Rational Software Corporation®

Rational® ClearCase®

Developing Software

VERSION: 2003.06.00 AND LATER

PART NUMBER: 800-026155-000

UNIX EDITION

Rational®
the software development company

support@rational.com
http://www.rational.com

Preface

Rational ClearCase is a comprehensive configuration management (CM) system that manages multiple variants of evolving software systems and tracks changes. ClearCase maintains a complete version history of all software development elements, 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 that use 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:

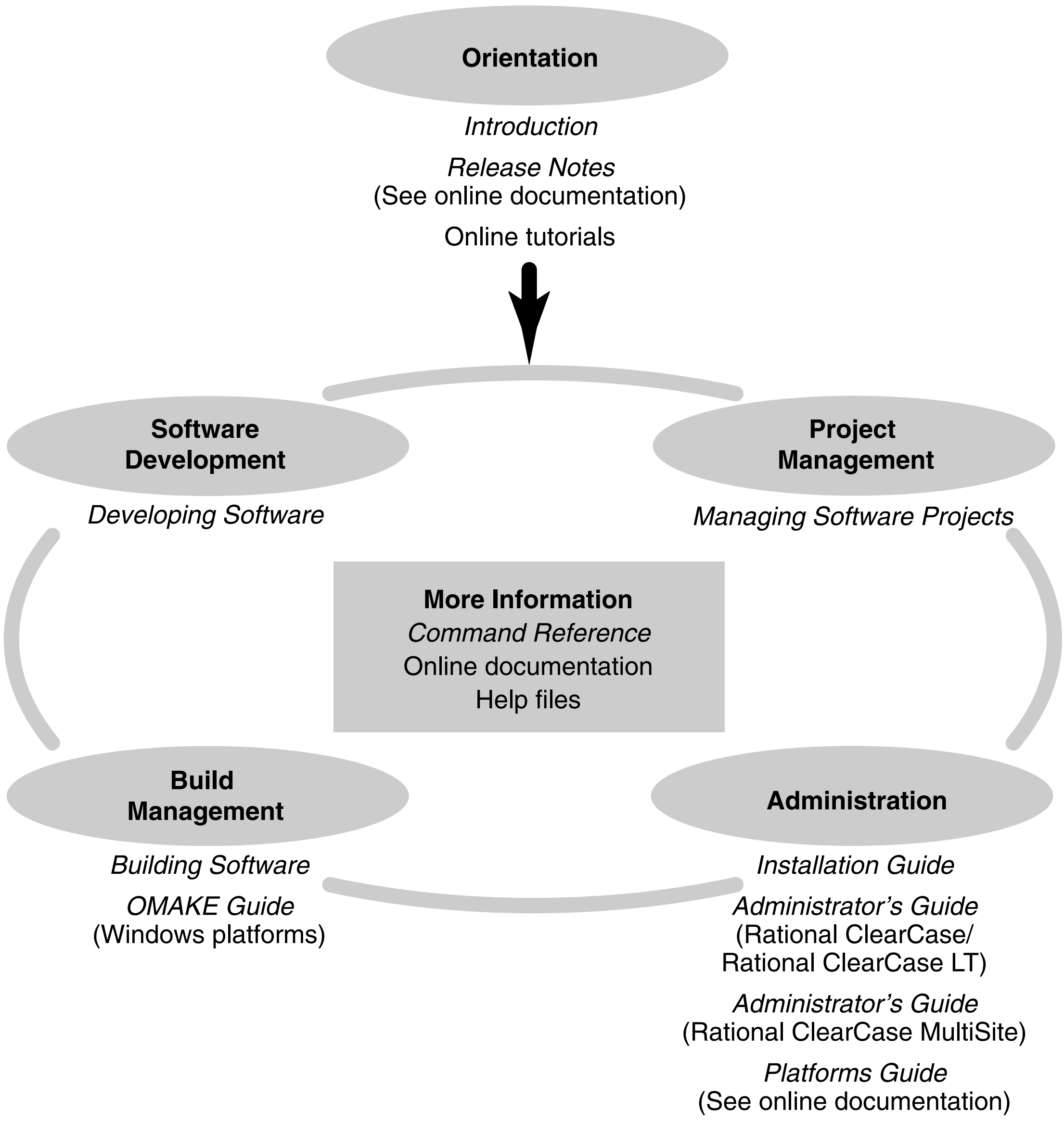
- Skim through this manual to understand at what points in the development cycle you will need to use ClearCase. Consider skipping the sections titled *Under the Hood*, which describe advanced concepts and technical details.
- Read through this manual and follow the procedures.
- As you become familiar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

Assumptions

This manual assumes:

- ClearCase has been installed and configured on your workstation. For information about installing ClearCase, see the *Installation Guide* for the ClearCase Product Family.
- 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*.
- If your development teams uses the UCM-ClearQuest integration, your project manager has set up an ClearQuest user database that your project uses.
- Your organization has established some other development strategy that uses base ClearCase as the configuration-management system, and you are familiar with this strategy.

ClearCase Documentation Roadmap



ClearCase Integrations with Other Rational Products

Integration	Description	Where it is documented
Base ClearCase-ClearQuest	Associates change requests with versions of ClearCase elements.	ClearCase: *Developing Software* ClearCase: *Managing Software Projects* ClearQuest: *Administrator's Guide*
Base ClearCase-Apex	Allows Apex developers to store files in ClearCase.	*Installing Rational Apex* (UNIX)
Base ClearCase-ClearDDTS	Associates change requests with versions of ClearCase elements.	*ClearCase ClearDDTS Integration*
Base ClearCase-PurifyPlus	Allows developers to invoke ClearCase from PurifyPlus.	PurifyPlus Help
Base ClearCase-RequisitePro	Archives RequisitePro projects in ClearCase.	*RequisitePro User's Guide* RequisitePro Help
Base ClearCase-Rose	Stores Rose models in ClearCase.	Rose Help
Base ClearCase-Rose RealTime	Stores Rose RealTime models in ClearCase.	*Rose RealTime Toolset Guide* *Rose RealTime Guide to Team Development*
Base ClearCase-SoDA	Collects information from ClearCase and presents it in various report formats.	*Using Rational SoDA for Word* *Using Rational SoDA for Frame* SoDA Help
Base ClearCase-XDE	Stores XDE models in ClearCase	XDE Help
UCM-ClearQuest	Links UCM activities to ClearQuest records.	ClearCase: *Developing Software* ClearCase: *Managing Software Projects* ClearQuest: *Administrator's Guide*
UCM-PurifyPlus	Allows developers to invoke ClearCase from PurifyPlus.	PurifyPlus Help

Integration	Description	Where it is documented
UCM-RequisitePro	Allows RequisitePro administrators to create baselines of RequisitePro projects in UCM, and to create RequisitePro projects from baselines.	*RequisitePro User's Guide* RequisitePro Help *Using UCM with Rational Suite*
UCM-Rose	Stores Rose models in ClearCase.	Rose Help *Using UCM with Rational Suite*
UCM-Rose RealTime	Associates activities with revisions.	*Rose RealTime Toolset Guide* *Rose RealTime Guide to Team Development*
UCM-SoDA	Collects information from ClearCase and presents it in various report formats.	*Using Rational SoDA for Word* *Using Rational SoDA for Frame* SoDA Help
UCM-TestManager	Stores test assets in ClearCase.	*Rational TestManager User's Guide* TestManager Help *Using UCM with Rational Suite*
UCM-XDE	Stores XDE models in ClearCase	XDE Help
UCM-XDE Tester	Stores XDE Tester Datastores in ClearCase	XDE Tester Help

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 /opt/rational/clearcase on UNIX and C:\Program Files\Rational\ClearCase on Windows.
- *cquest-home-dir* represents the directory into which Rational ClearQuest has been installed. By default, this directory is /opt/rational/clearquest on UNIX and C:\Program Files\Rational\ClearQuest on Windows.
- **Bold** is used for names the user can enter; for example, command names and branch names.
- A sans-serif font is used for file names, directory names, and file extensions.

- **A sans-serif bold font** is used for GUI elements; for example, menu names and names of check boxes.
- *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 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, you can use "**...**" within a pathname as a wildcard, similar to "*" or "?". For more information, see the **wildcards_ccase** reference page.
- If a command or option name has a short form, a "medial dot" (·) character indicates the shortest legal abbreviation. For example:

  lsc·heckout

Online Documentation

The ClearCase Product Family (CPF) includes online documentation, as follows:

Help System: Use the **Help** menu, the **Help** button, or the F1 key. To display the contents of the online documentation set, do one of the following:

- On UNIX, type **cleartool man contents**
- On Windows, click **Start > Programs > Rational Software > Rational ClearCase > Help**
- On either platform, to display contents for Rational ClearCase MultiSite, type **multitool man contents**
- Use the **Help** button in a dialog box to display information about that dialog box or press F1.

Reference Pages: Use the **cleartool man** and **multitool man** commands. For more information, see the **man** reference page.

Command Syntax: Use the **–help** command option or the **cleartool help** command.

Tutorial: Provides a step-by-step tour of important features of the product. To start the tutorial, do one of the following:

- On UNIX, type **cleartool man tutorial**
- On Windows, click **Start** > **Programs** > **Rational Software** > **Rational ClearCase** > **ClearCase Tutorial**

PDF Manuals: Navigate to:

- On UNIX, *ccase-home-dir*/doc/books
- On Windows, *ccase-home-dir*\doc\books

Customer Support

If you have any problems with the software or documentation, please contact Rational Customer Support by telephone, fax, or electronic mail as described below. For information regarding support hours, languages spoken, or other support information, click the **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

Figures

UCM Workflows 1

Unified Change Management (UCM) structures the efforts of your software development team into a defined, repeatable process. This chapter describes Rational ClearCase concepts and the UCM workflows and how the flows affect your work as a developer.

