Actions>WorkOn**.

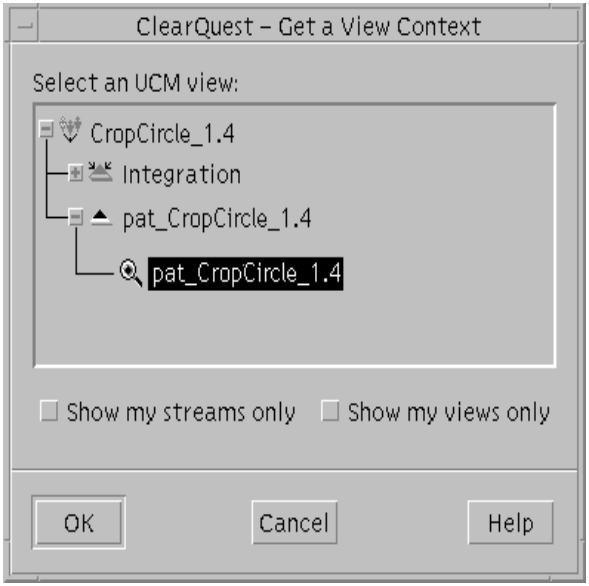
5. In the **Select View** dialog box, select **Show Only My Streams**.

6. Click the plus sign (**+**) next to your stream; then select your development view and click **OK**. (Figure 17)

The status bar in ClearQuest indicates that your view is being set to the activity.

To start working on your project's source files, navigate to your view and check out the source files.

Figure 17 Select Your View



Setting Activities with cleartool setactivity

In a project that is enabled for ClearQuest, before you use **cleartool setactivity**, a ClearCase activity object must exist in the stream and you must know the activity's ClearQuest record ID. For information about creating an activity object, see *Setting Activities from ClearQuest* on page 47 and *Creating Activities* on page 43. For information on finding an activity's activity selector, see *Finding Activities with cleartool lsactivity* on page 40.

To Set an Activity with cleartool setactivity

1. Access your development view. (For more information, see *Accessing Your Development View* on page 35.)
2. Enter the following command:

   cleartool setactivity *activity-selector*

For example, in a project that is not enabled for ClearQuest, enter the following command:

cleartool setactivity changing_copyright_strings

When executed in a view that is associated with a ClearQuest-enabled project, the **setactivity** command takes an activity-selector that is a ClearQuest record ID of an existing ClearQuest record. For example:

cleartool setactivity SAMPL123456

You can set only one activity per view at a time. ClearCase associates all checkouts in your view with the currently set activity until you unset the activity or set another one.

For more information, see the **setactivity** reference page in *ClearCase Reference Manual*.

Setting Activities from ClearCase Dialog Boxes

ClearCase provides the File Browser, from which you can browse the VOB namespace and issue **checkout** and **checkin** commands for the elements in your view. When you issue a **checkout** or **checkin** command from the File Browser, ClearCase opens a dialog box that prompts you to enter a version creation comment and to set an activity.

In a project that is enabled for ClearQuest, the dialog boxes list the names of activities found by the UCMCustomQuery1 query. (For more information, see *UCMCustomQuery1* on page 20.)

In projects that do not use the UCM-ClearQuest integration, the dialog boxes show the names of activity objects in the stream that is associated with the view from which you issued the command.

Unsetting Your View from an Activity

You can change any view to have no currently set activities, for example, if you need to remove the view or remove the activity currently set in the view.

To unset your view from an activity, enter the following command from a view:

cleartool setactivity –none

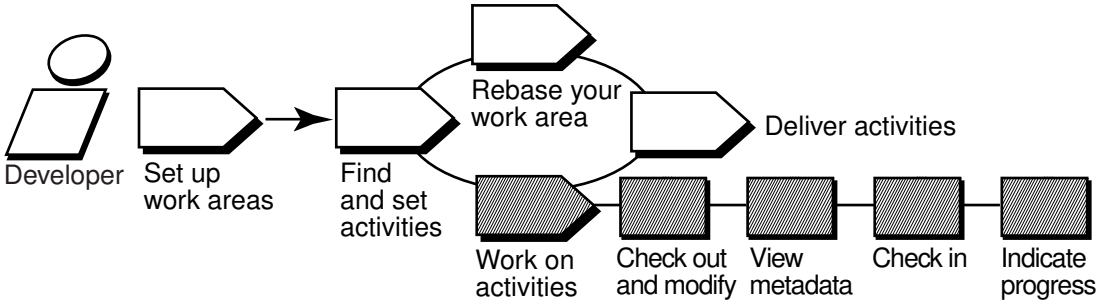
To See Which Activity Is Currently Set

1. Access your development view. (For more information, see *Accessing Your Development View* on page 35.)

2. Enter the following command:

   cleartool setactivity –none

4 Working on Activities



Working on activities involves the following tasks:

- Checking out elements
- Working with checkouts
- Canceling checkouts
- Checking in elements
- Testing your work
- Indicating your progress

The work in your development work area is unavailable to the project team until you deliver it.

4.1 Checking Out Elements

To modify files and directories under ClearCase control, you must check them out. Checking out does not add versions to the activity that is currently set in the view. (Checking in creates new versions, which ClearCase adds to the activity's change set.)

Placing files and directories under source control is a separate procedure described in *Adding Files and Directories to Source Control* on page 95.

NOTE: For information on checking out VOB links in a snapshot view, see *Under the Hood: VOB Links* on page 53.

To Check Out Elements

1. In a view, enter this command:

   cleartool checkout –query *list-of-elements*

   ClearCase prompts you to enter a comment.

2. Describe the changes you plan to make.

3. To finish entering comments, press RETURN, and type a period or press CTRL+D on a blank line.

   You can cancel the checkout operation by entering a standard interrupt signal such as CTRL+C before typing a period or pressing CTRL+D.

cleartool checkout includes several options. These are most commonly used:

–query
: Detects potential problems in the checkout process caused by inappropriate config specs or out-of-date snapshot views and prompts for action.

–nc
: Prevents ClearCase from prompting for a comment.

–cq
: Prompts for and applies a comment to all elements in the list.

ClearCase assigns the version you checked out to the activity currently set in your view.

For more information, see the **checkout** reference page in *ClearCase Reference Manual*.

Under the Hood: VOB Links

A VOB link makes a file element or directory element accessible from more than one location in the VOB namespace. There are two kinds of VOB links: *symbolic links*, which are available for file and directory elements, and *hard links*, which are available for file elements only. We recommend that you use VOB symbolic links instead of VOB hard links whenever possible.

You use the **cleartool ln** command to create VOB links. See the **ln** reference page in *ClearCase Reference Manual* for more information.

Symbolic Links and Hard Links in Dynamic Views

In *dynamic views* (which use the MVFS, or multiversion file system), VOB links behave similarly to symbolic links or hard links in a UNIX file system: symbolic links point to a file or directory element in a different location, and hard links are alternate names for a single file element.

You cannot check out a VOB symbolic link; you must check out the symbolic link target.

When you check out a hard-linked element from a given pathname, ClearCase considers other pathnames for the element as "checked out but removed." That is, to prevent you from modifying the element from multiple pathnames, ClearCase executes standard checkout behavior at only one pathname (the one from which you entered the checkout command), but does not create view-private files at other pathnames. For information about standard checkout behavior, see the **checkout** reference page in *ClearCase Reference Manual*.

Symbolic Links in Snapshot Views

Snapshot views created from a UNIX host maintain standard symbolic link behavior.

NOTE: When you create a snapshot view from a UNIX host, ClearCase assumes that the file system that contains the view supports symbolic links. If your file system does not support symbolic links, ClearCase reports errors if it encounters VOB links during the update operation.

If a *load rule* selects a symbolic link, ClearCase copies the link as well as the link target into the snapshot view (regardless of whether a load rule selects the link target). As with dynamic views, you cannot check out a symbolic link; you must check out the symbolic link target.

Hard Links in Snapshot Views

Instead of creating hard links in a snapshot view, each time a load rule selects a hard link, ClearCase loads the element into the view as a standard file.

Caution: Losing Data Because of VOB Hard Links

If you load multiple instances of a hard-linked element into a snapshot view, you must be careful to check out, modify, and check in only one instance of the file.When you check in a hard-linked file, ClearCase updates all other instances in your view, which could result in loss of data if you modified multiple copies of the same file. (Note that, when updating instances of files because of a checkin, ClearCase renames any *hijacked* file to *filename*.**keep** before updating it.)

For example, the following sequence of events will lead to lost data:

1. You check out the hard-linked file **foo/util.c**.
2. ClearCase removes the read-only permission from **util.c** in the **foo** directory only (which is the location from which you issued the **checkout** command).
3. You modify **foo/util.c** but do not check it in.
4. Later, you lose track of which file you checked out. You then remove the read-only permission and modify **util.c** in the **bar** directory.
5. You check in **bar/util.c**. Even though you checked out and modified **foo/util.c**, ClearCase does not prevent you from checking in **bar/util.c**; with a VOB hard link, **bar/util.c** is just another name for **foo/util.c**.
6. Any changes you made to **foo/util.c** are lost upon checkin because ClearCase updates all copies of duplicated files when you check in an element. Note that ClearCase does not consider any copy of **util.c** to be hijacked (even if you change permissions), because you checked out the element in the VOB.

4.2 Working with Checkouts

After you check out a file, you do not need to interact with ClearCase until you're ready to check it in. However, you may want to use some ClearCase tools for the following tasks:

- Viewing an element's history
- Comparing versions of elements
- Tracking checked-out versions
- Viewing an activity's change set
- Moving versions to a different activity

Viewing an Element's History

The History Browser displays the history of an element's modifications, including version-creation comments (entered when someone checks out or checks in an element).

To View an Element's History

In a view, enter this command:

cleartool lshistory –graphical *pathname*

You can use this command from a snapshot view whether or not the element specified by *pathname* is loaded into the view.

Comparing Versions of Elements

As you modify source files, you may want to compare versions to answer such questions as these:

- What changes have I made in my checked-out version?
- How does my checked-out version differ from a particular historical version or from the version being used by one of my colleagues?

To Compare with a Predecessor

In a view, enter this command:

cleartool diff –graphical –predecessor *pathname*

To Compare with a Version Other Than the Predecessor

1. In a shell, enter this command:

   cleartool lsvtree –graphical *pathname*

2. In the Version Tree Browser, select a version.

3. Click **Version>Diff>Selected vs. Other**.

4. In the **Enter other versions** dialog box, select other versions and click **OK**.

If you prefer using the command line:

1. Use **cleartool lsvtree** to list an element's versions.

2. Use the **cleartool diff** command with *version-extended naming*. For example, to compare the current version of **foo.c** with version 4**:**

   cleartool diff foo.c foo.c@@/main/4

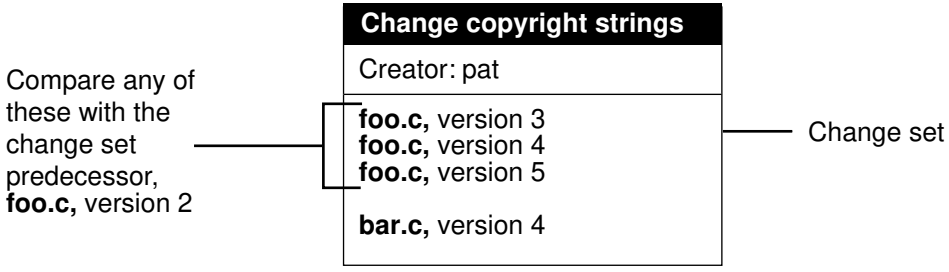
You can use the **lsvtree** and **diff** commands from a snapshot view whether or not the element specified by *pathname* is loaded into the view. For more information, see the **diff** and **pathnames_ccase** reference pages in *ClearCase Reference Manual*.

Comparing with a Change-Set Predecessor

A change set lists all versions you have created while working on a specific activity. In some cases, you create more than one version of an element for a given activity. For example, for the activity **Change copyright strings** you check in a version of **foo.c** and later realize that you missed a string in the file. You then set your view to the activity **Change copyright strings**, check out **foo.c**, modify the string, and check in the file.

To see the changes you have made to an element while working on an activity, you can compare a version in the activity's change set with the version that immediately precedes the change set. In Figure 18, the change set predecessor of **foo.c** is **version 2**. You can compare any version of **foo.c** in the change set with **version 2**.

Figure 18 Compare with Change Set Predecessor



To Compare with a Change-Set Predecessor

1. Display an activity's properties sheet. For example:
   a. Enter **clearprojexp**.
   b. In Project Explorer, navigate to the stream that contains the activity.
   c. Right-click the activity and click **Properties**.
2. In the activity's properties sheet, click the **Change Set** tab.
3. Right-click the version you want to compare and, from the shortcut menu, click **Compare with Change Set Predecessor**.

Tracking Checked-Out Versions

Depending on how you work, you may forget exactly how many and which files are checked out. To list all the files and directories you currently have checked out to your view, access the view and use the **lscheckout** command with the following options:

cleartool lscheckout –cview –me –avobs

For more information, see the **lscheckout** reference page in *ClearCase Reference Manual* or type **cleartool man lscheckout** in a shell.

Viewing an Activity's Change Set

You can see which versions are in an activity's change set from ClearQuest, Project Explorer, or with **cleartool lsactivity**.

To View an Activity's Change Set from ClearQuest

1. Use a query to find the activity. For example, to use MyToDoList:
   a. Enter the following command:

      clearquest

b. In the left pane, click **Workspace>Public Queries>UCMUserQueries** and run the **MyToDoList** query.

   ClearQuest displays MyToDoList list in the Query builder pane.

2. In the Query builder pane, select a record.

3. In the Record form pane, display the **Change Set** list. Your project manager determines the location of this list when she designs the record's record type.

To View an Activity's Change Set from Project Explorer

1. Enter the following command:

   clearprojexp

2. In Project Explorer, select a stream and right-click an activity in the stream; on the shortcut menu, click **Properties**.