Recommended Reading Paths

Read this chapter first. Then, to start working immediately, use Help and the tutorials 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.

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

Working in UCM refers to tasks performed by both project managers and integrators as well as by developers. For information about tasks performed by other individuals within the UCM workflow, see *Managing Software Projects*.

To Access ClearCase Online Information

To start ClearCase Help, type this command:

cleartool man contents

The help provides links to introductory material. To access the tutorials, type this command:

cleartool man tutorials

For more information, see the *Preface* on page v.

UCM Projects and Workflows

When your work team uses UCM, your project manager creates a UCM object called a project. The project is established using one of the following models.

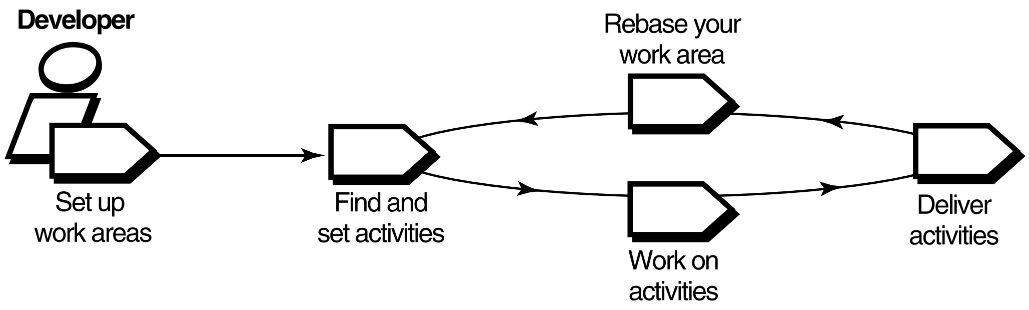
- Multiple-stream—you access source files in a private work area. No one can see your changes. You cannot see others' changes. Then, when you complete your changes, you make your work available to others in a shared work area. Periodically, you have to update your private work area with the latest changes from the shared work area.
- Single-stream—you access source files from a work area that you share with your team members and work on those files in a private area. You can save your changes in your private area so that other team members cannot see them. However, when your changes are ready to be made available, others can see those changes immediately and can access those files to make their own changes.

In multiple-stream development, a relatively large number of individuals works in isolation in their own streams. Team members control when changes of other team members are visible and are responsible for integrating their changes with those of other team members.

In single-stream development, a small number of individuals works relatively closely in a simplified work environment. In this model, isolation is minimal and integration is fairly rapid because team members do not have to perform an explicit integrate action.

The UCM Cyclic Workflow

In multiple-stream project development, your work as a software or information developer follows a cyclic flow.

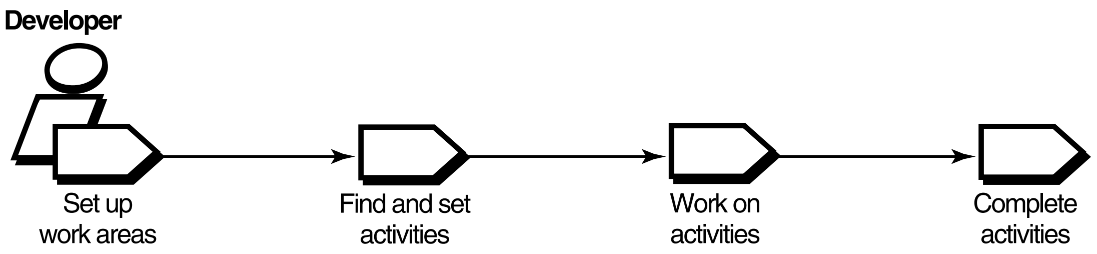


To access your project source files, you set up work areas. As you modify source files, you use activities to organize, identify, and deliver your work.

Regularly, you rebase your work area to take advantage of changes that other members in the project make. You can repeat the steps in this cycle as many times as your project work demands.

The UCM Serial Workflow

In single-stream projects, your work as a software or information developer follows a serial flow.



To access your project source files, you set up work areas. To modify source files, you use activities to organize, identify, and complete your work. When you complete your work, your changes are immediately available to project team members. Unlike in multiple-stream projects, in single-stream projects, there is no delivery or rebasing.

Single-Stream Projects and Rational ClearCase MultiSite

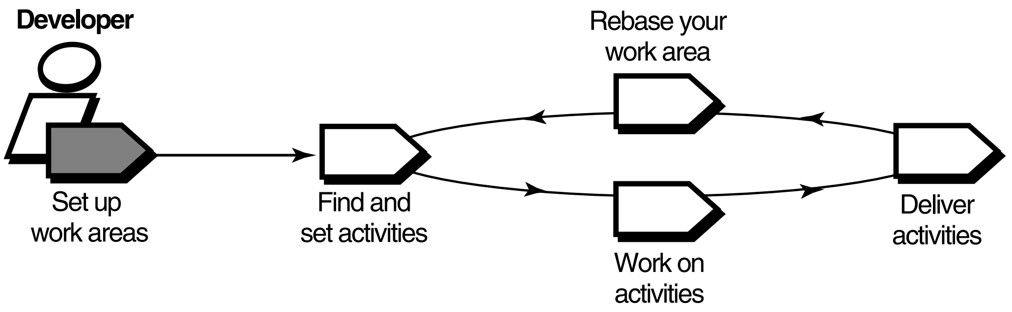
If your organization uses MultiSite, we recommend that you not use UCM serial workflow with teams that are spread across multiple sites and working against multiple replicas. Single-stream projects are not designed for use with teams that are not collocated and working against the same replica. Changing mastership on a single stream restricts people at one site from doing any work while mastership is at another site. Setting up single-stream projects is optimum if the work done in those projects is confined to a single site.

UCM Work Steps

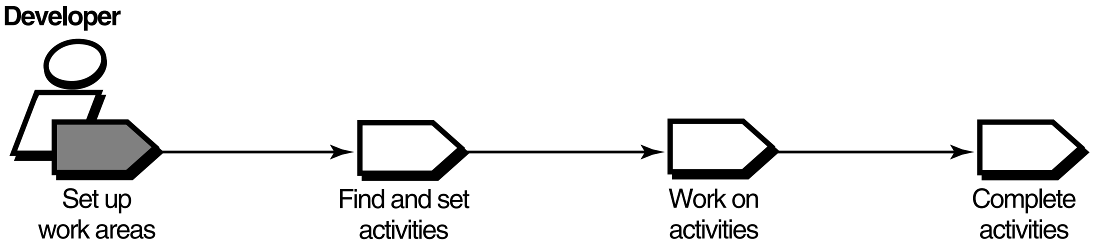
Whether you work in multiple- or single-stream projects, the principles of working in UCM are the same. However, in working in single-stream projects, much of the workflow is simplified in comparison to the workflow in multiple-stream projects.

Setting Up Work Areas

In a Multiple-Stream Project



In a Single-Stream Project



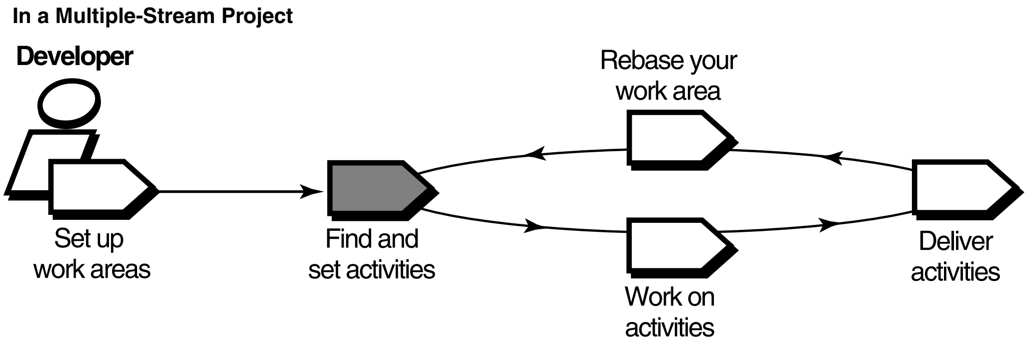
To contribute work to a UCM project, you must set up work areas. This involves joining the project. If you work in a multiple-stream project, you set up two work areas:

- A private work area for working on activities in isolation
- A shared work area for testing the activities you deliver

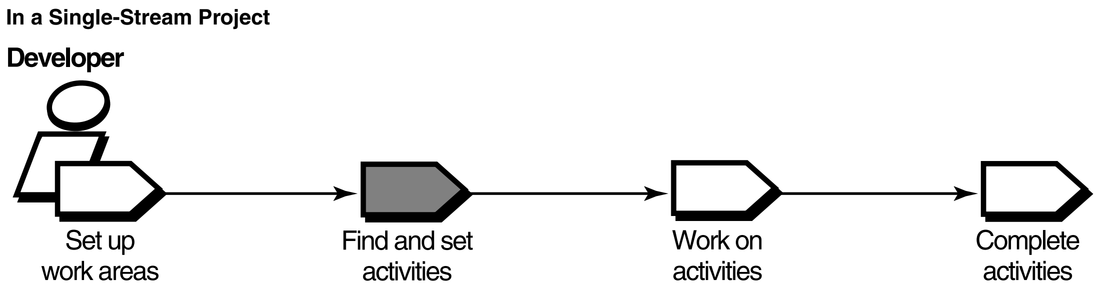
A work area consists of a view and a stream. A view is a directory tree that shows a single version of each source file in your project for you to modify. A stream is a ClearCase administrative object that keeps track of activities and baselines and determines which versions of elements appear in your view. For more information, see *Streams and Views* on page 15.

If you work in a single-stream project, you share a stream with other team members and set up your own view in which you work privately. You do your testing in your view.

Finding and Setting Activities



In a Single-Stream Project



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

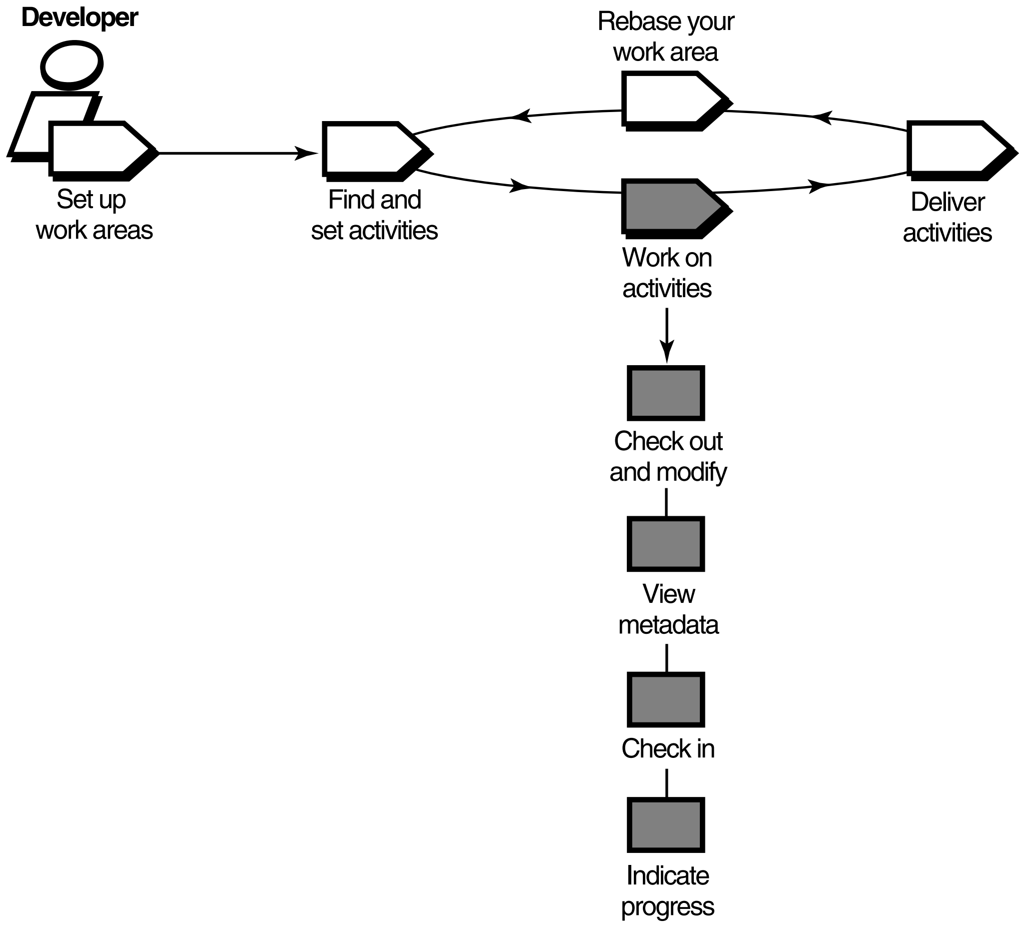
Finding Activities

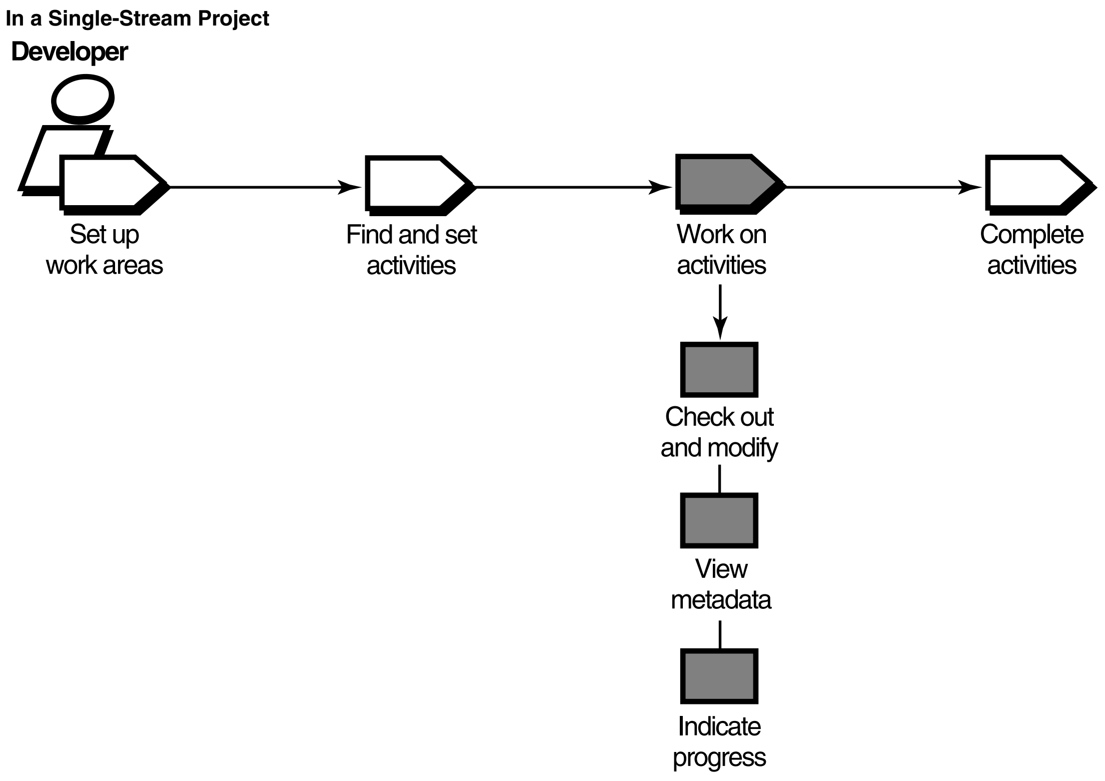
The **cleartool lsactivity** command shows the activities that are in your work area. If your project uses the ClearQuest change-request management system, you can use queries such as MyToDoList to find activities that are assigned to you but are not yet in your work area. For more information, see *UCM-Enabled Record Types and UCM Activities* on page 26 and *Queries* on page 28.

Setting Activities

Setting your view to an activity instructs ClearCase to assign to the activity change set any versions of source files that you create. You must set your view to an activity before you can work on an activity.

Working on Activities





To work on activities, you check out source files, modify them, 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 keeps about your activities and source files. While working with checked-out files (checkouts), you may want to use ClearCase 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

If you work in a multiple-stream project, when you want to keep a record of the current state of a file, check it in. We recommend checking in to store versions for backup. 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 private work area before it is available to other team members.

If you work in a single-stream project, you do not check in until you test your source files and want to complete your activity.

Indicating Your Progress

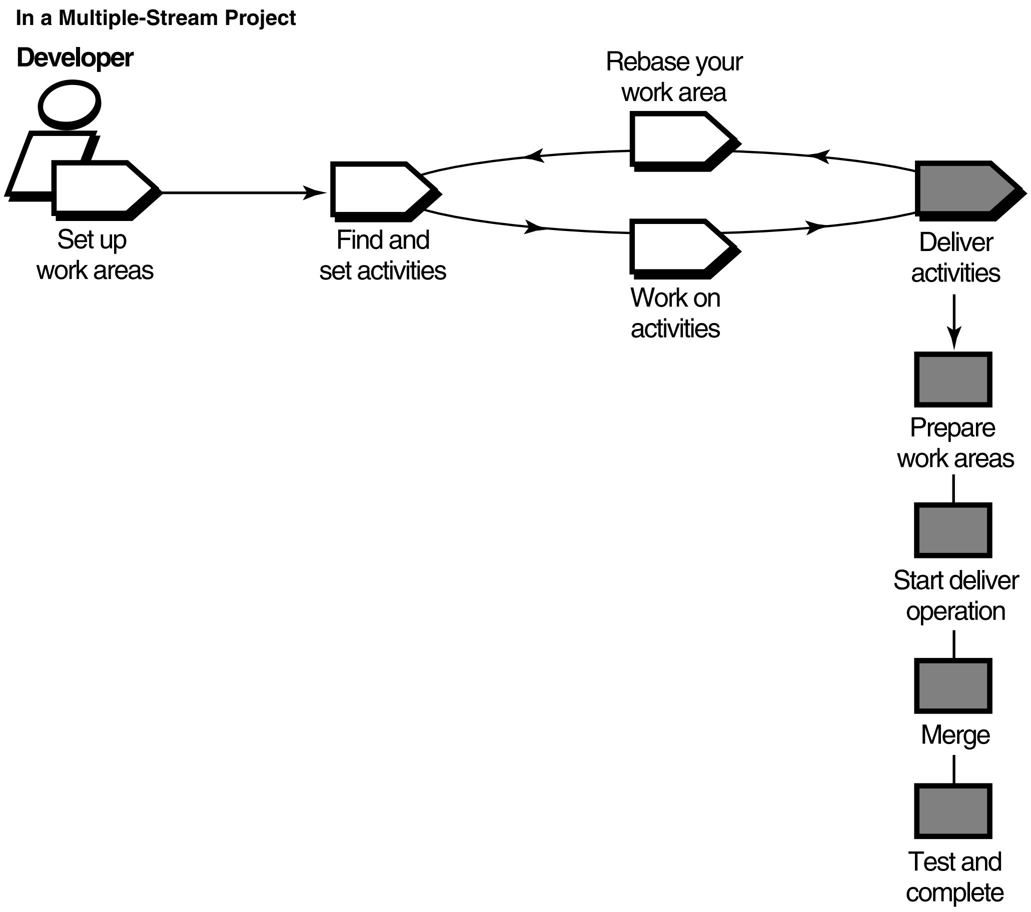
If your project is enabled for ClearQuest, you can indicate your progress on an activity in several ways:

- Modifying information in a ClearQuest user database
- 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. For more information, see *Indicating Your Progress* on page 72.