3. In the **Properties** dialog box, click the **Change Set** tab.

To View an Activity's Change Set with cleartool lsactivity

To see the change sets for all activities in your stream, enter the following command from your development view:

cleartool lsactivity –long –cview

To see the change set for the currently set activity only, enter the following command from your development view:

cleartool lsactivity –long –cact

For an example, see Table 3. (Lines 9 through 14 display the activity's change set.)

Table 3 Viewing an Activity's Change Set

```
1  % cleartool lsactivity –long –cact
2  activity "Changing Copyright Strings"
3  05-Aug-99.09:14:17 by Pat (pat.user@bread)
4  "Created for the cropcircle_1.4 project on 09/17/99 13:25:24."
5  owner: pat
6  group: user
7  stream: pat_1.4_CropCircle
8  title: changing_copyright_strings
9  change set versions:
10 /guivob/foo.c@@/main/stream990805.090736/5
11 /guivob/spices.c@@/main/stream990805.090736/2
12 /guivob/onions.c@@/main/stream990805.090736/9
13 /guivob/tomatoes.c@@/main/stream990805.090736/1
14 /guivob/flour.c@@/main/stream990805.090736/3
```

NOTE: In a project that is enabled for ClearQuest, **lsactivity** provides the ClearQuest record ID instead of the ClearCase activity headline. For example, in a ClearQuest-enabled project, line 2 in Table 3 would provide the following information:

```
activity "SAMPL123456"
```

Moving Versions to a Different Activity

If you assign one or more versions to the wrong activity or if you create new activities to better represent your work, you can assign the versions to a different activity that is in your stream.

To Move Versions with ClearCase

1. Display an activity's change set. (For more information, see *Viewing an Activity's Change Set* on page 57.)
2. On the **Change Set** tab, select one or more elements.
3. Right-click a selected element, and on the shortcut menu, click **Move to Activity**.

To Move Versions with ClearQuest

1. Display an activity's change set in ClearQuest. For more information, see *To View an Activity's Change Set from ClearQuest* on page 57.
2. Select an element from the **Change Set** list.

3. Right-click the element, and on the shortcut menu, click **Move to Activity**.

4. Follow the ClearCase prompts.

4.3 Canceling Checkouts

If you check out a file but don't want to check in your changes or want to start with a fresh copy, you can cancel the checkout as follows:

1. In the view from which you checked out a file, enter this command:

   cleartool uncheckout *pathname*

   ClearCase prompts you to save your modifications in a view-private file with a **.keep** extension.

2. To save the modifications in a view-private file, press RETURN. Otherwise, enter **no**.

To avoid being prompted about saving modifications, use one of the following options with the **uncheckout** command:

–keep
: Saves modifications

–rm
: Does not save modifications. Any changes you made to the checked-out version are lost.

Under the Hood: Canceling Checkouts

When you cancel the checkout of a file element, ClearCase handles the request as follows:

1. It prompts you to rename the file in your view to *filename***.keep**.

2. It notifies the VOB that you no longer have the version checked out in your view.

3. In a snapshot view, it copies from the VOB the version that was in your view when you performed the checkout operation.

   In a dynamic view, it uses the config spec's version-selection rules to select a version.

Canceling Directory Checkouts

When you cancel a directory checkout, ClearCase notifies the VOB that you no longer have the version of the directory checked out to your view. ClearCase does not prompt you to rename a canceled directory checkout to *directory-name***.keep**.

If you cancel a directory checkout after changing its contents, any changes you made with **cleartool rmname**, **mv**, and **ln** are lost. Any new elements you created (with **mkelem** or **mkdir**) become orphaned. ClearCase moves orphaned elements (and any data that exists in the view at the pathname of the new element) to the VOB's **lost+found** directory under names of this form:

element-name.UUID

In such cases, **uncheckout** displays this message:

```
cleartool: Warning: Object "foo.c" no longer referenced.
cleartool: Warning: Moving object to vob lost+found directory as
"foo.c.5f6815a0a2ce11cca54708006906af65".
```

In a snapshot view, ClearCase does not remove *view-private objects* or start the update operation for the directory in the view. To return the directory in your view to its state before you checked it out, you must start the Update Tool. For information on starting the Update Tool, see ClearCase online help.

In a dynamic view, ClearCase does not remove view-private objects, but it does revert the view to its previous state.

To move an element from the **lost+found** directory to another directory within the VOB, use the **cleartool mv** command. To move an element from the **lost+found** directory to another VOB, use the **relocate** command. For more information about moving elements to another VOB, see the **relocate** reference page in *ClearCase Reference Manual*.

To permanently delete an element in the **lost+found** directory, take note of the orphaned element's name and use this command:

cleartool rmelem *VOB-pathname***/lost+found/***orphaned-element-name*

For example, from a dynamic view:

```
% cleartool rmelem /guivob/lost+found/foo.c.5f6815a0a2ce11cca54708006906af65
```

From a snapshot view:

```
% cd ~/pat_v1.4_cropcircle_sv
% cleartool rmelem guivob/lost+found/foo.c.5f6815a0a2ce11cca54708006906af65
```

NOTE: In a snapshot view, ClearCase treats the **lost+found** directory, which is located immediately below the root directory of a VOB, as any other directory. To load the directory in your view, you must use a load rule that specifies either the element's parent directory or the directory itself. However, as with any other directory in a snapshot view, you do not need to load the **lost+found** directory to issue ClearCase commands for elements in the directory.

4.4 Checking In Elements

Check in when you want to keep a record of the checked-out file or directory in its current state. When you check in from your development view, ClearCase records the new version in your development stream; even though the new version is a permanent part of the element, the modifications are unavailable to the rest of the development team until you deliver them to the integration stream.

To Check In Elements

1. In a view, enter the following command:

   cleartool checkin [**–cact**] *list-of-elements*

   Use the **–cact** option to check in all versions checked out for the activity currently set in your view.

   ClearCase prompts you to append your checkout comments.

2. Type any additional comments, and then press RETURN and type a period or press CTRL+D on a blank line.

   You can cancel the checkin operation by entering a standard interrupt signal such as CTRL+C before typing a period or pressing CTRL+D.

cleartool checkin includes several options. These are the most commonly used:

–nc
: Prevents ClearCase from prompting for a comment.

–cq
: Prompts for and appends a single additional comment to all elements in the list.

For a complete description of all checkout options, see the **checkin** reference page in *ClearCase Reference Manual*.

Snapshot View: Checking In VOB Links

When you issue a checkin command from a snapshot view, ClearCase handles the request as follows:

1. It copies your modifications to the VOB as a new version.

   The version you check in remains in the view, regardless of the view's config spec.

2. It removes write permission for the file.

For any other instance of a hard-linked file loaded into a snapshot view, ClearCase copies the new version from the VOB into your view. (If your load rules specify a hard-linked element that appears in more than one VOB location, the element is copied into each of the appropriate locations in your view's directory tree.)

4.5 Testing Your Work

To reduce the amount of merging needed when you deliver your work, you should build and test the modifications in your development stream, especially after you rebase the stream and before you deliver activities. If your organization uses **clearmake**, you can use these ClearCase build tools for your test builds; however, the *build auditing* and *build avoidance* features are available only from dynamic views.

For more information about using **clearmake**, refer to its reference page in *ClearCase Reference Manual* and *Building Software with ClearCase*.

4.6 Indicating Your Progress

If your project is enabled for ClearQuest, you can complete the following tasks to indicate your progress on an activity:

- Modifying information in ClearQuest
- Closing an activity
- Using schema-specific actions

You may need to reassign or delete an activity regardless of whether your project is enabled for ClearQuest.

Modifying Information in ClearQuest

Your ClearQuest activities may include fields for keeping additional information. For example, an activity record form may contain an **Attachments** tab to link to supporting documents written in other applications. The form may also contain fields to describe a defect's symptoms. Adding or modifying such information does not change the activity's state or have any effect on the corresponding activity in your stream.

For information on adding information to multiple activities, see ClearQuest online help.

To Modify ClearQuest Information

1. Use a query to find the activity. For example, to use MyToDoList:

   a. Enter the following command:

      clearquest

   b. In the left pane, click **Workspace>Public Queries>UCMUserQueries** and run the MyToDoList query.

      ClearQuest displays MyToDoList in the Query builder pane.

2. In the Query builder pane, select a record.

3. In the Record form pane, click **Actions>Modify**.

   Your project manager can include help for the record form. To see information about a field, right-click it and then click **Help** on the shortcut menu.

4. In the Record form, make your changes. (Changing the Owner field reassigns the activity; for information on reassigning an activity, see *Reassigning Activities* on page 67.)

5. To save your changes, click **Apply**. To clear changes you made without saving them, click **Revert**.

Closing Activities

Your project manager may have set up a policy for your project that closes activities in ClearQuest when you deliver them. If the policy is not in effect for your project, close your activity when you have completed the associated work (which is usually when you deliver it).

Completed activities do not appear on MyToDoList, but they remain in your stream and appear in your stream's properties even after you deliver them.

To Close a ClearQuest Activity

Your project manager may have customized your database schema to implement various ways to move an activity to a state based on the Complete state type. The following procedure assumes that your database schema includes an action to close activities:

1. Use a query to find the activity. For example, to use MyToDoList:

   a. Enter the following command:

      clearquest

   b. In the left pane, open **Workspace>Public Queries>UCMUserQueries** and run the **MyToDoList** query.

      ClearQuest displays MyToDoList in the Query builder pane.

2. In the Query builder pane, select a record.

3. In the Record form pane, click **Actions** and select an action that closes activities.

   Your project manager can include help for the record form. To see information about a field, right-click it and then click **Help** on the shortcut menu.

4. Click **Apply**.

Using Schema-Specific Actions

Your ClearQuest database schema, which your project manager can customize (see *The UCM-ClearQuest Schema* on page 17), may support these common change-request management actions:

- Marking activities as duplicate
- Reopening closed activities
- Postponing activities

Your database schema may provide various ways to use these actions. For example, record types that can be postponed may have a **Postpone** command that you can use after clicking **Actions** in the record's Record form pane.

Keep the following UCM-ClearQuest integration concepts in mind while using any schema-specific actions:

- Your project manager can set up each UCM-enabled record type to behave differently; some actions may not be available for all record types.

- You cannot move an activity from one development stream to another. However, you can move versions in an activity's change set to another activity in the same stream (see *To Move Versions with ClearCase* on page 59) or to an activity in a different stream (see *Moving Work from a Development Stream to Another Project* on page 105).

- You cannot delete an activity that is either currently set in a view or that contains versions in its change set. The **UCM View** field in the ClearQuest Query results pane indicates whether an activity is currently set. The **Change Set** list, which is available from the ClearQuest Record form pane or from the activity's properties sheet, displays the activity's change set.

- We recommend against working in a development stream that contains activities you do not intend to deliver. Keeping undelivered activities in your development stream increases the risk that, after a rebase operation, your development stream does not contain a cohesive set of versions. If you begin working on an activity and then postpone it until the next release, we recommend that you move the work to the new project, and then create and work in a new development stream. (For more information, see *Moving Work from a Development Stream to Another Project* on page 105.)

Reassigning Activities

You can assign an activity on MyToDoList to another project team member by changing the activity's owner, unless your project manager has enabled the UCM policy **Check Before Work On**, which prevents you from assigning your work to someone else.

If the ClearQuest activity has been linked with a corresponding activity in your stream (which occurs when you set your view to the activity), you can change the owner in ClearQuest, but you cannot move the ClearCase activity to the new owner's development stream. The **UCM Stream** field in the ClearQuest Query results pane indicates whether an activity is linked with a stream.

To reassign an activity that is linked with a stream, change the activity's owner in ClearQuest, and then do one of the following:

- If you have not created versions for the activity (that is, the activity's change set is empty), you can remove the activity in your development stream. (See *To Delete a ClearCase Activity* on page 68.) Then, when the new activity owner sets his view to the activity, ClearCase creates a new activity in the new owner's stream. The activity's **Change Set** list, which is available from the ClearQuest Record form pane or from an activity's properties sheet, displays its change set.
- If the activity's change set is not empty, the new owner can create and work in a view attached to the stream containing the activity. For information on attaching a view to an existing stream, see ClearCase online help.

To Change an Activity's Owner

1. Use a query to find the activity. For example, to use **MyToDoList**:
   a. Enter the following command:

      clearquest

   b. In the left pane, click **Workspace>Public Queries>UCMUserQueries** and run the **MyToDoList** query.

      ClearQuest displays MyToDoList in the Query builder pane.
2. In the Query builder pane, select an activity.
3. In the Record form pane, click **Actions>Modify**.

4. In the **Owner** field, select the new owner.

5. Click **Apply**.

Deleting Activities

NOTE: Your project manager determines whether developers can delete ClearQuest records of any given record type.

If a ClearQuest activity is linked with a ClearCase activity (which happens if you set a view to the activity) and if the ClearCase activity contains versions in its change set, you must delete the ClearCase activity first.

The **UCM Stream** field in the ClearQuest Query results pane indicates whether an activity is linked with a stream. The activity's Change Set list, which is available from the ClearQuest Record form pane or from the activity's properties sheet, displays its change set.

To Delete a ClearCase Activity

NOTE: You cannot delete an activity if the change set contains one or more versions. We recommend that you use some other action for an activity with a nonempty change set. For example, you can deliver all other activities in the stream, abandon the stream, and create a new one to keep track of your work on the project. But if no other action is appropriate, you must first use **cleartool rmversion** to remove the versions from the VOB. Note that removing versions from the VOB is irreversible and may have unintended consequences, for example, if there are dependencies on the versions that you remove.

1. Enter the following command:

   clearprojexp

2. In the left pane of Project Explorer, navigate to your project. Then select the stream that contains the activity.

3. In the right pane, select the activity.

4. Click **File>Delete**. If the activity is linked with a record in ClearQuest, you must delete the ClearQuest record from the ClearQuest GUI.