On a multiple-stream project, if you are ready to make your work available to the rest of the team, you deliver it (see *Delivering Activities* on page 9). On a single-stream project, if you want to indicate that you are done with your work, you complete your activity (see *Completing Activities* on page 11).

Delivering Activities



If you work in a multiple-stream project, other developers on the project do not see your changes until you deliver your work to a shared work area.

When you are ready to make one or more activities available to the project team, prepare your work areas.

Preparing Your Work Areas

To prepare your work areas:

- If your project integrator has created a new recommended baseline since you last rebased, rebase your development area.

  Periodically, the integrator for your project 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.

- 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 shared work area. It completes trivial merges for you. If it encounters merge conflicts, it prompts you to resolve them.

Testing

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

As part of building and testing, you may need to perform other operations:

- 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.

Completing the Deliver Operation

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

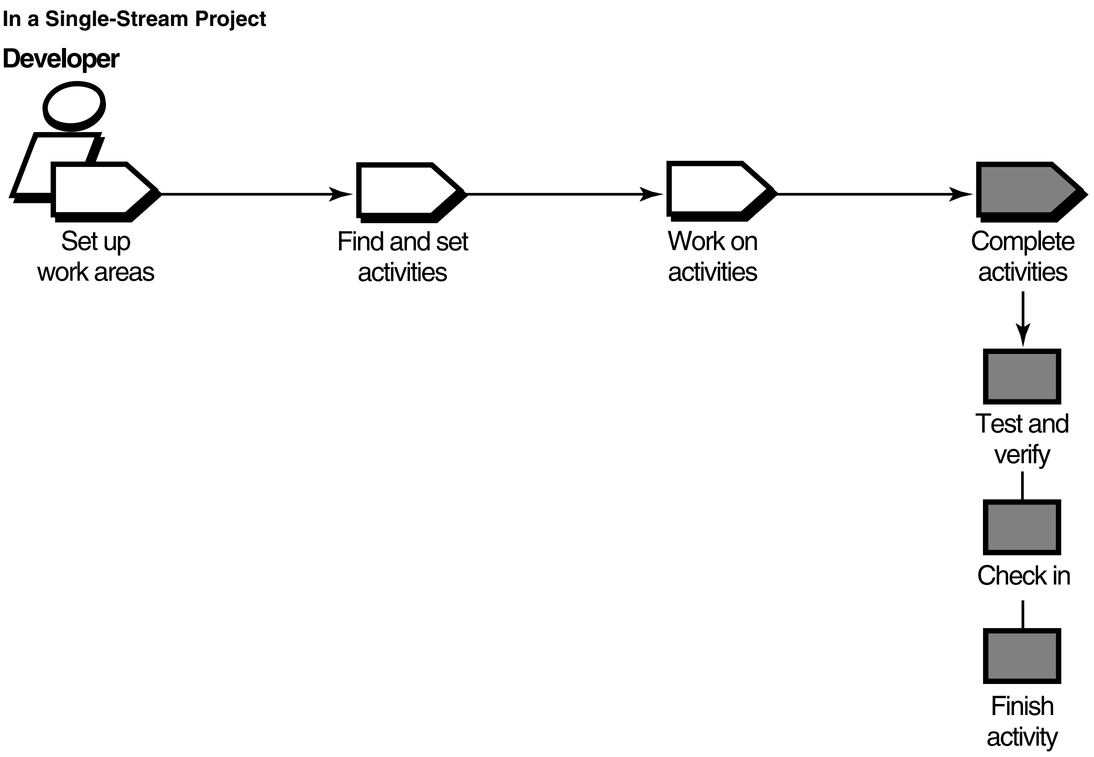
Other Considerations for Deliveries

If your project uses Rational ClearCase 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 source data, you do the following:

- Post a deliver operation to the parent stream replica.
- Notify the integrator at the site that controls your project source data.

The project integrator at the other site merges your activities to the parent stream and tests your work.

Completing Activities

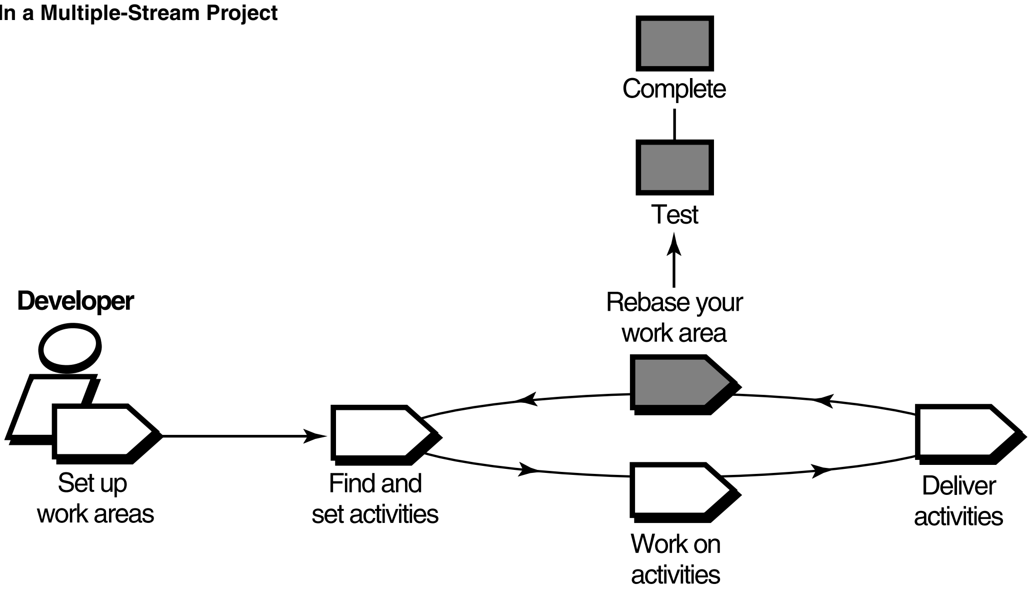


If you work in a single-stream project, other developers on the project see your changes when you check in your work to the shared work area. Because any work you check in from your integration view is immediately available to other team members, you must test your work first.

If your project is enabled for ClearQuest, you indicate that you are done with your work by finishing your activity. For more information, see *Finishing Your Activity* on page 79.

Rebasing Your Development Work Area

In a Multiple-Stream Project



If you work in a multiple-stream project, the project integrator 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 integrator 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.

Under the Hood: ClearCase UCM Concepts

This section describes fundamental ClearCase and UCM concepts:

- Elements and views
- Projects and UCM activities
- Components and baselines
- Streams and their relationship to views
- Delivering, creating baselines, and rebasing

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

Elements and Views

Your project manager places items such as files and directories under ClearCase version control (or source control). Such items are generally categorized as elements. However, files and directories placed under source control are called elements. Each checked-in revision of an *element* is called a *version*.

ClearCase stores elements (and their versions) in data repositories called *VOBs* (versioned object bases).

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 Help.)

Projects

A *project* is the ClearCase 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 product functionality, or for a product release.

Policies in a project 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 other ClearCase objects:

- UCM activities
- Baselines
- An integration stream
- Development streams

A project supports either parallel or serial development. In parallel development, different project members in separate streams can work concurrently with different versions of the same set of source files. In serial development, a small team of project members work from one stream in a simplified work environment.

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

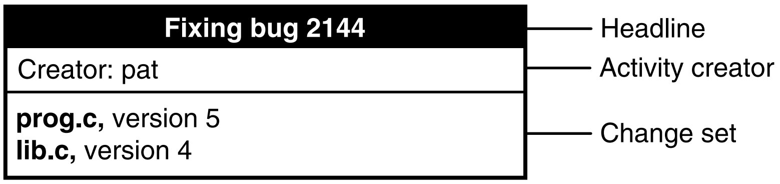
UCM Activities

To create a version in a UCM project, you must first assign it to a UCM activity. A UCM activity is a ClearCase object that identifies the versions created to complete a development task. These versions are referred to as the change set for an activity. 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 UCM 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 1).

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

Figure 1 UCM Activity Object



An activity object belongs to one stream and cannot be moved to another stream. (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 68.)

When you deliver an activity, ClearCase merges the versions in the activity change set to the target stream, but does not move the activity object to the target 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 94.

Components and Baselines

The project manager organizes ClearCase elements into components. Usually, a *component* can be built into a functional unit of software. Multiple projects can use the same component. For the project responsible for maintaining the component, the project manager makes the component modifiable. For projects that may use that component to build other components, the project manager makes the same component read-only.

To keep track of different configurations of versions, your project manager or integrator creates a *baseline*, which records one version of each element in a component. In effect, a baseline is a version of a component (Figure 2).

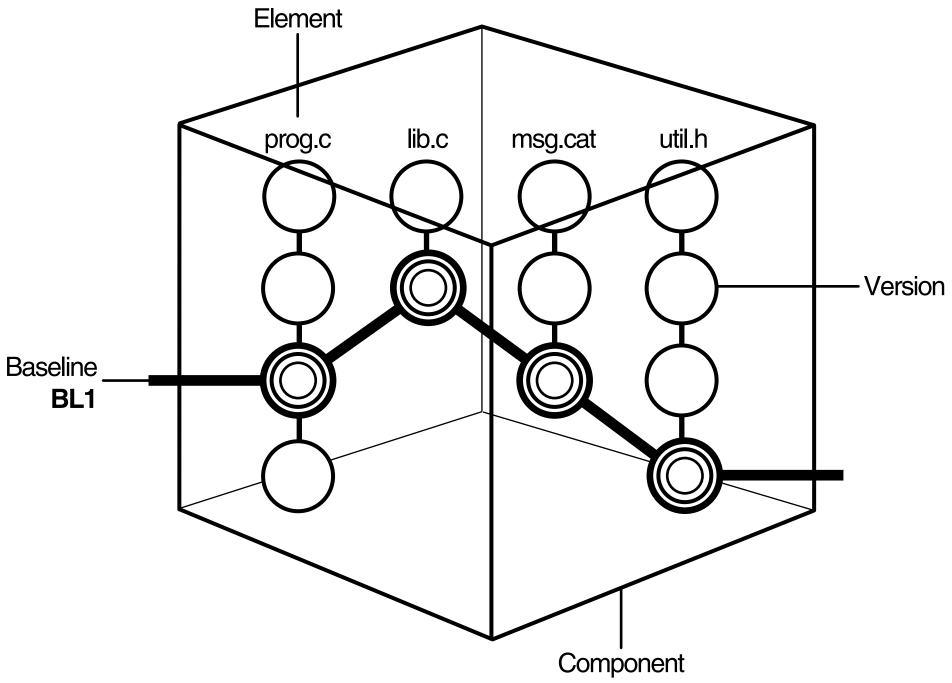
ClearCase stores component and baseline definitions in PVOBs. For more information about components, see *Managing Software Projects*.

Streams and Views

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

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 to which a view is attached determines which versions of elements are in the view. A stream identifies the exact set of versions currently available for you to access, modify, or build.

Figure 2 Elements, Components, and Baselines

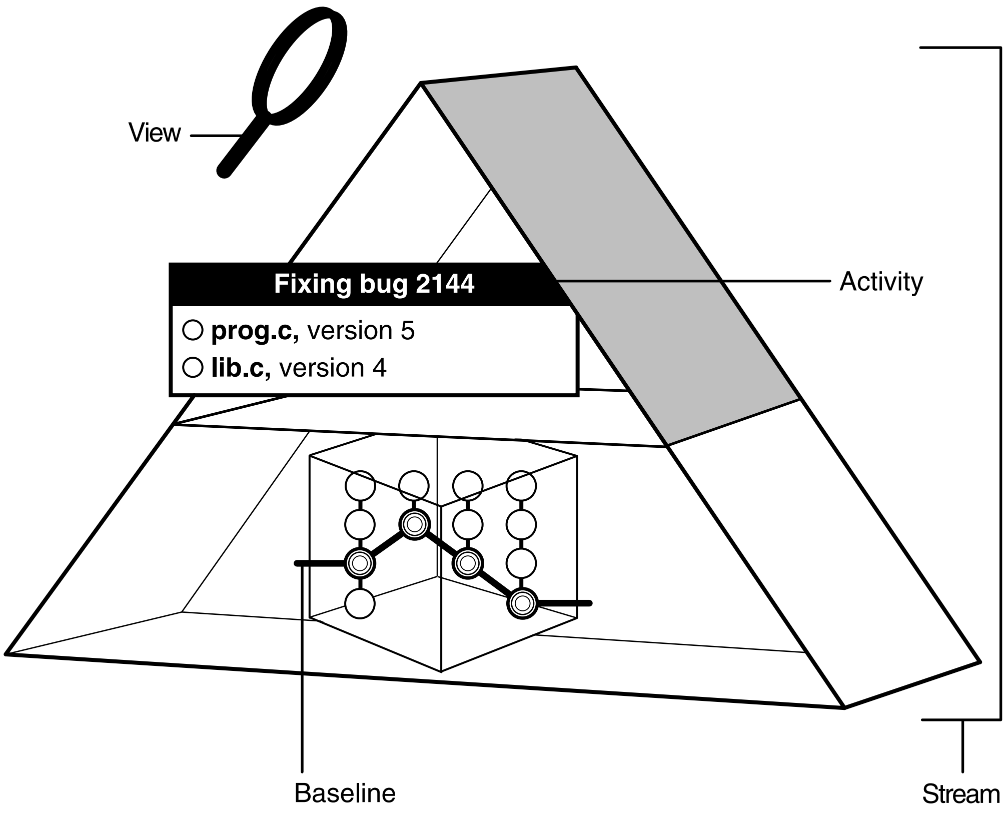


UCM uses baselines and activities to encapsulate a stream configuration in a multiple-stream project. When a stream is created, its original configuration is the same as a baseline (that is, the stream identifies a single version of each element in a component). When you create new versions of elements,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

Your view accesses versions of elements provided by a stream. The view that you use is said to be attached to the stream. Your view for your private work area is called a development view and the related stream is called a development stream.

A multiple-stream project includes two kinds of streams: an integration stream and development streams. A single-stream project does not allow development streams; your view is attached to the integration stream.

Figure 3 A Stream Configuration



In a project, the following actions modify a stream configuration:

- Checking in versions from a view attached to the stream. (Multiple views may be attached to a stream.)

- Rebasing, which replaces the baseline in the stream configuration with a more recent baseline.
- Delivering activities and baselines (multiple-stream projects only).

  When you deliver activities, you change the configuration of the shared work area. This type of delivery adds activities that were previously available only to the contributing development stream. (Delivering activities does not modify the development stream.)

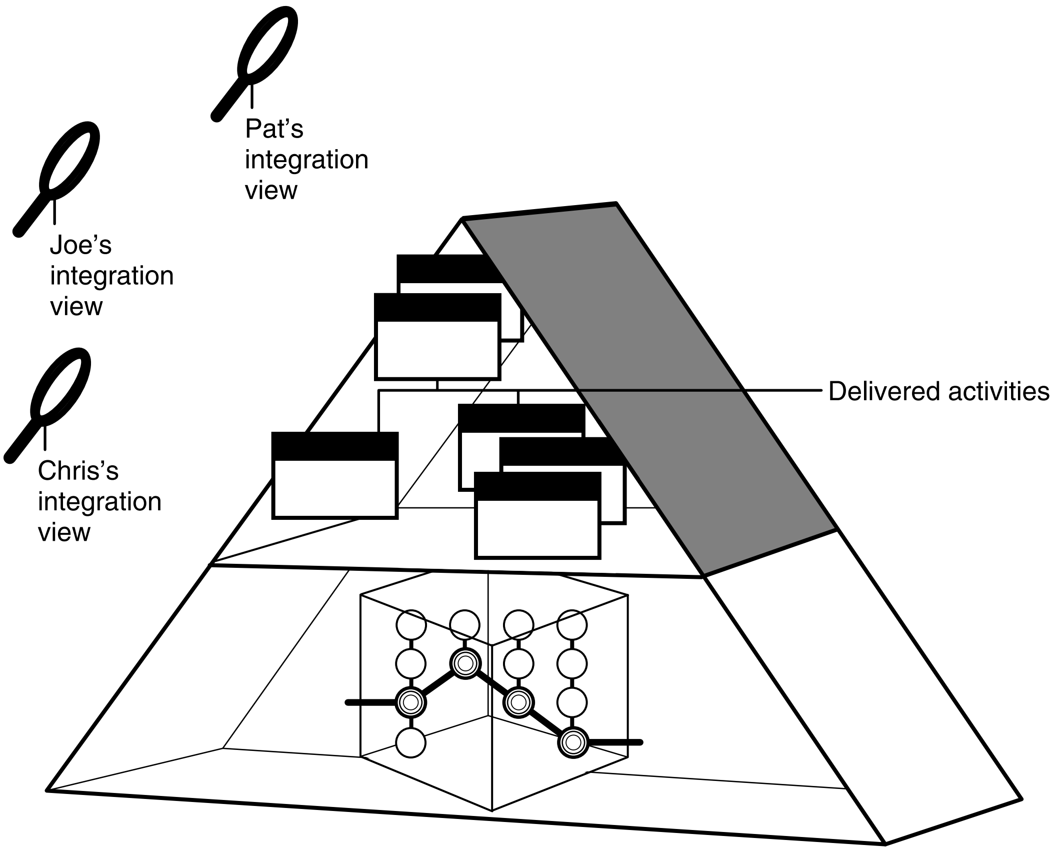
  When you deliver baselines, you introduce new changes to the target stream. This type of delivery is usually performed only by a project integrator.

In a single-stream project, when you check in versions from your view, you modify the configuration of the integration stream. The latest versions in the stream are available to all project members.

Integration Streams

A project contains one integration stream, which is the project main shared work area (Figure 4).