To delete an activity from a shell, use **cleartool rmactivity**. For more information about **rmactivity**, see *ClearCase Reference Manual*.

To Delete a ClearQuest Activity

1. Use a query to find the activity. For example, to use **MyToDoList**:

   a. Enter the following command:

      clearquest

   b. In the left pane, click **Workspace>Public Queries>UCMUserQueries** and run the MyToDoList query.

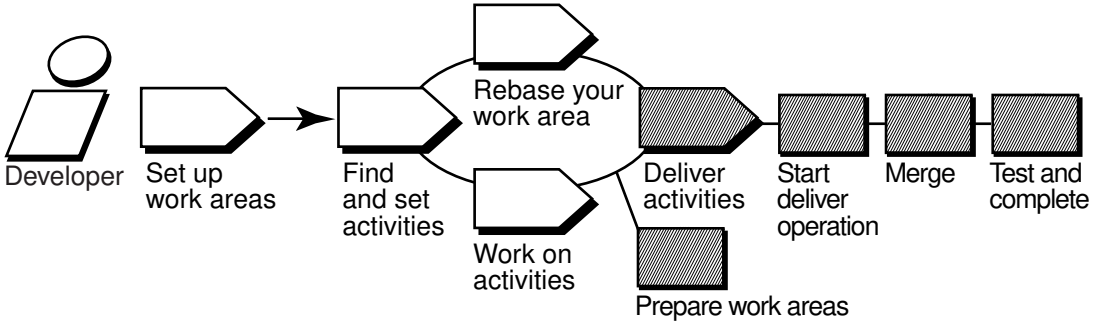
      ClearQuest displays MyToDoList in the Query builder pane.

2. In the Query builder pane, select an activity.

3. In the Record form pane, click **Actions>Delete**. If a ClearQuest activity is linked with a ClearCase activity (and the activity's change set is empty), ClearQuest deletes the activity in the stream as well.

4. Click **Apply**.

5 Delivering Activities



When you're ready to make the activities in your development stream available to the rest of the project team (for example, to contribute to official project builds), you deliver your work. Delivering activities involves the following tasks:

- Preparing your work areas
- Starting the deliver operation
- Selecting activities
- Merging versions
- Testing your work
- Completing the deliver operation

NOTE: If your project uses MultiSite to share source data with developers in other geographical locations and if the project's streams are mastered at a different site, you complete only the first three tasks; the project integrator completes the remaining tasks. For more information, see *MultiSite: Posting Work to Deliver* on page 85.

5.1 Preparing Your Work Areas

Before starting the deliver operation, prepare your work areas as follows:

- If your project manager has created a new *recommended baseline* since you last rebased, rebase your development work area to minimize the amount of merging needed during the deliver operation. For information on rebasing, see Chapter 6, *Rebasing Your Work Area*.
- In your development work area, find, compare, and check in all the work you want to deliver.
- If your integration view is a snapshot view, update it and resolve any *hijacked files*.

Finding, Comparing, and Checking In Work from Your Development View

ClearCase delivers only checked-in changes. In addition, your project manager may have set up your UCM project to require that you check in all files and directories in your development view before starting the deliver operation.

To Find, Compare, and Check In Your Work

1. From your development view, enter **cleartool lscheckout** with these options:

   cleartool lscheckout –cview –me –avobs

   lscheckout produces a list of file or directory versions checked out to your development view and indicates which activity each checked-out version belongs to.

   For more information, see the **lscheckout** reference page in *ClearCase Reference Manual*.

2. To see which changes are in a checked-out version:

   - To see the differences between the previously checked-in version and the current version, type this command:
   - **cleartool diff –predecessor** *filename*
   - To see checkin comments, type this command:
   - **cleartool lshistory** *filename*

3. Do **one** of the following:

   - To deliver changes for a given file, check it in by typing this command:

     cleartool checkin *filename*

   - To cancel your changes for a file, undo its checkout by typing this command:

     cleartool unco *filename*

     During the cancel checkout operation, you can choose to save your changes in a *view-private file*.

4. To keep your changes without delivering a file, leave it checked out. If you leave a file checked out, ClearCase delivers the predecessor version in the activity's change set. If the checked-out version is the only version in the change set, ClearCase does not deliver any version of the element.

Updating Your Integration View

During the deliver operation, use your integration view to make sure that your delivered work builds successfully with the work other team members have delivered.

If your integration view is a snapshot view, update it and resolve any *hijacked files* to ensure that it contains the latest versions delivered to the integration stream. For more information, see ClearCase online help and the **view** reference page in *ClearCase Reference Manual*.

To Update a Snapshot View and Resolve Hijacked Files

1. Enter the following command:

   cleartool update –graphical *snapshot-view-pathname*

2. In the **Update** dialog box, click **Advanced** and select an option for resolving *hijacked files*. (Refer to the online help for information on each option.)

3. To see the actions needed to update the view without actually updating, click **Preview only** on the **View** tab.

4. Click **OK**.

   ClearCase displays its progress as it updates the view (or previews the update).

5. In the Update window, check out or undo any hijacked files. For more information, refer to ClearCase online help.

5.2 Starting the deliver Operation

If your project uses the UCM-ClearQuest integration, your project manager may enable a ClearQuest policy called **Check Before ClearCase Delivery**. This policy runs a customized ClearQuest script when you start the deliver operation. For example, your project manager may create a script that checks for dependencies between ClearQuest records. Before you can continue the deliver operation, you must satisfy any requirements established by the **Check Before ClearCase Delivery** policy.

From the Command Line

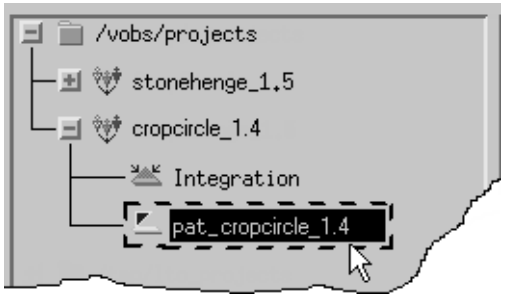
1. Access your development view. (For more information, see *Accessing Your Development View* on page 35.)

2. Enter the following command:

   cleartool deliver –graphical

3. In the **Deliver from Stream** dialog box, select your stream and click **OK** (Figure 19).

Figure 19 Select Your Stream



From the Project Explorer

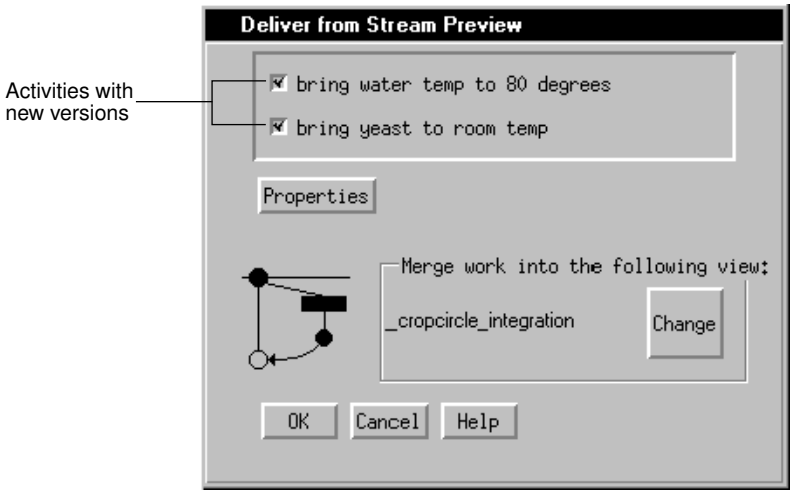
NOTE: To start the Project Explorer, type **clearprojexp**.

1. In the left pane, click + (plus sign) next to the PVOB that contains your project.
2. Click + next to your project.
3. Click your project. (Figure 19)
4. Right-click to display the shortcut menu. Then click **Deliver from Stream**.

5.3 Selecting Activities

After you start the deliver operation, ClearCase opens another **Deliver from Stream Preview** dialog box to display the activities in your development stream that have never been delivered or that include versions created since the last deliver operation (Figure 20).

Figure 20 Undelivered Activities



You can select all activities in the dialog box or a subset. If you select a subset of the activities, ClearCase checks for dependencies between versions in the change sets before continuing the operation. If your selection does not meet dependency requirements, ClearCase prompts you to change it.

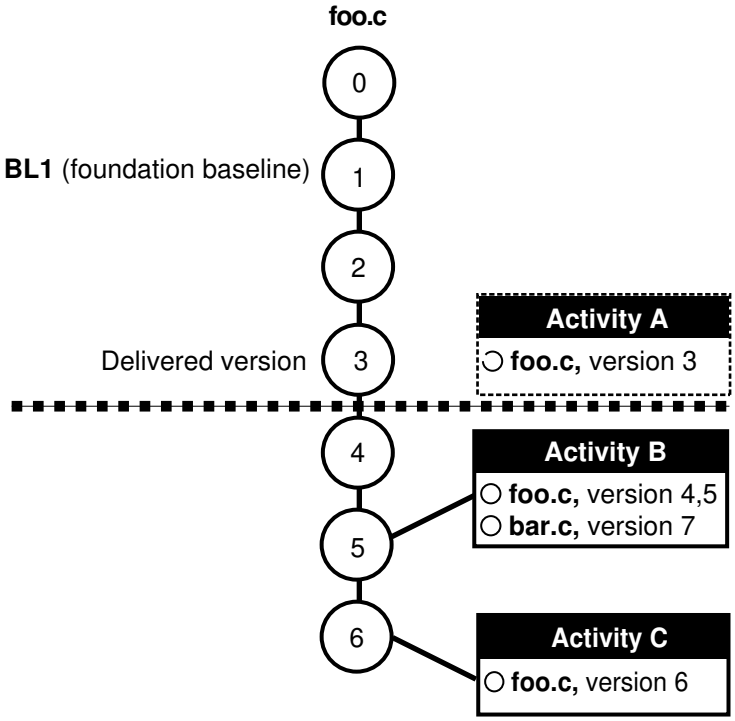
To see an activity's change set, select the activity, click **Properties**, and click the **Change Set** tab in the **Properties** dialog box.

Activity Dependencies in the deliver Operation

For any given element in an activity's change set, you must deliver all versions in your stream in the range between the version in the change set and either the version most recently delivered or the version in the stream's *foundation baseline* (whichever is later). For example, in Figure 21, because **version 3** of **foo.c** was delivered in a previous deliver operation, the dependencies between **Activities B** and **C**, which contain versions of **foo.c**, are as follows:

- You can deliver **Activity B** without delivering **Activity C**.
- To deliver **Activity C**, which contains **version 6**, you must also deliver **Activity B**, which contains the versions between **6** and the most recent delivery of **foo.c**.
- Because **Activity B** also contains versions of **bar.c**, you may be required to deliver other activities to satisfy dependencies for **bar.c**. For example, if **Activity D** (not shown in Figure 21) contained an undelivered **version 6** of **bar.c**, delivering **Activity B** would require you to deliver **Activity D** as well.

Figure 21 Activity Dependencies



5.4 Merging Versions

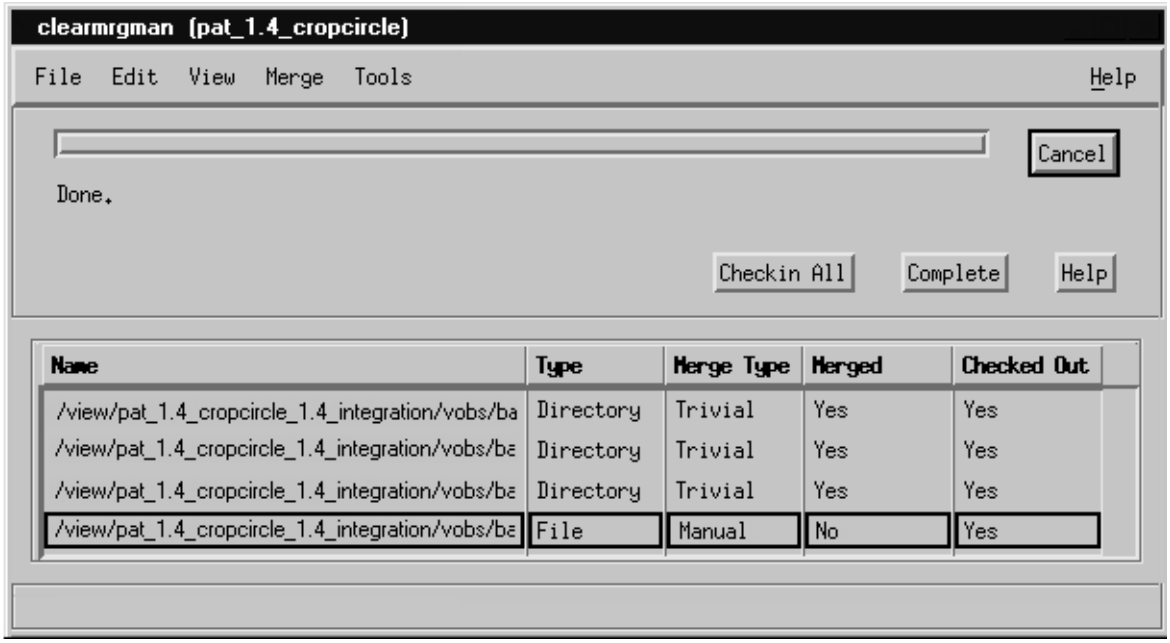
NOTE: If your project uses MultiSite to share source data with developers in other geographical locations, and if the project's streams are mastered at a different site, you do not merge versions. For more information, see *MultiSite: Posting Work to Deliver* on page 85.

After selecting activities, click **OK** in the **Deliver from Stream Preview** dialog box to start merging versions in your development view with versions in the integration view. As it merges files and directories, ClearCase displays its progress in a **clearmrgman** window. (**clearmrgman** is a ClearCase tool used for merging files and directories.)

For each new version delivered to the integration view, ClearCase merges all nonconflicting differences. As it merges versions, ClearCase changes the value in the Merged column of the **clearmrgman** window to **Yes**. If versions in your development view contain changes that conflict with the corresponding versions in the integration view, the value in the Merged column remains

No, even after the **Done** box is completely filled (Figure 22). To merge a version that contains merge conflicts, click the version and click **Merge>Merge Elements.** ClearCase starts Diff Merge, a tool that helps you resolve conflicting differences.

Figure 22 Deliver Progress



Under the Hood: Concurrent Deliver Operations

ClearCase allows developers to deliver activities to the integration stream concurrently. However, if another team member's deliver operation has already checked out a given element, you cannot deliver any changes to that element until that deliver operation is completed or canceled.

When delivering activities, ClearCase follows a specific order:

1. It evaluates each directory sequentially, checking out and merging changes to the integration stream when needed. If ClearCase encounters a reserved directory checkout in the integration stream, it stops the deliver operation (but does not undo any successful directory merges).

2. It then attempts to check out each element that requires a merge. If ClearCase encounters an element that it cannot check out (for example because another team member's deliver operation has already checked out the element), it skips the element and attempts to check out the remaining elements.

   If ClearCase is unable to check out an element, it does not proceed to the following step. In this case, you have two options:

   - Wait until the other developer completes or cancels her deliver operation, and then restart your deliver operation. Note, however, that with a deliver operation in progress, any new versions you create in your development stream are not delivered until the next deliver operation.
   - Undo your deliver operation and continue working on your activities. (See *To Undo a deliver Operation* on page 84 for more information.)

3. After successfully checking out all elements, it merges each element, starting Diff Merge and requesting your input when it encounters merge conflicts.

Under the Hood: Integration Activities and Baselines

ClearCase uses a special kind of activity, an integration activity, to keep track of the changes that occur during deliver and rebase operations. ClearCase creates integration activities and adds versions to them; you do not need to create or maintain them. You may want to view an integration activity's properties to see exactly which versions were merged and created during a deliver or rebase operation. You can view an integration activity's properties from the Project Explorer; the activity name is of the form *deliver.stream-name.date-stamp*.

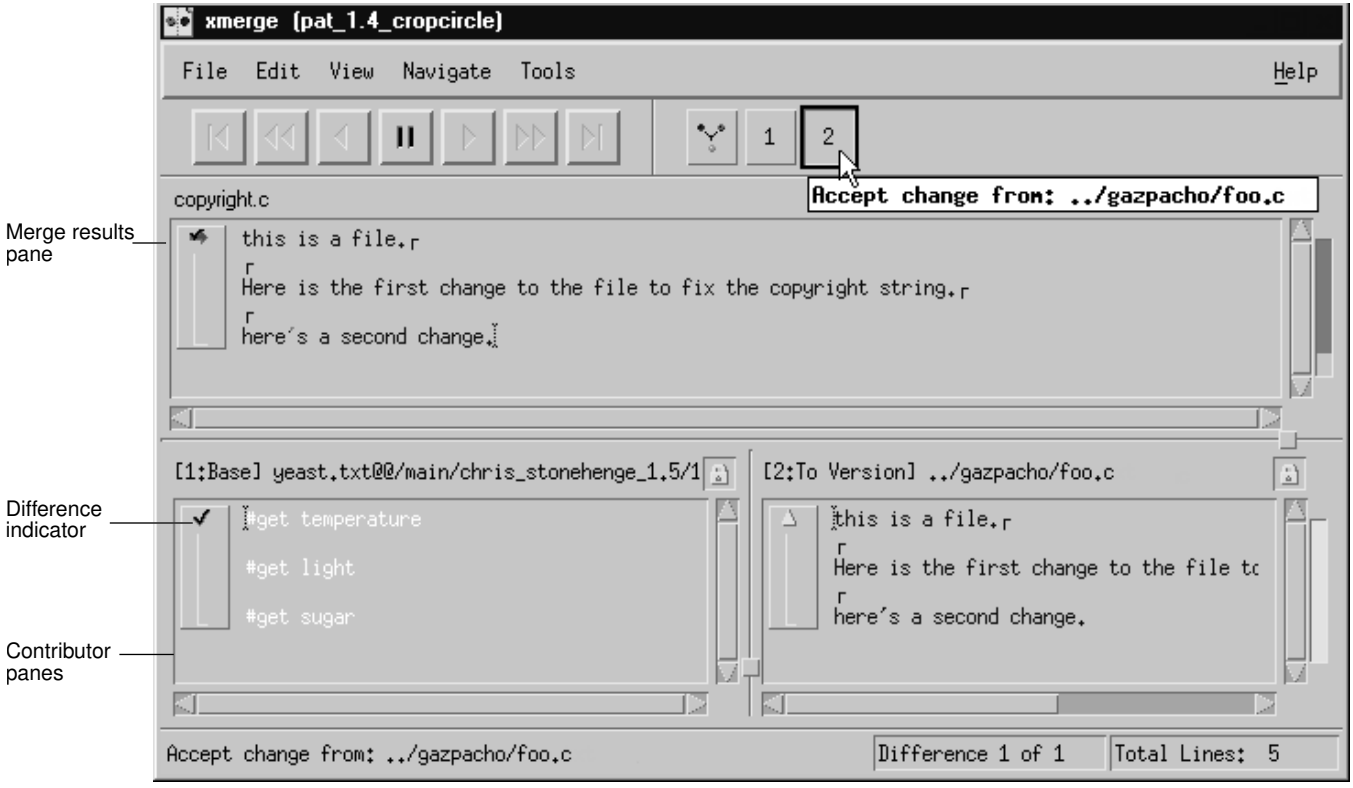
If your project uses the UCM-ClearQuest integration, ClearQuest links a UCM-enabled record based on the UCMUtilityActivity record type with the ClearCase activity. During the deliver or rebase operation, the activity is in a state based on the Active state type. After the operation, ClearQuest moves the activity to a Complete state type.

As an intermediate step in the delivery process, ClearCase creates a baseline in your development stream that identifies the state of your stream at the time of delivery. You can browse this baseline and its properties in the Component Browser. For more information, see *Managing Software Projects with ClearCase*.

Using Diff Merge to Resolve Differences

When Diff Merge starts, it indicates that it has resolved all nonconflicting differences. Then, it highlights the first unresolved difference (Figure 23).

Figure 23 Diff Merge Window



To Resolve Differences

1. In Diff Merge, compare the differences in the contributor panes and click the numbered button on the navigation toolbar that corresponds to the contributor pane number containing the resolution you prefer.

   For example, Figure 23 shows that clicking **2** moves the difference region in contributor pane 2 to the results pane.

In some cases, you can click more than one numbered button to include changes from multiple contributor panes. In addition, after you select a contributor to resolve the difference, you can edit the results pane directly.

2. Click **Navigate>Next Unresolved Difference**.

3. Repeat Step 1 and Step 2 until you have resolved all unresolved differences.

4. Click **File>Save** and then click **File>Exit**.

NOTE: For more information on using Diff Merge, choose a command from the Diff Merge **Help** menu.

After you exit Diff Merge, ClearCase continues merging the versions in your activities.

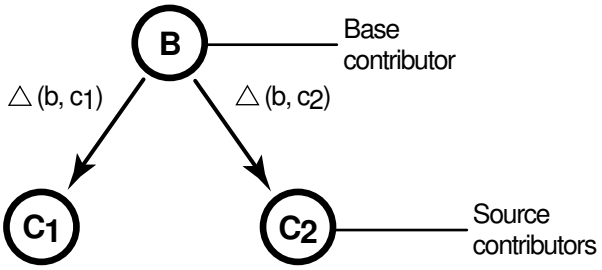
Under the Hood: How ClearCase Merges Files and Directories

A merge combines the contents of two or more versions into a single new version. During a deliver operation, the ClearCase merge algorithm involves the following versions:

- The source contributors: one version from your development stream and one version from the integration stream.
- The base contributor: the common ancestor of the source versions.
- The destination version: the merge output, which becomes a new version in the integration stream.

As illustrated in Figure 24, ClearCase starts with the base contributor and adds changes that are present in the source contributors.

Figure 24 ClearCase Merge Algorithm



If the two source contributors contain conflicting changes (for example, if each contains a line that is different from the base contributor), ClearCase starts the Diff Merge tool so you can choose which change to add to the destination version.

For more information on the ClearCase merge algorithm, see the **merge** reference page in *ClearCase Reference Manual*.

Handling a Binary File in a deliver Operation

If you have a binary file in any of your activities when you perform the deliver operation, and your organization does not have a type manager for that file type, ClearCase generates an error when it encounters the binary file during the delivery. The error occurs because there is no type manager that can merge the version of the file in your development stream with the version of the file in the integration stream. Your project manager can overcome this problem by creating a special element type for the binary file type and associating that element type with the binary file in the VOB. This special element type can tell ClearCase not to attempt merging the file. See *Managing Software Projects with ClearCase* for more information.

Stopping the deliver Operation

You can stop the deliver operation temporarily by clicking **Cancel** in the **clearmrgman** dialog box. Resume the operation by typing **cleartool deliver –resume –graphical** from your development view.

5.5 Testing Your Work

At this point in the deliver process, the destination versions (the new versions resulting from the merges) are checked out and visible only in your integration view.

In addition to your delivered activities, your integration view shows the work other team members have delivered. To make sure that your work is compatible with this, we recommend that you test the work in your integration view before completing the deliver operation.

For information on accessing your integration view, see *Accessing a Dynamic View* on page 35 or *Accessing a Snapshot View* on page 35.

If your organization uses **clearmake**, you can use these ClearCase build tools for your builds; however, the *build auditing* and *build avoidance* features are available only from dynamic views.

For more information about **clearmake**, refer to its reference page in *ClearCase Reference Manual* and to *Building Software with ClearCase*.

In addition to building and testing, you may need to do the following:

- Edit the versions checked out to your integration view to resolve build errors
- Check out and edit additional elements to resolve build errors
- Update your integration view
- Undo a deliver operation

Checking Out Versions

In the process of building and testing, you may need to check out versions in your integration view and modify them to resolve build errors. As described in *Under the Hood: Integration Activities and Baselines* on page 79, ClearCase adds any modifications you make during the deliver operation to an integration activity.

NOTE: We recommend that you postpone checking in your modifications until you complete the delivery. Checking in complicates efforts to undo the deliver operation, if you find it necessary to do so.

Under the Hood: Checking Out Versions from Snapshot Views

If your project allows concurrent deliveries and if your integration view is a snapshot view, the view becomes out of date when other team members deliver to the integration stream. If you try to check out a version that is not the latest in the integration stream, ClearCase prompts you to check out the latest version in the integration stream and loads it into your view.

Undoing a deliver Operation

At any time before completing the deliver operation, you can undo changes made as a result of the operation.

This procedure is valid only if you have started a deliver operation and have not completed it.

To Undo a deliver Operation

1. If you started the Diff Merge tool, close it.
2. If the **Deliver Progress** dialog box is open, click **Cancel**.
3. From your development view, type the following command:

   cleartool deliver –cancel

ClearCase cancels all checkouts in the integration view and removes all *merge arrows* created during the current deliver operation. (ClearCase uses merge arrows in VOBs to keep a history of merges.)

If you checked in any versions to the integration view, using the –**cancel** option is not sufficient to undo the deliver operation. You must either complete the operation (recommended) or use **cleartool rmver –hlink** to remove the versions you checked in before canceling the operation. Because **rmver** erases part of your organization's development history, your organization may not allow individual developers to use it. Because it may have unintended consequences, be very conservative in using this command, especially with the –**hlink** option. For information on using **cleartool rmver**, see the **rmver** reference page in *ClearCase Reference Manual*.

5.6 Completing the deliver Operation

Because the deliver operation is not instantaneous (for example, you may need to build, test, and correct errors several times during a deliver operation), ClearCase imposes a simple state model: deliveries are either *in progress* or *complete*, and you cannot start a new deliver operation from a development stream when one is already in progress for the stream.

To Complete a deliver Operation

If it isn't open, open the **Deliver Progress** dialog box (Figure 22 on page 78) by entering the following command from your development view:

cleartool deliver –complete –graphical

What Happens When You Complete a deliver Operation

When you issue the **complete** command for a deliver operation, ClearCase does the following:

1. It checks in the merge results in the integration stream.
2. If your project is enabled for ClearQuest and your project manager has enabled the DoClearQuestActionAfterDelivery policy, ClearCase executes a ClearQuest script. If the script returns an error, ClearCase stops the deliver operation. You must correct the error before ClearCase can continue.
3. It changes the state of the operation to **Complete**.

5.7 MultiSite: Posting Work to Deliver

If your organization uses MultiSite to distribute development among multiple geographical sites, your project may share source files with developers at other sites. Each site has its own replica of project *components*, and developers work in site-specific replicas. Each replica controls (masters) a set of streams, and only developers at that replica's site can modify them. If your site does not master a stream, you cannot deliver work from or to that stream. Complete the following tasks to make your work on a remote stream available to the rest of the project:

- Prepare your work areas (see *Preparing Your Work Areas* on page 72)
- Start the deliver operation (see *Starting the deliver Operation* on page 74)
- Select activities (see *Selecting Activities* on page 75)

➤ Post work to be delivered

After you post your work, the project integrator at the site that masters your project's integration stream completes the deliver operation.