Figure 4 Project Integration Stream



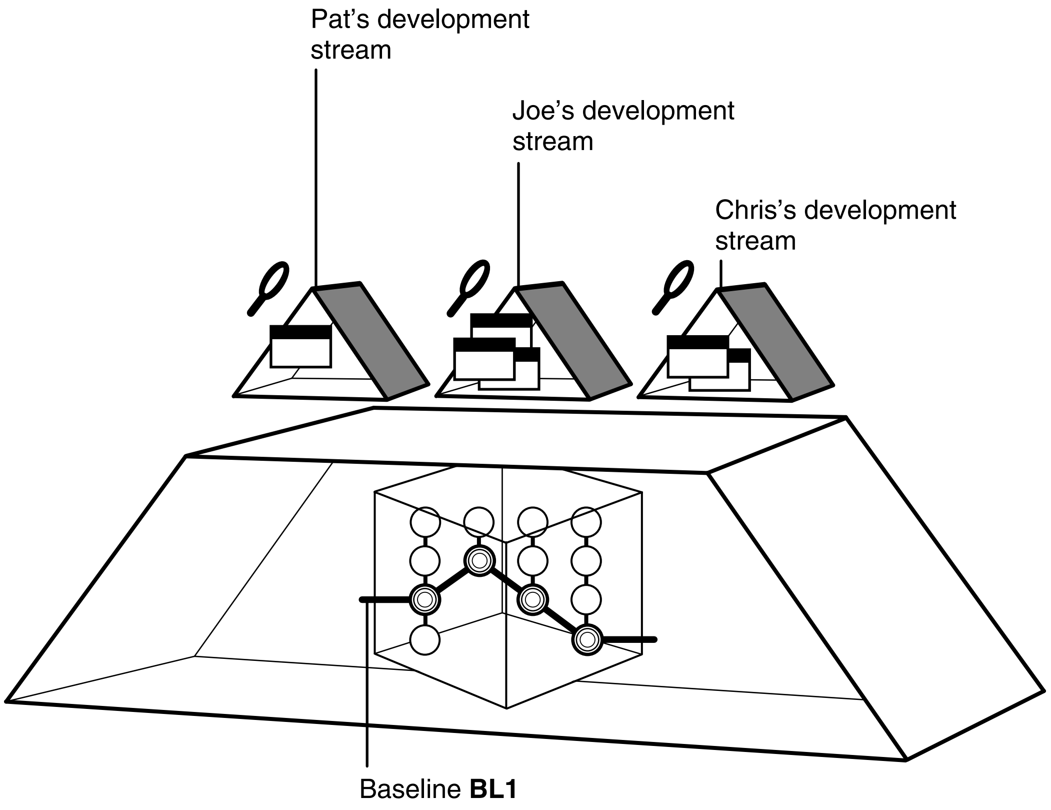
Many integration views can be attached to the project integration stream. The project integration stream records the project baselines and enables access to all versions of the project shared elements. It collects all the work that all team members contribute.

In a single-stream project, all work is done on the integration stream. When you join a single-stream project, you create a view that is attached to the integration stream. The integration stream enables access to the latest versions of the project shared elements.

Development Streams

Typically, each multiple-stream project includes many development streams, one for each developer on the project (Figure 5.)

Figure 5 Many Development Streams



All development streams start from a baseline. As a private work area, your development stream evolves separately from other development streams as you add activities.

Single-stream projects do not use development streams.

Feature-Specific Development Streams

In the basic UCM process, the integration stream is the only shared work area for the entire project. However, a multiple-stream project may have additional shared work areas for developers who work together on specific parts of the project. Using the UCM development stream hierarchy feature, the project can have multiple shared work areas.

A manager of a multiple-stream project can set up a stream hierarchy in which multiple development streams can have child streams. Each parent development stream supports a small team of developers which develops a specific feature. The parent development stream serves as the shared work area for the feature development. For more information about stream hierarchies, see *Managing Software Projects*.

When you join the project, you can create your development stream under a different parent stream than the integration stream, that is, under a feature-specific development stream.

Your Parent Stream

When you join a multiple-stream project, you specify from which stream you will be working. Typically, you join a project at its integration stream. If your project uses feature-specific development streams, you can join the project at the level of the parent development stream rather than at the integration stream.

The stream at which you join the project becomes the parent stream to your development stream and serves as your shared work area. Therefore, if you join the project at its integration stream, the project integration stream becomes your shared work area.

The integration view that you use is attached to your parent stream. Therefore, the integration view shows the baseline for the parent stream and all delivered activities in that stream.

The parent and child relationship defines the default deliver and rebase relationship between the streams. The default deliver relationship says that the child stream delivers to its default target, its parent stream. The default rebase relationship says that the child rebases with the recommended baselines of the parent stream. The default relationships are not modifiable. However, you can deliver your work to a target stream other than the default target.

When you deliver your work, by default, you deliver to the parent stream. Periodically, the project integrator incorporates the delivered work into new baselines. Then, you rebase your development stream to the new recommended baseline of the parent stream.

If you are working from a parent development stream and all the developers finish working on the feature, they deliver their last work to the parent development stream. The integrator then incorporates all the delivered work into a final set of baselines. A parent development stream is an intermediate integration area, the changes in which migrate upward to the project integration stream.

Delivering Your Work

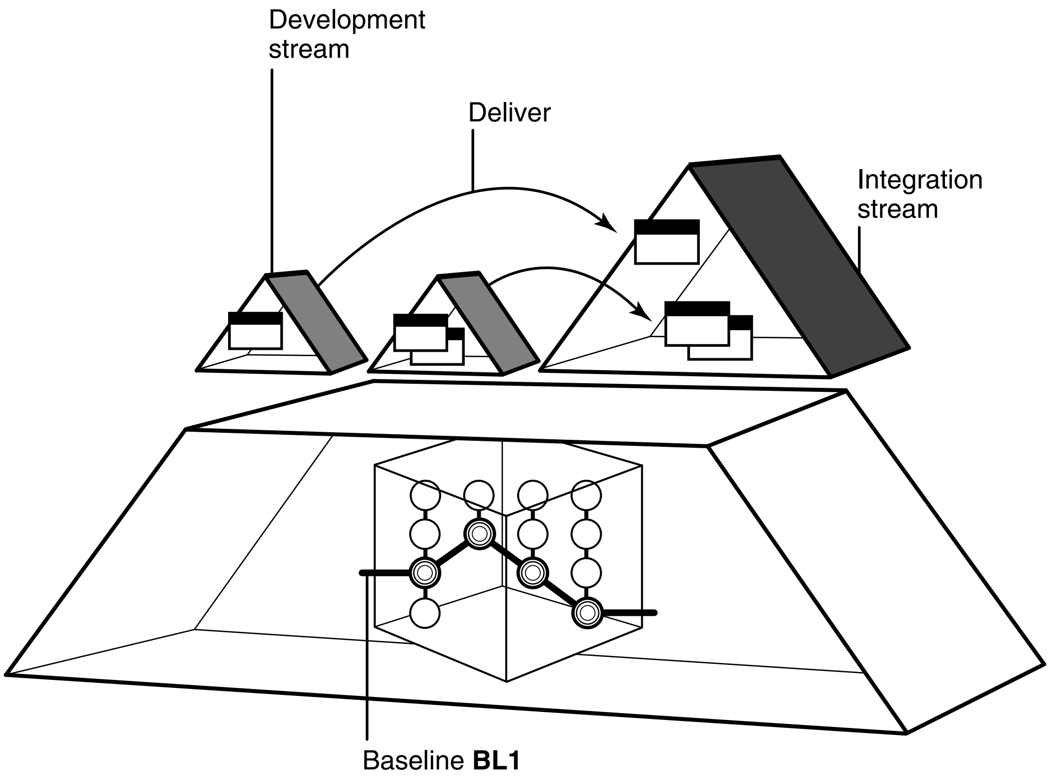
In a multiple-stream project, you start work from a baseline of your parent stream, which becomes your shared work area. To facilitate delivering activities, you have a separate integration view attached to your parent stream. Your integration view shows the baseline for your parent stream and all delivered activities for that stream. By default, you deliver activities from your development stream to your parent stream. The parent stream is the default target for delivering your work.

If your parent stream is the project integration stream, the default target for your deliver operation is the integration stream (Figure 6).

The integration stream identifies a shared set of versions to be used for projectwide building and testing.

If your parent stream is another development stream, the default target for your deliver operation is that development stream (see *Your Parent Stream* on page 20). When you deliver your work, you merge it with the work in your parent development stream rather than with the integration stream. Work from multiple developers is delivered to the parent stream and tested. When all the work in the parent development stream is complete, that work is delivered to its parent stream.

Figure 6 Delivering to the Integration Stream



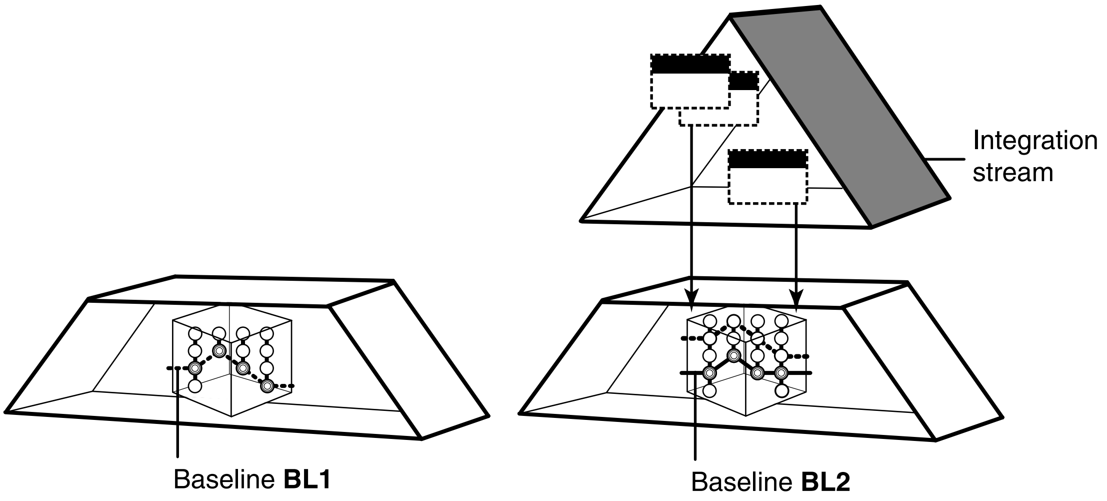
Eventually, the integrators merge all the work from all development streams into the project integration stream.