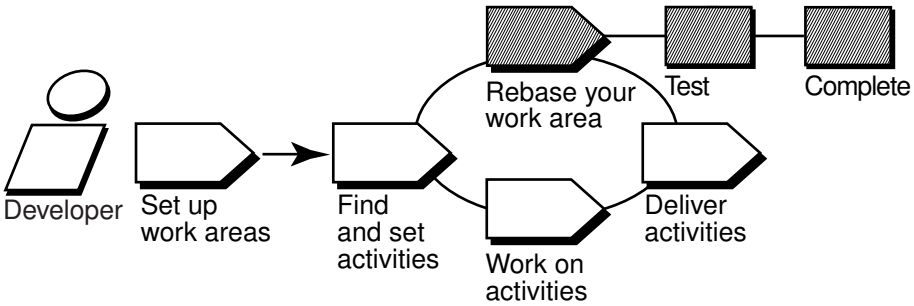
Posting Work To Be Delivered

The **deliver** operation determines whether your development and integration streams are mastered at different replicas. If they are, after you select activities to deliver, ClearCase changes mastership of the development stream object to match that of the integration stream object, and it notifies you that it has assigned the development stream the *posted* status.

After posting your work, notify the project integrator at the site that masters the integration stream's replica. The project integrator can then continue the operation or cancel it to return the development stream to its previous state. Note that, after you post your work, only someone at the master site can cancel or complete the operation.

You can create activities and perform checkins and checkouts for your development stream while the remote deliver is in process. However, you cannot add, remove, or create baselines; add or remove components; or rebase the development stream.

6 Rebasing Your Work Area



Periodically, your project manager groups delivered activities into baselines. Some of these baselines constitute a stable and significant source configuration, and your project manager will recommend that you rebase your development work area to the recommended configuration.

To reduce the number of merges and possible build errors while integrating your work, rebase your development work area each time your project manager creates a new recommended baseline.

Rebasing involves the following tasks:

- Preparing your development view
- Starting the rebase operation
- Testing your development work area
- Completing the rebase operation

6.1 Preparing Your Development View

Before starting, you must check in all files and directories in your development view. You cannot start a rebase operation from a view that contains checkouts.

To Prepare Your Development View

1. From your development view, enter **cleartool lscheckout** with these options:

   cleartool lscheckout –cview –me –avobs

   lscheckout produces a list of file or directory versions checked out to your development view and indicates which activity each checked-out version belongs to.

   For more information, see the **lscheckout** reference page in *ClearCase Reference Manual*.

2. To see which changes are in a checked-out version:

   - To see the differences between the previously checked-in version and the current version, type this command:
   - **cleartool diff –predecessor** *filename*
   - To see checkin comments, type this command:
   - **cleartool lshistory** *filename*

3. Do **one** of the following:

   - To deliver changes for a given file, check it in by typing this command:

     cleartool checkin *filename*

   - To cancel your changes for a file, undo its checkout by typing this command:

     cleartool unco *filename*

     During the cancel checkout operation, you can choose to save your changes in a *view-private file*.

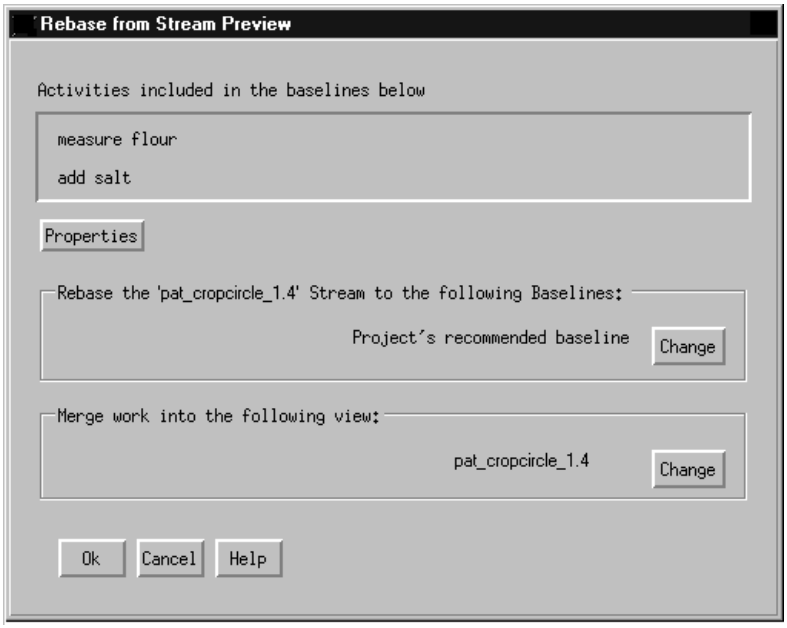
6.2 Starting the rebase Operation

After you prepare your development view (see *Preparing Your Development View* on page 88), you can start the rebase operation. To start the rebase operation, enter the following command from your development view:

cleartool rebase –graphical

ClearCase opens the **Rebase from Stream Preview** dialog box (Figure 25).

Figure 25 Rebase from Stream Preview Dialog Box



By default, the **Rebase from Stream Preview** dialog box presents the latest recommended baseline. To choose a different baseline, click **Change**. You cannot choose a baseline that is older than the baseline currently in your development stream.

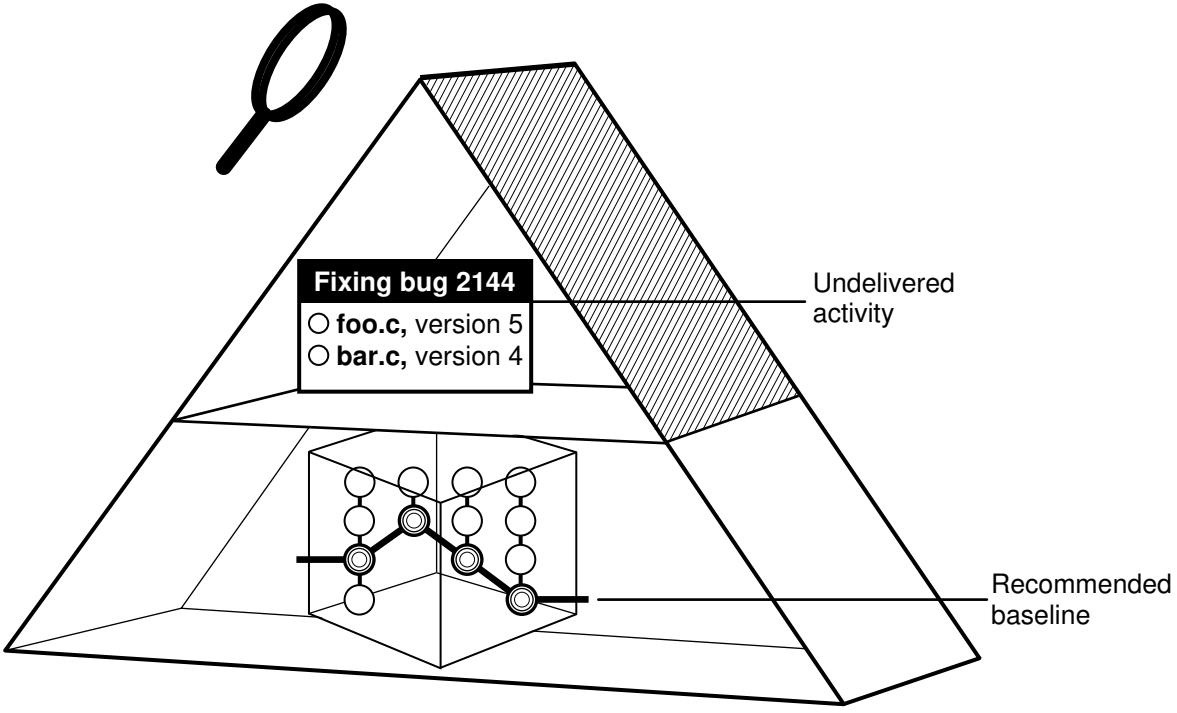
To continue the rebase operation, click **OK**.

Rebasing Your Development Work Area

To rebase your work area, ClearCase takes the following actions:

- It changes your development stream's configuration to select the new recommended baseline. The stream's configuration continues to select any undelivered activities (Figure 26).
- For development views that are *snapshot views*, it starts an update operation to copy versions in the new baseline into your view. If there are other snapshot views attached to your development stream, you must update them so that they contain the changes in the stream. Development views that are dynamic views show versions in the new baseline without requiring an update operation.

Figure 26 Development Work Area After Rebasing



As it rebases your development work area, ClearCase displays its progress in a **clearmrgman** window.

You can stop the rebase operation temporarily by clicking **Cancel** in the **clearmrgman** window. To resume the operation, type this command from your development view:

cleartool rebase –resume –graphical

Merging Versions

For any version you modify in your development stream, if another team member has modified and delivered a version of the same element, you must merge when you rebase to a baseline that contains the delivered version.

For example:

1. As part of working on an activity, you check in a version of **foo.c** from your development view.
2. Another developer checks in and delivers a new version of **foo.c**.
3. Your project manager adds the new version to the recommended baseline.
4. You rebase your development stream. As part of the rebase operation, you must merge the other developer's version into your development stream.

As it does during the deliver operation, ClearCase merges all nonconflicting differences. If versions in the baseline contain changes that conflict with the corresponding versions in your development work area, ClearCase prompts you to start Diff Merge, a tool with which you resolve conflicting differences. For information on using Diff Merge to resolve conflicting differences, see *Using Diff Merge to Resolve Differences* on page 80.

6.3 Testing Your Development Work Area

After ClearCase changes your stream's configuration and completes any needed merges, verify that any undelivered work builds successfully with the versions in the new baseline by initiating a test build in your development view. If your organization uses **clearmake**, you can use these ClearCase build tools for your builds; however, the *build auditing* and *build avoidance* features are available only from dynamic views.

For more information about **clearmake**, refer to its reference page in *ClearCase Reference Manual* and to *Building Software with ClearCase*.

In addition, you may need to do the following:

- Check out elements
- Undo a rebase operation

Checking Out Elements

In the process of building and testing, you may need to check out elements in your development view and modify them. As described in *Under the Hood: Integration Activities and Baselines* on page 79, ClearCase creates an integration activity to record any modifications you make during the rebase operation.

NOTE: We recommend that you postpone checking in your modifications until you complete the rebase. Checking in complicates efforts to undo the rebase operation, if you find it necessary to do so.

Undoing a rebase Operation

At any time before completing the rebase operation, you can undo changes made as a result of the operation.

This procedure is valid only if you have started a rebase operation and have not completed it.

To Undo a rebase Operation

1. If you started the Diff Merge tool, close it.
2. If the **clearmrgman** window is open, click **Cancel**.
3. From your developer view, enter the following command:

   cleartool rebase –cancel

ClearCase cancels all checkouts in the development view and removes all *merge arrows* created during the current operation. (ClearCase uses merge arrows in VOBs to keep a history of merges.)

If you checked in any versions to the development view, you must remove them using the **cleartool rmver** command. Because **rmver** erases part of your organization's development history, your organization may not allow individual developers to use it. Be very conservative in using this command.

For information on using **cleartool rmver**, refer to the **rmver** reference page in *ClearCase Reference Manual*.

6.4 Completing the rebase Operation

As with the deliver operation, ClearCase imposes a simple state model: rebase operations are either *in progress* or *complete*, and you cannot start a new operation for a development stream when one is in progress.

Completing a rebase operation consists of two tasks: checking in any merge results and changing the state of the operation to complete.

After testing your work, click **Complete** in the **clearmrgman** window. ClearCase checks in any versions checked out to your development view and notifies your development stream that the rebase operation is complete.

If you closed the **clearmrgman** window before completing the operation, complete it by entering this command from your development view:

cleartool rebase –complete –graphical

7 Other Development Tasks

Chapter 4, *Working on Activities*, describes tasks that you perform daily or weekly. You may need to perform some of these tasks less often:

- Adding files and directories to source control
- Moving, removing, and renaming elements
- Accessing elements not loaded into a snapshot view
- Adjusting the scope of a view
- Moving work from a development stream to another project
- Moving views
- Access views and VOBs across platform types
- Regenerating the **.view.dat** file
- Regenerating **.ccase_svreg**

7.1 Adding Files and Directories to Source Control

You can add files or directories to source control at any time during the development cycle. To add view-private files and directories to source control or to make placeholders for nonexisting files and directories:

1. Go to your development view and set an activity.
2. Change to the parent directory under which you want to add files and directories to source control.

   For snapshot views, the path from which you add to source control does not need to be loaded. However, it must match the VOB namespace.

3. Check out the parent directory element by entering **cleartool checkout –nc** *directory-name*. We suggest using the **–nc** option because ClearCase appends appropriate comments when you modify directory elements.

4. Do one of the following:

   - To add a directory to source control, enter this command:

     cleartool mkdir *directory-name*

   - To add a file to source control, enter this command:

     cleartool mkelem *file-name*

   - To make placeholders for nonexistent objects, enter one of these commands:

     cleartool mkdir *directory-element-pathname*

     cleartool mkelem *file-element-pathname*

By default, when you add an element, it remains checked out. When you finish modifying the new elements, check them in. Elements you add to a directory element are visible only in your view until you check in the directory.

For more information on the **mkelem** command, see *Under the Hood: What Happens When You Add a File or Directory to Source Control* and the **mkelem** reference page in *ClearCase Reference Manual*.

Under the Hood: What Happens When You Add a File or Directory to Source Control

The **mkelem** command always creates an element and initializes its version tree by creating a single branch (named **main**) and a single, empty version (version 0) on that branch. The following arguments for the **mkelem** command determine optional ClearCase behavior:

- Using **mkelem** with no arguments checks out the element. Any view-private data that corresponds to the element pathname remains in your view only and is added to version 1 in the VOB when you check in (Figure 27).

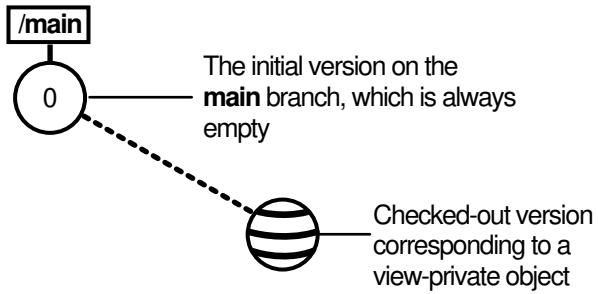
- Using **mkelem –ci** checks in the element, using any existing view-private data that corresponds to the element pathname as the content for version 1. Your view's config spec determines the branch on which ClearCase creates version 1.