Baseline Creation

On projects, project integrators incorporate the work delivered to parent streams into new baselines (Figure 7).

For example, the activities with changes delivered to the project integration stream are included in the new baseline **BL2**. The new baseline records the current configuration of the component in the integration stream.

Figure 7 Creating a Baseline for the Integration Stream

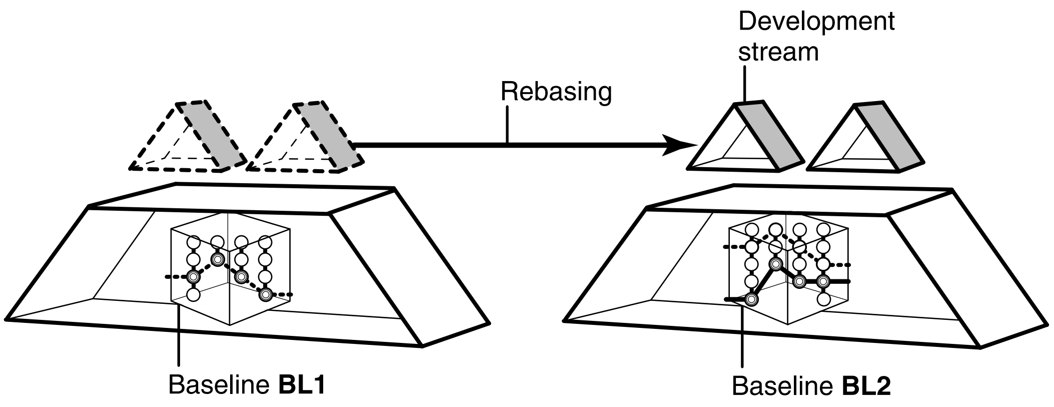


When the integrators build and test these new baselines according to the criteria established for the project, they characterize as recommended the specific baselines that include stable and significant changes. The recommended baseline becomes the new stable configuration for the parent stream. For example, if the integration stream recommends these baselines, they become the required configuration for all development streams whose parent stream is the integration stream. Similarly, for a parent development stream, integrators must recommend new baselines to establish stable configurations for their child development streams.

Rebasing Your Work Area

On a multiple-stream project, you update your development work area with a new baseline by performing a rebase operation. This operation merges files and directories from the parent stream to your development stream and allows you 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 Your Work Area**



This allows your work area to stay synchronized with other team member's work. By rebasing often, you minimize the overhead of merging excessive numbers of changes.

Display of ClearQuest User Database Information

Rational ClearQuest provides a user database of change-request management information that can integrate with UCM projects. Access to a ClearQuest user database is through the ClearQuest client. The tree controls under Workspace in the ClearQuest client provide navigation to folders and queries in a ClearQuest user database (see *Logging On to a ClearQuest User Database* on page 43).

Displaying UCM Activities Records

Under Workspace in the ClearQuest client, you see two high-level folders, **Personal Queries** and **Public Queries** (see *Queries* on page 28). The **Public Queries** folder typically contains database search requests provided with the software and by your project administrator.

When you run a query (see *Queries* on page 28), the result set displays data from activity records returned from the ClearQuest user database, one line per record. Filters defined in the query limit the number of records returned. If you run a query without any filters, all of the records in the database are returned.

The Record form, displayed below the Query builder pane, shows data and state information for the activity record whose line is selected in the result set. Clicking another line updates the Record form with information for the newly selected record. Access different fields and state information by clicking tabs in the Record form. The tabs available vary based on integration support. When you click a tab whose area is not visible, the software accesses and displays the related information from the database.

Typically, entries in **Public Queries** are read-only. However, you can modify a public query for your personal use. For example, you could run a public query that found all open defects for all components. Then, you could click **Query editor** in activity details to modify the filters to find such records for just your components. You can run a modified query to sample its results. When you are satisfied, you can save it under a folder in **Personal Queries**. You can also export queries to a file to share them with other users.

Controlling the Result Set Display

Click **Display editor** to control which fields are displayed in each line for records returned in queries. Double-click a column heading to change sort order in the column. You can also right-click to use commands on a shortcut menu.

Controlling Record Content

Buttons to the right of the record form provide commands used to modify fields and transition records. Commands available on these buttons vary with the integrations distributed with ClearCase. A control in the bottom left corner of the record form allows you to single-step through records in a result set, jump to the first or last record, and provide an ID of any record to retrieve.

Under the Hood: ClearQuest and UCM Concepts

The fundamental concepts for the UCM-ClearQuest integration are:

- Activities
- Integrating elements and activities
- Schema enabled for UCM
- UCM-enabled record types and UCM activities
- Queries and UCM-specific queries
- State types and transitions

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

Activities

Your UCM project manager can use ClearQuest customizable defect and change tracking features to set up a repository to store information about your work. Information used to define, schedule, and track your work is referred to as an *activity*. Using these features, your project manager can manage every type of change activity associated with software development, including enhancement requests, defect reports, and documentation modifications. A repository that stores this information is referred to as a ClearQuest user database.

Integrating ClearCase and ClearQuest

Your project manager can customize your work environment to fit activity management into your UCM workflow. Projects can be enabled for the UCM-ClearQuest integration. Your project manager can relate the elements and versions on which you work to the information about your work. For example, a ClearQuest user database could be set up to contain defect reports about software for which you have development responsibility. When you start working on the versions that relate to the defect, the software asks you to provide information that is later stored with the defect. Your project manager can then examine information in the ClearQuest user database to see the state of your work regarding that defect. The same capability can be used to track progress on requests for enhancement on the software that you work on.

If your project manager integrates elements and activities, a schema that is enabled for UCM is provided. Your project manager can tailor the schema to the ways that your organization does its work.

The UCM-ClearQuest Schema

ClearQuest includes predefined schemas that provide change and defect tracking processes and integration with UCM projects.

A *schema* defines the types of records in a ClearQuest user database, the states available to each type of record, and other attributes of the database. A schema basically defines the process for how developers work with records in a ClearQuest user database. An organization may have many different ClearQuest user databases, each one using different schemas for varying types of records tailored for specific purposes.

A schema includes:

- Record type definitions
- Forms used to submit and modify a record
- Field definitions and behavior
- States a record can be in

- Actions used to modify or change the state of a record
- Hooks for scripting languages that allow fields and actions to be customized

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

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

Record Types and Records

Each change request is stored as a record in a ClearQuest user database. A record type is template that defines the actions, fields, behaviors, and possibly state information associated with a record. Schemas can contain state record types for records that can move from state to state and stateless record types for records that remain static.

An activity record has associated states and actions. The state indicates what stage in processing a record is in. A developer uses an action to change the state of a record. The schema also governs the transition from one state to another, determines which actions can change which states, and which states are possible from other states.

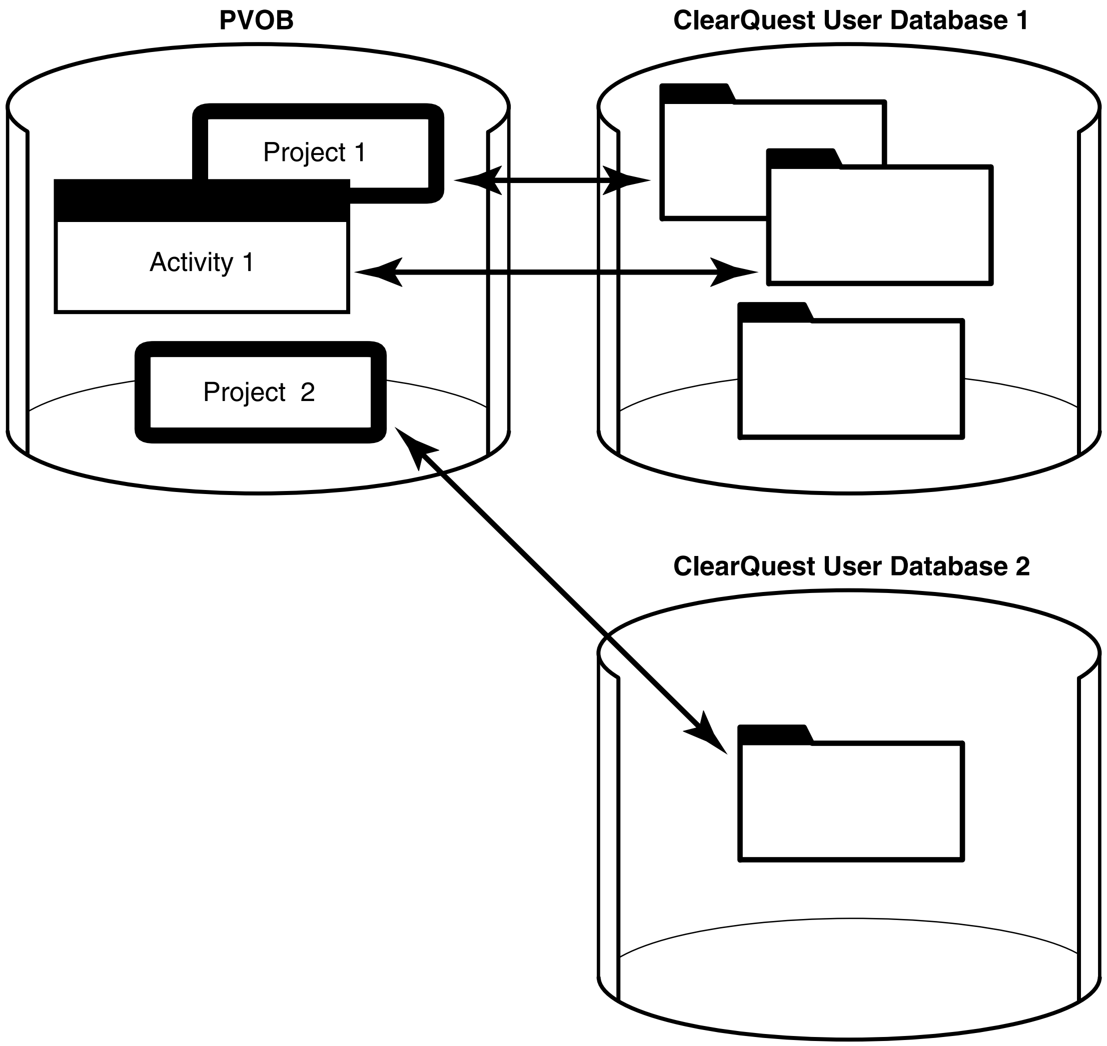
UCM-Enabled Record Types and UCM Activities

A UCM-enabled record type is a template that includes definitions for a set of fields that stores information about UCM activities. Your ClearQuest user 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.