- Using **mkelem –nco** suppresses automatic checkout; **mkelem** creates the new element, along with the **main** branch and version**/main/0**, but does not check it out. If *element-pathname* exists, it is moved aside to a **.keep** file.
- (Replicated VOBs only) Using **mkelem –master** assigns to your *current replica* mastership of all branches created during element creation. You will be able to create new versions on the branches.

  Using **mkelem** without the **–master** option assigns mastership of a new branch to the VOB replica that masters the associated branch type. If this replica is not your current replica, you cannot create new versions on the branch.

Other views do not see the element until you check in the element's parent directories and check in the file or directory.

Figure 27 Creating an Element



Importing Files

If you're adding a large number of files and directories to source control, use the **clearexport_ffile** command (or other **clearexport** commands) and **clearimport** command. For more information, see the **clearexport_ffile** and **clearimport** reference pages in *ClearCase Reference Manual*.

7.2 Moving, Removing, and Renaming Elements

This section explains how to move, remove, and rename elements.

Moving and Removing Elements

Because directories as well as files are under ClearCase control, you can move or remove elements from specific versions of directories without affecting the element itself. Moving or removing elements creates new versions of the parent directories to record the modifications.

For example, version 4 of **/gui_vob/design** contains an element named **foo.c**. If you remove **foo.c** from the **design** directory, ClearCase creates version 5 of **/gui_vob/design**, which does not contain the **foo.c** file element. The element **foo.c** itself is not modified.

```
% cd pat_v1.4_cropcircle/gui_vob
% cleartool ls design@@/main/4
  foo.c@@/main/2
  bar.c@@/main/10
% cleartool checkout –nc design
   Checked out "design" version "/main/4"
% cleartool rmname foo.c
  Removed "foo.c"
% cleartool checkin –nc design
  Checked in "design" version "/main/5"
% cleartool ls design@@/main/5
  bar.c@@/main/10
% cleartool ls design@@/main/4
  foo.c@@/main/2
  bar.c@@/main/10
```

Before you move or remove an element name from a directory, verify with your project manager that your changes will not adversely affect other team members or break projectwide builds.

To Move an Element Within a VOB

1. Check out the parent directory and the destination directory.
2. Enter the following command:

   cleartool mv *element-name destination-directory*

3. Check in the new parent directory and the source directory.

To Move an Element to Another VOB

Use the **cleartool relocate** command.

WARNING: The **relocate** command makes irreversible changes to at least two VOBs and their event histories. We recommend that you not use it for minor adjustments. Furthermore, you are advised to stop VOB update activity before and during a relocate operation. Check with your project manager and ClearCase administrator before using the **relocate** command.

To Remove an Element Name from a Directory

1. Check out the parent directory.
2. Enter the following command:

   cleartool rmname *element-name*

3. Check in the parent directory.

Other Methods for Removing Elements

Removing an element from its parent directory does not affect the element itself, but two other types of a removal operation do irrevocably affect an element, and we recommend that you be very conservative in using these operations:

- Removing a version from an element's version tree. For more information, see the **rmver** reference page in *ClearCase Reference Manual*.
- Removing an element from a VOB. For more information, see the **rmelem** reference page in *ClearCase Reference Manual*.

Renaming Elements

Renaming an element creates a new version of the parent directory to catalog the new element name. The element uses its new name in subsequent versions of its parent directory, but previous versions of the parent directory refer to the element by its previous name.

```
% cd pat_v1.4_cropcircle/gui_vob
% cleartool ls design@@/main/4
  foo.c@@/main/2
  bar.c@@/main/10
% cleartool checkout –nc design
   Checked out "design" version "/main/4"
% cleartool mv foo.c  bas.c
  Moved "foo.c" to "bas.c"
% cleartool checkin design
  Default:
  Added file element "bas.c".
Removed file element "foo.c".
Checkin comments for ".":  ("." to accept default)
.
  Checked in "design" version "/main/5"
% cleartool ls design@@/main/5
  bas.c@@/main/2
  bar.c@@/main/10
% cleartool ls design@@/main/4
  foo.c@@/main/2
  bar.c@@/main/10
```

Before you move or remove an element name from a directory, verify with your project manager that your changes will not adversely affect other team members or break project builds.

To Rename an Element

1. Check out the parent directory.
2. Enter the following command:

   cleartool mv *pname target-pname*

3. Check in the parent directory.

7.3 Accessing Elements Not Loaded into a Snapshot View

While working with source files in a snapshot view, you may need to see the contents of elements that are not loaded into the view or see ClearCase information about these nonloaded elements. For example, you may have chosen not to load a VOB that contains functional-specification documents. However, you might want to check periodically whether the functional specifications have been modified by reviewing the element's ClearCase history.

Listing All Elements in the VOB Namespace

You can use the **cleartool ls** command to see all elements in the VOB namespace, even if they are not loaded into your snapshot view. This command lists the names of elements cataloged in the VOB namespace that your view's config spec selects. The output of **cleartool ls** includes this information:

- The version ID of the particular version the view selects
- The version-selection rule in the config spec that selects this version

To see all elements in a directory, enter this command:

cleartool ls *pathname...*

For more information, see the **ls** reference page in *ClearCase Reference Manual*.

To access a version of a file not loaded into your view, use the **cleartool get** command, which copies the version you specify into your view.You can view nonloaded files or copy them into your view for build purposes, but you cannot check them out. Only file elements that are loaded into the view can be checked out.

NOTE: You cannot use **cleartool get** for directory elements.

To copy a nonloaded version of a file element into your view, type this command:

cleartool get –to *filename version-extended-pathname*

For example, **get –to foo.c.previous.version foo.c@@/main/v3.1_fix/10** copies **foo.c@@/main/v3.1_fix/10** into your view under the name of **foo.c.previous.version**.

7.4 Adjusting the Scope of a View

At any time during a development cycle, you may need to change the set of elements available to your view. For example, if you work in a snapshot view, you can reduce the amount of time required for deliver and rebase operations by narrowing your view's scope to include only the set of elements related to your activities. (The fewer elements in your snapshot view, the less time needed to update it.) Note, however, that to test your work during the deliver operation, your view must include any files required to satisfy build dependencies.

To adjust the scope of a view, you can do the following:

- Change which elements are loaded into a snapshot view
- Mount or unmount VOBs for dynamic views

Adjusting the scope of a view does not affect the view's stream or any other views that are attached to the stream. For example, if a recommended baseline includes a new set of files and directories, rebasing your development stream makes the new files and directories available to your stream, but they may not be visible in your development view, depending on the view's scope.

To Change Which Elements Are Loaded into a Snapshot View

A set of load rules determine which elements ClearCase copies into the snapshot view. You can add or modify load rules in any of the following ways:

- When editing the view's config spec.

  A config spec is a set of rules, written and maintained by the view's stream, that determine which versions of elements are in the view. You can modify some parts, such as the load rules, but other parts of the config spec, which are clearly marked with comment tags, must be modified only by the stream. Any time you or the stream modify a snapshot view's config spec, ClearCase updates the entire view. This is an appropriate course of action when you deliver or rebase the view's stream, or when you remove elements by changing load rules; but it may be cumbersome if you want only to add a few elements to the view's scope.

- By using **update –add_loadrules**.

  The **–add_loadrules** option of **cleartool update** adds load rules to your view's config spec, but updates only the portion of the view that is affected by the new load rules.

To Add or Modify Load Rules When Editing the Config Spec

1. Open the view's config spec for editing, as follows:

   a. Open a shell and change to a directory in the view.

   b. Enter the following command:

      cleartool edcs

      ClearCase opens the view's config spec in your default text editor.

   In your text editor, your view's load rules are below the line `ADD CUSTOM LOAD RULES AFTER THIS LINE`. By default, your view uses one load rule for each component loaded in the view:

   load *name-of-component*

2. Modify the existing load rules or create new ones, using the following syntax:

   load *component-name* [*pathname* ...]

   For example, the rule **load /guivob** loads the **/guivob** component and all the files and directories it catalogs. To limit the scope of this load rule to a subtree within the component, add the subtree's pathname. For example, the rule **load /guivob/batch/calc** loads the subtree starting from the **calc** directory. You can use a load rule to specify a single file. For example, the rule **load /guivob/make/include.h** loads only the file **include.h** from the **make** directory.

   Load rules with a broader scope override rules with a narrower scope. For example, if your config spec contained **load /guivob** and **load /guivob/batch/calc**, ClearCase loads the entire **/guivob** component.

   ClearCase imposes no practical limit on the number of load rules you can use for any one view.

3. Save the config spec and exit the text editor.

4. In your shell, answer **Yes** to the ClearCase prompt for setting the config spec.

To Add Load Rules with update –add_loadrules

1. Open a shell and change to a directory in the view.

2. Enter the following command:

   cleartool update –add_loadrules *element-pathname* [...]

element-pname names an element in the component namespace at a pathname that is relative to a snapshot view. For example, the following command loads all elements in a component named **/guivob** into the view **pat_v1.4_cropcircle_sv**:

cleartool update –add_loadrules ~/pat_v1.4_cropcircle_sv/guivob

You can also use a relative pathname for the *element-pathname* argument. For example, these commands load all elements in **guivob**:

```
% cd ~/pat_v1.4_cropcircle_sv
% cleartool update –add_loadrules guivob
```

These commands loads only the **batch** directory recursively:

```
% cd ~/pat_v1.4_cropcircle_sv
% cleartool update –add_loadrules guivob/batch
```

Mounting or Unmounting VOBs for Dynamic Views

Mounting a VOB makes its files and directories available to your dynamic views. Unmounting a VOB frees your workstation's mount points.

To Mount VOBs

Type this command:

cleartool mount *VOB-tag*

Usually, ClearCase mounts VOBs that were created with a public VOB-tag when you start or reboot your workstation. If public VOBs do not mount, type **cleartool mount –all** to mount them.

VOBs remain mounted until you reboot your workstation or unmount them with the **cleartool umount** command. For more information on mounting VOBs, see the **mount** reference page in *ClearCase Reference Manual*.

To Unmount VOBs

From a shell, enter this command:

cleartool umount *VOB-tag*

For more information on unmounting VOBs, see the **umount** reference page in *ClearCase Reference Manual* or enter **cleartool man umount** in a shell.

7.5 Moving Work from a Development Stream to Another Project

Under normal circumstances, project managers use baselines (which reside on a project's integration stream) to share work between projects. But if you start work on an activity in your development stream and then determine that you cannot complete the work for the current project, we recommend that you move the work to the project in which you intend to complete it. To facilitate the merge process, move the work to the new project as soon as you determine that you cannot complete it in the original project. (The longer you wait to merge the work, the less you may remember about it and the greater the potential for merge conflicts.)

We recommend against working in a development stream that contains activities you do not intend to complete for the current project because it may prevent the rebase operation from synchronizing your development stream with the work in the baseline.

To Move Work from a Development Stream to Another Project

1. Get the activity's *activity-selector* by entering the following command in your development view:

   % **cleartool lsactivity –long –cview**

2. Join the project in which you intend to complete the work. (See Chapter 2, *Setting Up Work Areas*.)

3. Decide whether to add the work to the new project's integration stream or to your development stream in the new project.

4. From your view in the new project, create an activity to identify the work you are moving from the original project. Then set the view to the activity. (See *Creating Activities* on page 43 and *Setting Activities* on page 46.)

5. From your view in the new project, use the **cleartool findmerge** command with the following syntax:

   % **cleartool findmerge** *activity-selector* **–fcsets –merge –gmerge**

   Supply the activity-selector for the activity from which you want to move versions, that is, the activity-selector from Step #1.

   For example:

   % **cleartool findmerge change_copyright_notice@/projects/cropcircle**
   –fcsets fl–merge –gmerge

6. If ClearCase encounters merge conflicts, it starts Diff Merge. In Diff Merge, resolve the conflict, save your merge results, and exit.

7. In your development stream for the original project, complete work for and deliver all other activities.

8. Discontinue work in your original development stream and create another one for the original project. If your organization's policies require you to work in only one development stream per project, consult your project manager.

   It is possible, but not recommended, to remove the versions in the original activity (using **cleartool rmver –hlink**) and then to remove the activity using **cleartool rmactivity**. However, removing versions effectively erases your organization's development history and should be done only when necessary and only when you and your project manager are certain that removing the work does not affect other work. By removing the versions and *merge arrows* (with the **–hlink** option), ClearCase has no record of the merge (although the element's history remains intact).

7.6 Moving Views

This section discusses the following tasks:

- Changing the physical location of a snapshot view's directory tree
- Moving a view storage directory

For information on changing a view-tag, see the **mktag** reference page in *ClearCase Reference Manual*.

Changing the Physical Location of a Snapshot View

If the snapshot view storage directory is in a storage location, you can use the standard **mv** command to move the snapshot view's directory tree of loaded elements and view-private files. You can move the view to a different workstation, but the workstation must run a UNIX operating system.

CAUTION: If the view storage directory is located below the root directory of the view, **do not use** the standard **mv** command to move the snapshot view. Instead, see *Moving a View Storage Directory*.

To Find the Location of the View Storage Directory

Enter the following command:

cleartool lsview –long *view-tag*

The Global Path field displays the pathname for the view storage directory.

Update After Moving

After moving a snapshot view, you must use **cleartool update** (or **cleartool update –print**) to modify **.ccase_svreg** in your home directory. Some ClearCase operations use information from this file and will not succeed until you use **update** to modify it.

Moving a View Storage Directory

Each dynamic view and snapshot view includes a view storage directory, which ClearCase uses to maintain the view. **Do not use** the standard **mv** command to move a view storage directory for the following reasons:

- The view storage directory includes a database. Moving the database without first shutting down the view's **view_server** process can corrupt the database.
- ClearCase stores the location of view storage directories in its own set of registries. The information in these registries must be correct for you to perform ClearCase operations in your views. In a dynamic view, the location in ClearCase registries must be correct for you to access any file or directory in the view.

We suggest that you ask your ClearCase administrator to move view storage directories because it may affect other, potentially many other, ClearCase users at your site. *Administering ClearCase* describes the procedure for moving view storage directories.

CAUTION: You will lose data (including view-private files in a dynamic view) if you move a view storage directory without following the procedure described in *Administering ClearCase*.

7.7 Accessing Views and VOBs Across Platform Types

ClearCase supports environments in which some ClearCase hosts use a Microsoft Windows operating system and others use a UNIX operating system.

This section discusses the following topics:

- Creating views across platform types
- Accessing VOBs across platform types
- Developing software across platform types

Creating Views Across Platform Types

Your administrator can set up storage locations on Windows and UNIX server hosts. Any snapshot view that you create can use one of these storage locations, regardless of the platform type of the server host. For more information about storage locations, see the **mkstgloc** reference page in *ClearCase Reference Manual*.

For a dynamic view, the view storage directory must be located on a host of the same platform type as the host from which you create the view: if you create a dynamic view from a UNIX host, you must locate the view storage directory on a ClearCase host on UNIX; if you create a dynamic view from a Windows host, you must locate the view storage directory on a Windows NT host that is set up to store view storage directories. We recommend that you locate dynamic view storage directories on the host from which you most often use the view.

Snapshot View Characteristics and Operating-System Type

For snapshot views, the operating system type from which you create the view determines view characteristics; the operating system type that hosts the files and processes related to a snapshot view do not affect the view's behavior.

For example, it is possible to create a snapshot view from a Windows host and locate the view directory tree and the view storage directory on a ClearCase host on UNIX (assuming that you use third-party software to access UNIX file systems from Windows computers). Even though all files related to the view are on a UNIX workstation, because you created the view from a Windows host, the view behaves as if its files are located on a Windows computer: it does not create symbolic links if the load rules encounter a VOB symbolic link, and you can issue ClearCase commands for the view only from Windows hosts (ClearCase hosts on UNIX will not recognize the directory tree as a snapshot view).

Accessing Views Across Platform Types

Accessing UNIX Snapshot Views from Windows Hosts

ClearCase supports a set of third-party products for accessing UNIX file systems from Windows computers. If your organization uses one of these products, you can access UNIX snapshot views from Windows Explorer (or a command prompt) just as you would access any other directory tree on a UNIX workstation.

You can access snapshot views across platforms, but you cannot issue ClearCase commands across platforms. For example, you cannot check out files in UNIX snapshot views from Windows hosts nor can you create shortcuts to UNIX snapshot views from ClearCase Explorer.

If, from a Windows host, you hijack a file in a UNIX snapshot view, ClearCase detects the hijack when you update the view from a ClearCase host on UNIX.

Accessing Windows Snapshot Views from UNIX Hosts

ClearCase does not support accessing Windows file systems from UNIX workstations.

Accessing UNIX Dynamic Views from Windows Hosts

ClearCase supports a set of third-party products for accessing UNIX file systems from Windows computers. If your organization uses one of these products, you can complete the following tasks to access UNIX dynamic views from Windows computers:

1. Create the UNIX view with the proper text mode. For more information, see *Developing Software Across Platform Types*.
2. Import the UNIX view's view-tag into your Windows network region.
3. Start the dynamic view.

Accessing Windows Dynamic Views from UNIX Hosts

ClearCase does not support products for accessing Windows file systems from UNIX workstations. You cannot access Windows views from UNIX hosts.

Accessing VOBs Across Platform Types

Your administrator sets up VOBs on Windows or UNIX hosts and creates *VOB-tags* in each ClearCase network region that needs to access the VOBs. (See *Administering ClearCase* for information on registering UNIX VOB-tags in a Windows network region.) Then, from any ClearCase host on Windows or UNIX you can create snapshot views to load elements from VOBs that have tags in your network region.

From a ClearCase host on Windows that supports dynamic views, you can access VOBs on Windows and UNIX from dynamic views as well as snapshot views. To access VOBs on UNIX from Windows dynamic views, you must use third-party software that provides access to UNIX file systems from Windows computers. From a ClearCase host on UNIX, you cannot access VOBs on Windows from dynamic views. Table 4 summarizes your options for accessing VOBs across platform types.

Table 4 Accessing ClearCase VOBs Across Platform Types

Platform Type of Your ClearCase Host	Platform Type on Which VOB Is Located	View from Which You Can Access Source Files
Windows computer	Windows computer or UNIX workstation	Snapshot view or dynamic view
UNIX workstation	Windows computer	Snapshot view
UNIX workstation	UNIX workstation	Snapshot view or dynamic view

Developing Software Across Platform Types

If developers check in source files from views created on both Windows and UNIX hosts, consider creating your views in insert_cr or strip_cr text mode. The text modes change how a view manages line terminator sequences. For more information about view text modes, refer to *Administering ClearCase* or ClearCase online help.

7.8 Regenerating the .view.dat File

The root directory of a snapshot view contains a hidden file, **.view.dat**. If you delete this file inadvertently, ClearCase no longer identifies the view as a ClearCase object, and you can no longer perform ClearCase operations on files or directories loaded in the view.

To Regenerate the .view.dat File

1. Open a command shell.
2. Type this command:

   % **Perl** *ccase-home-dir*/**etc/utils/regen_view_dot_dat.pl** \
   [**–tag** *snapshot-view-tag*] *snapshot-view-pathname*

   For example:

   % **Perl /usr/atria/etc/utils regen_view_dot_dat.pl** \
   –tag pat_v1.4_cropcircle_sv \
   ~/pat_v1.4_cropcircle_sv

   If the view storage directory is under the root directory of the view, you do not need to use the **–tag** *snapshot-view-tag* argument.

7.9 Regenerating the .ccase_svreg file

When you create a snapshot view, ClearCase creates or modifies the file **.ccase_svreg** in your home directory. Some ClearCase operations use information from this file.

If you inadvertently delete or corrupt this file, you must regenerate information in **.ccase_svreg** for each snapshot view that you use. To do so, update the view with either of the following commands:

- **cleartool update**
- **cleartool update –print**

Working in a Snapshot View While Disconnected from the Network

A

If you need to work with your source files from a computer that is disconnected from the network of ClearCase hosts and servers, you can set up a snapshot view for disconnected use.

This chapter describes the following tasks:

- Setting up a view for your hardware configuration
- Preparing the view
- Disconnecting the view
- Working in the view
- Re-connecting to the network
- Using the Update Tool

NOTE: While disconnected from the network, you cannot access ClearCase information about the files in your view or issue most ClearCase commands. If you want to work from a remote location and continue to access ClearCase information and issue ClearCase commands, consider using the ClearCase Web interface. Talk to your ClearCase administrator to see whether the ClearCase Web interface has been configured at your site and what URL you need to supply to your Web browser to access it. For information about using the Web interface, see the Web interface online help.

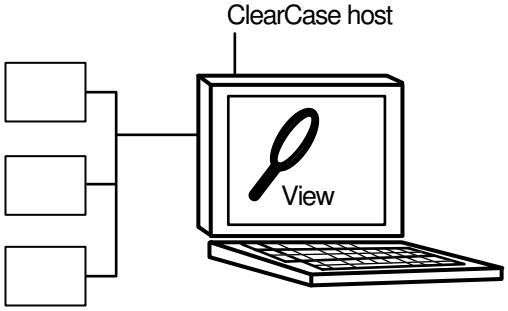
A.1 Setting Up a View for Your Hardware Configuration

You can use one of several hardware configurations to work in a snapshot view that is disconnected from the network.

This chapter describes the following recommended configurations:

- Creating and using the view on a laptop computer that periodically connects to the network (Figure 28).

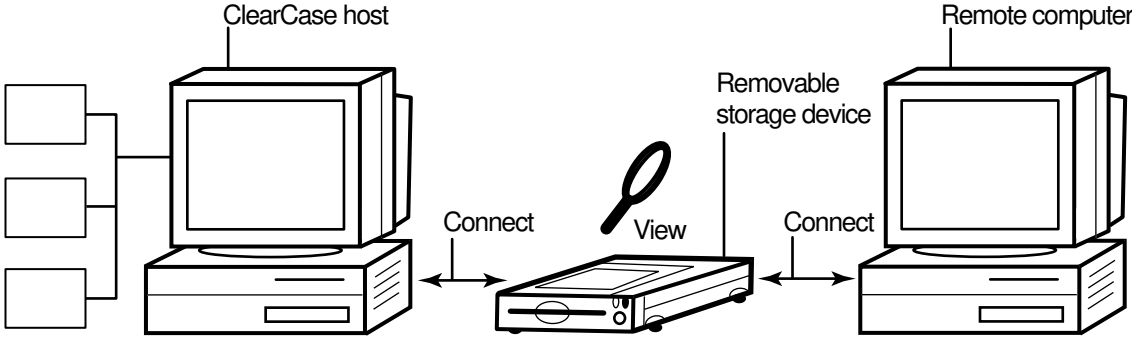
Figure 28 View on a Laptop



NOTE: The laptop computer must run a UNIX operating system.

- Creating and using the view on a removable storage device such as an external hard drive or some other device (such as a Jaz™ drive) that provides satisfactory read/write performance (Figure 29).

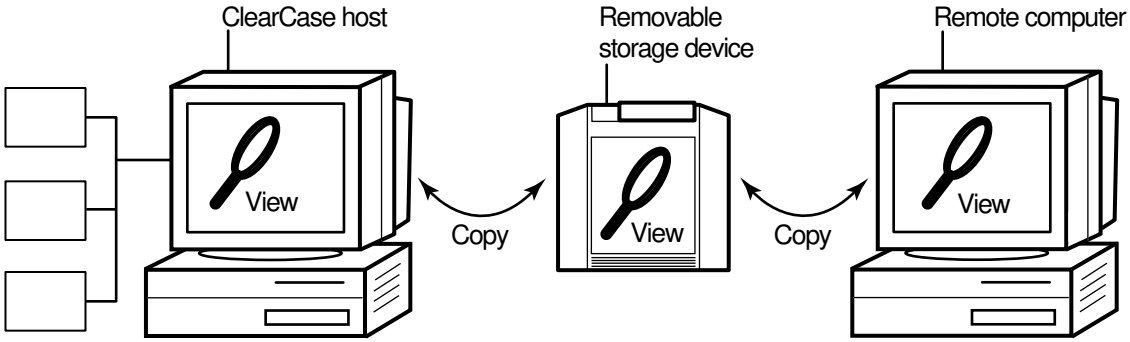
Figure 29 View On a Removable Storage Device



NOTE: The remote computer must run a UNIX operating system.

- Copying the view from a ClearCase host to a temporary, removable storage device such as a diskette or a tape drive, which usually does not provide satisfactory read/write performance, and then copying the view from the storage device to a computer that is disconnected from the network (Figure 30).

Figure 30 Copy the View



NOTE: The remote computer must run a UNIX operating system.

Under the Hood: Location of the View Storage Directory in Disconnected-Use Configurations

In all the configurations recommended for disconnected use, the snapshot view storage directory is in a server storage location. We recommend this configuration because a view's **view_server** process runs on the host that contains the view storage directory. A **view_server** is a long-lived process that manages activity for a specific view. If the view storage directory is in the root directory of the snapshot view and you disconnect the view from the network while the **view_server** process is writing to the storage directory, you can corrupt the data ClearCase uses to maintain your view.

A.2 Preparing the View

Before you disconnect the view from the network, complete these tasks:

- Update the view to establish a checkpoint. (For information on updating the view, see *Updating the View* on page 122.)
- Check out the files you expect to modify. After you're disconnected from the network, you cannot check out files, although there are workarounds. (See *Hijacking a File* on page 117.)

When you are no longer connected to the network you cannot use most ClearCase commands. At this point, the disconnected computer does not distinguish a snapshot view directory from any other directory in the file system.

A.3 Disconnecting the View

If the view is located on a laptop or removable storage device, disconnect the device from the network; reconnect the removable media to a remote computer.

If you do not have a storage device with satisfactory read/write performance, use a standard UNIX copy command to copy files from your view to the storage media and from the storage media to the remote computer. To prevent ClearCase from identifying copied files as *hijacked*, use copy command options to preserve file times. For example:

```
% cp –Rp
```

A.4 Working in the View

You cannot use most ClearCase commands when disconnected from the network. Yet you may need to work on files that you did not check out or locate files you have modified. This section provides workarounds for these ClearCase operations.

Hijacking a File

If you need to modify a loaded file element that you have not checked out, you can *hijack* the file. ClearCase considers a file hijacked when you modify it without checking it out. For more information, see *Under the Hood: How ClearCase Determines Whether a File is Hijacked* on page 122.

When you reconnect to the network, you use the Update Tool to find the files you hijacked. You can do the following with a hijacked file:

- Check out the file. You can then continue to modify it and, when you're ready, check in your changes.
- Undo the hijack. For more information, see *Undoing a Hijack* on page 121.

To Hijack a File

Use **chmod** to add the **write** permission and then modify the file. For example:

```
% chmod +w foo.c
```

Finding Modified Files While Disconnected

To find all files that have been modified within a specified number of days, use the following command:

find *snapshot-view-pathname* **–mtime –***number-of-days* **–ls –type f**

For example, to find all files modified within the last two days, enter this command:

```
% find ~/pat_v1.4_cropcircle –mtime -2 –ls –type f
```

For more information, see the **find** manpage.

A.5 Connecting to the Network

If the view is located on a laptop or removable storage device, connect the device to the LAN and make sure that the view is accessible to the host on which the view storage directory is located.

If you copied the view onto removable media, use a standard UNIX copy command to copy files back to the original location on the network computer.

A.6 Using the Update Tool

When you're connected to the network, use the Update Tool for the following tasks:

- Determine how to handle hijacked files
- Update the view

Determining How to Handle Hijacked Files

Handling hijacked files involves the following tasks:

- Finding hijacked files
- Comparing a hijacked file to the version in the VOB
- Checking out a hijacked file
- Undoing a hijack
- Choosing other ways to handle hijacked files

To Find Hijacked Files

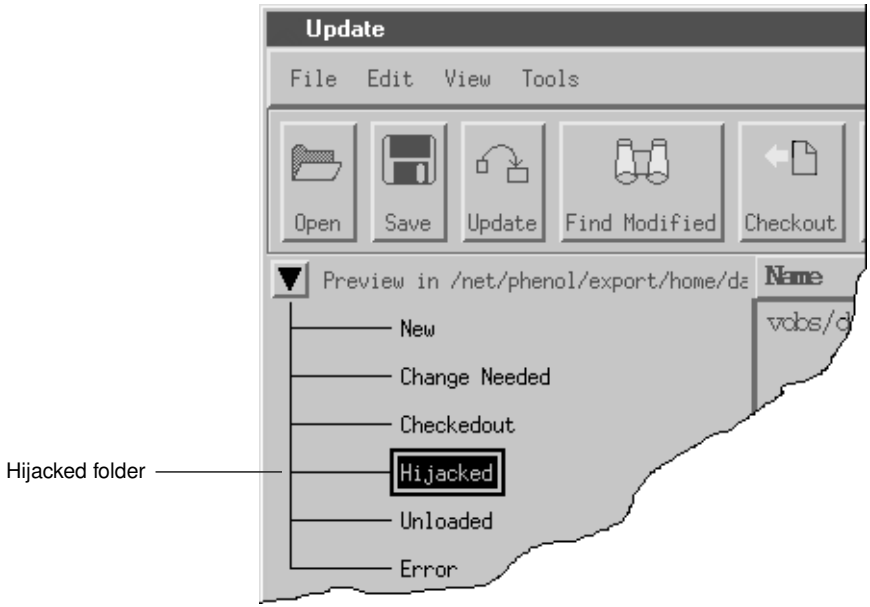
1. Enter the following command:

   cleartool update –graphical *snapshot-view-pathname*

2. In the **Update** dialog box, click **Preview only**. Then click **OK**.

3. If any hijacked files are in your view, the ClearCase Snapshot View Update window displays a folder in the left pane titled **Hijacked** (Figure 31).

Figure 31 Hijacked Files in the Update Window



To Compare a Hijacked File to the Version in the VOB

You can use the Diff Merge tool to see how the hijacked file differs from the checked-in version of the file:

1. In the right pane of the ClearCase Snapshot View Update window, click a hijacked file.

2. Click **Tools>Compare with old**. For information on using the Diff Merge tool, see the online help.

Checking Out a Hijacked File

To keep the modifications in a hijacked file, check out the file:

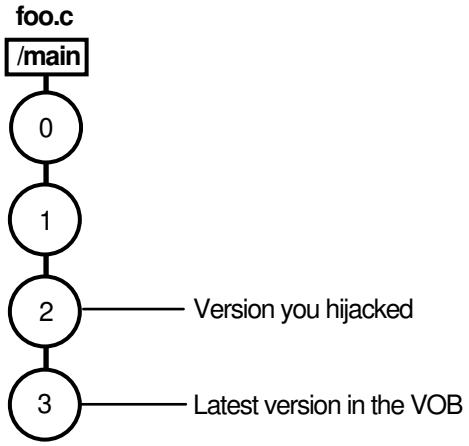
1. In the right pane of the ClearCase Snapshot View Update window, click a hijacked file.
2. Click **Tools>Checkout**.
3. ClearCase treats a checked-out hijacked file as it does any other checkout.

When you're ready, you can check in the file or, if necessary, merge your changes with a version in the VOB.

You May Be Prompted to Merge

If you're working with a shared set of versions and someone has checked in a newer version of the file while it was hijacked in your view (Figure 32), you have to merge the hijacked file with the newer version in the VOB at checkout.

Figure 32 Hijacked Version May Not Be the Latest Version



To Merge with the Latest Version

To merge the latest version in the VOB to the version in your view, enter the following command:

cleartool merge –graphical –to *file-or-directory-in-your-view* \
*file-or-directory-name***@@/main/LATEST**

NOTE: @@/main/LATEST is a *version-extended pathname*. For more information, refer to the **pathnames_ccase** reference page in *ClearCase Reference Manual*.

For example:

```
% cleartool merge –graphical –to foo.c   foo.c@@/main/LATEST
```

Using the **–graphical** option starts the Diff Merge tool. For information on using the Diff Merge tool, refer to ClearCase online help. After merging, save the results and check in the version by entering the **cleartool checkin** command from the view.

Undoing a Hijack

If, for specific hijacked files, you want to discard your changes and get a fresh copy of the version from the VOB, you can undo the hijack.

1. In the right pane of the ClearCase Snapshot View Update window, select one or more hijacked files.
2. Click the selected files, and click **Tools>Undo hijacked file**.

ClearCase overwrites the hijacked file with the version that was loaded in the view. If you want to overwrite hijacked files with the versions the config spec selects in the VOB, refer to Step #2 in *Updating the View* on page 122.

Under the Hood: How ClearCase Determines Whether a File is Hijacked

To keep track of file modifications in a snapshot view, ClearCase stores a loaded file's size and last-modified time stamp (as reported by the UNIX file system). ClearCase updates these values each time you check out a file, check in a file, or load a new version into the view.

To determine whether a file is hijacked, ClearCase compares the current size and last-modified time stamp of a non-checked-out file with the size and time stamp recorded in the view database. If either value is different from the value in the view database, ClearCase considers the file hijacked.

Changing a non-checked-out file's read-only permission alone does not necessarily mean ClearCase considers the file hijacked.

Other Ways to Handle Hijacked Files

While updating the view, you can handle hijacked files in any of the following ways:

- Leave hijacked files in place
- Rename the hijacked files and load the version from the VOB
- Overwrite hijacked files with the version the config selects in the VOB

See *Updating the View* for more information.

Updating the View

1. Enter the following command:

   cleartool update –graphical *snapshot-view-pathname*

2. To configure the Update Tool for handling hijacked files, in the **Update** dialog box click the **Advanced** tab and select a method for handling the remaining hijacked files. You have three choices:

   - Leave hijacked files in place
   - Rename the hijacked files and load the version from the VOB
   - Overwrite hijacked files with the version the config spec selects in the VOB

3. To start the update, click **OK**.

Index

.ccase_svreg file 112
.keep files, canceled checkouts 60

A

activities
- about 2
- assigning in ClearQuest projects 45
- basic tasks 51–69
- changing assignment 67
- ClearCase implementation details 9
- ClearQuest record types and 17
- closing in ClearQuest 65
- creating in ClearQuest 43
- creating, methods for 43–44
- deleting 66, 68
- delivering to project 5, 71, 86
- dependences in deliver operation 76
- duplicates 66
- finding, methods for 38–42
- how to locate and create 37
- measuring progress 64–69
- moving to another development stream 66
- postponing 66
- reopening 66
- setting from development view, methods for 46–49
- type for integration 79
- undelivered change sets 75
- undelivered, in development stream 66
- unsetting from development view 49
- viewing change sets 57
- viewing currently set 50
- working with 3–4

adding files to source control 95–96
***attache-home-dir* directory** vi

B

baselines 7
- *See also* rebase operation

binary files, merging 82
building software
- ClearCase build tools 63

C

***ccase-home-dir* directory** vi
change requests
- state types and transitions 21

change sets
- comparing 56
- undelivered 75
- viewing 57

checking in
- about 62
- activities 72
- effect of on VOB links 54
- procedure 62

checking out
- about 52
- during deliver operation 83
- during rebase operation 92
- for remote use 116
- hijacked files 120
- nonloaded files 101
- when disconnected from network 117

checkouts
- canceling 60
- how cancellation is handled 60

clearexport and clearimport commands 97
ClearQuest integration
- assigning activities 45
- closing activities 65
- creating activities 43
- creating custom queries 39
- deleting activities 69
- logging in 36
- modifying activity information 64
- policies for deliver operation 74
- queries 38
- setting activities 47
- UCM concepts 17
- UCM schemas 17

comparing versions
hijacked files 119
procedures 55
components
about 8
choosing for snapshot view 34
loading into snapshot view 34
conventions, typographical vi
copying
nonloaded versions into views 101
snapshot views to removable storage devices 116
views from removable storage devices 118

D

deliver operation
about 5, 71
activity dependencies 76
binary files 82
concurrent merge algorithm 78
how completed 85
merging activities 77
MultiSite method for 6, 85
preparing work area 72
starting 74
state model 85
stopping and restarting 82
undelivered change sets 75
undoing changes 84
updating views 73
development streams
about 13
listing activities in 40
moving activities to 66
moving work from 105
naming convention 27
renaming 27
undelivered activities 66
development views
access to 35
activity currently set 50
adjusting scope of 102
delivering activities 72
preparing for rebase operation 88
removing versions from 93
setting activities in 46–49
unsetting an activity 49
development work areas
about 2
effect of rebase operation 90
preparing for deliver operation 72
testing after rebase 91
Diff Merge 80
directories
adding to source control 95
canceling checkouts 61
finding checkouts from 57
importing directory trees to source control 97
listing nonloaded files 101
remote use 114
removing element names 99
documentation
online help description vii
dynamic views
accessing files in 35
activating and deactivating VOBs 104
behavior of VOB links 53
build tools available for 83
default view-tag convention 32
mounting and unmounting VOBs 104
view storage directory location 33
when to use 30

E

elements
See also files; versions
checkout procedure 52
history of changes 55
moving and removing 98
nonloaded, accessing 101
renaming 100

F

files
See also elements; versions
accessing 35
adding to existing directory tree 95
adding to source control 95
finding checked-out 57
listing nonloaded 101
VOB link 54

G

get command 101

H

hard links 53
hardware configurations for remote use 114
hidden file 112
hijacked files
- about 117
- checking out 120
- comparing to version on VOB 119
- finding 119
- handling 118
- how determined 122
- merging 120
- undoing hijack 121

History Browser 55

I

importing directories to source control 97
installing ClearCase iv
integration activity 79
integration streams
- about 13
- creating 28
- evolution of 15
- MultiSite issues 85

integration views
- recommended view type for 30
- removing versions from 84
- updating for deliver operation 73
- uses of 83

interoperation on Windows and UNIX 108–110

J

Join Project Wizard, how to start 26
joining projects
- about 25
- creating streams 27
- creating views 29

L

laptops
- configuration for remote use 114

loading files into snapshot views 35
ls command 101
lsactivity command 40

M

merging files
- binary files 82
- during deliver operation 77
- hijacked files 120
- how it works 81
- in rebase operation 91
- resolving differences 80

MultiSite, deliver operation and 6, 85
MyToDoList query 19, 38

O

online help, accessing vii

P

Project Explorer 40
project VOBs 8
projects
- about 8
- choosing components to load 34
- creating activities 43–44
- joining 25
- loading components into view 34
- moving work to other 105
- referencing objects in 28

Q

queries in ClearQuest
- about 18, 38
- customizing 39

R

read/write performance of remote storage devices 114
rebase operation
about 7, 87
effect on development work area 90
how to complete 93
interrupting 91
merging requirements 91
preparing development view 88
starting 89
testing tasks 91
undoing changes 92
relocate command 99
remote deliver operation 6, 85

S

schemas, ClearQuest 17
shortcut menus, deactivated 112
snapshot views
See also updating snapshot views; working from a remote location
access to nonloaded elements 101, 112
accessing files in 35
behavior of and operating system 108
behavior of symbolic links 53
changing elements loaded 102
checking out during deliver operation 84
choosing project components 34
constraints on location 31
copying nonloaded versions to 101
copying to removable storage devices 116
hardware configurations for remote use 114
improving performance 34
loading project components 34
location of storage directory 31
moving 107
rebasing 90
transferring to laptop 116
updating for deliver operation 73
view.dat file 112
when to use 29
storage devices, removable
disconnecting from network 116
performance of views copied to 114
streams
ClearCase implementation details 11
creating for project 27
symbolic links 53

T

technical support vii
to-do list, adding activities 46
typographical conventions vi

U

UCM (Unified Change Management)
about 1
accessing development views 35
basic concepts 7
UCMCustomQuery 20
umask settings
effect of when joining project 26
under the hood
.ccase_svreg file 32
adding files to source control 96
canceling checkouts 60
checking out from snapshot views 84
ClearQuest UCM concepts 17
concurrent merge algorithm 78
hijacked files, how determined 122
integration activities and baselines 79
merging files and directories 81
project object identifiers 28
snapshot view storage directory 31
UCM concepts 7
Update Tool
about 118
detecting hijacked files 117
handling hijacked files 122
updating snapshot views
canceled directory checkout 61
handling hijacked files 122
moving the view 107
remote use of view 116
VOB links 54

V

version IDs, viewing 101
versions
See also elements; files
comparing 55
copying nonloaded into views 101
moving to another activity 59
removing from development view 93
version-selection rules
listing for elements 101

view storage directories
about 31
constraints on location 33
disk space required 31
location for dynamic views 33
location for remote use 115
moving 107
view.dat file
regenerating 112
views
See also development views; dynamic views; integration views; snapshot views
about 11
creating for project 29
creating on Windows and UNIX 108
types of 29
VOB namespace
listing elements in 101
VOBs
activating for dynamic view 36
mounting and unmounting 104
project VOBs in UCM 8

W

Web interfaces 37
workflow in UCM 2
working from a remote location
about 113
hardware configurations 114
removable storage devices 114
updating view 116