Projects Enabled for ClearQuest

If a project is enabled for ClearQuest, ClearCase provides links between PVOBs and ClearQuest user databases (Figure 9). A ClearQuest user database stores change requests as user-owned records.

Figure 9 A PVOB Connected to ClearQuest User Databases



One link connects the project in the PVOB with a ClearQuest user database. Multiple ClearQuest user databases can be used. Another link connects the project UCM activity objects in the PVOB to UCM-enabled records in the project associated ClearQuest user database.

UCM activity objects in the PVOB can be linked with activity records based on a UCM-enabled record type in the ClearQuest user database (see Figure 9). The link enables ClearCase to display information about the UCM activity (such as its change set, its stream, and whether it is currently set in any view). It also enables policies that govern when you can deliver an activity described in the PVOB and when you can close an activity in the ClearQuest user database. Because of the close association between linked UCM activities and UCM-enabled records, the UCM documentation usually refers to both linked entities as *activities*.

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

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

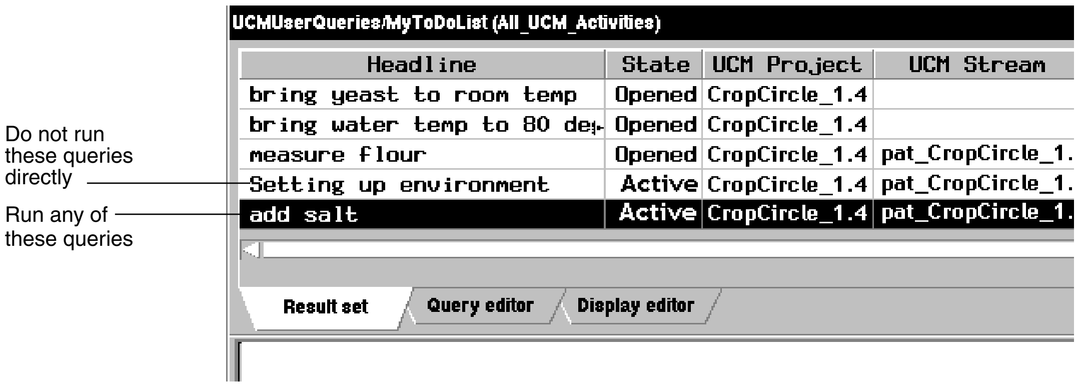
Queries

Queries are the means for navigating through the ClearQuest user database. Before viewing or modifying records in a ClearQuest user database, 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 that are associated with your project, or you may want to see activities created before or after a particular date.

UCM-Specific Queries

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

Figure 10 UCM-Enabled Queries in a ClearQuest User Database


Do not run these queries directly
Run any of these queries
UCMUserQueries/MyToDoList (All_UCM_Activities)
Result set
Query editor
Display editor

Headline	State	UCM Project	UCM Stream
bring yeast to room temp	Opened	CropCircle_1.4	
bring water temp to 80 de	Opened	CropCircle_1.4	
measure flour	Opened	CropCircle_1.4	pat_CropCircle_1.
Setting up environment	Active	CropCircle_1.4	pat_CropCircle_1.
add salt	Active	CropCircle_1.4	pat_CropCircle_1.

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

Do not run queries in the UCMSystemQueries folder; they are intended to be run by the 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 your own queries.

MyToDoList

The MyToDoList query finds UCM-enabled records that match all of 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 to which the current view is attached

The ClearCase checkin and checkout dialog boxes present the activities that UCMCustomQuery1 finds. Although you can also see this query in the context of a

ClearQuest user database, it is not designed to be run outside the context of a ClearCase view.

Your project manager can modify this query, but you cannot.

Other Queries

A schema enabled for UCM 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.
AllActivitiesinStream	For specific stream, lists all activities regardless of state type.
MyCompletedWork	Lists all activities in a completed state type.
UCMProjects	Selects all UCM-project records in ClearQuest user database.

State Types and State Transitions

Change requests in a ClearQuest user database move through a pattern, or lifecycle, from submission through resolution. Each stage in this lifecycle is called a *state*, and each movement from one state to another is called a *state transition*.

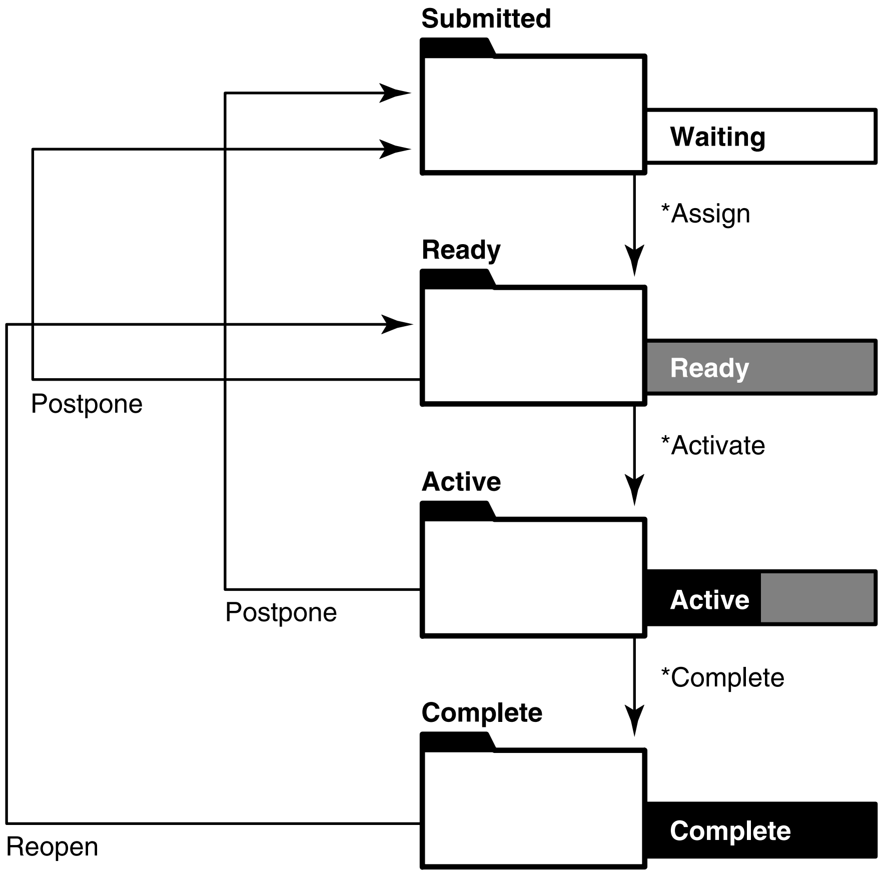
As with record types and records, a state type is a template that defines actions and other attributes associated with a state. The states in a tailored schema must be based on one of the following state types:

- Waiting
- Ready
- Active
- Complete

State Transitions

Schemas include rules for changing records from one state type to the next. Some examples of state transitions are shown in Figure 11.

Figure 11 Example of ClearQuest User Database State Transitions



Some state transitions that records in a schema must follow are:

- **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.
- **Ready**. Assigning an activity usually places it in a state based on this state type to indicate that the work is pending.

- **Active**. Setting an activity in a state based on this state type indicates that the work is in progress.
- **Complete**. When you have completed an activity, you change it to a state based on the Complete state type. Activities in this state do not appear in MyToDoList.

State Types and State Transitions in a Schema Enabled for UCM

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

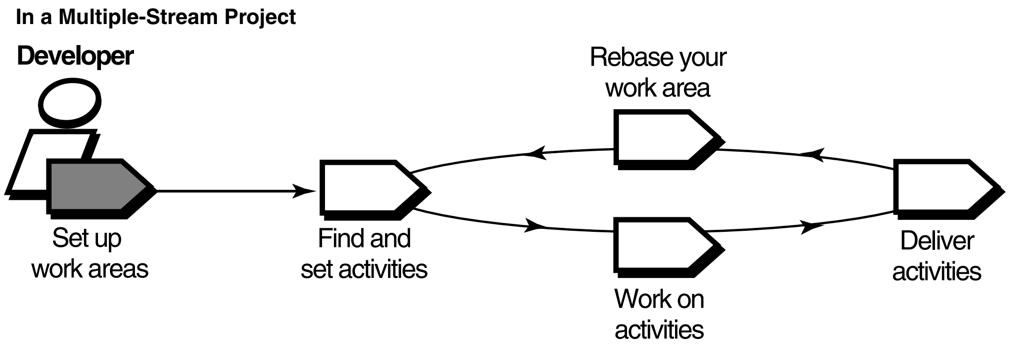
In a schema enabled for UCM, records must follow these state-transition rules:

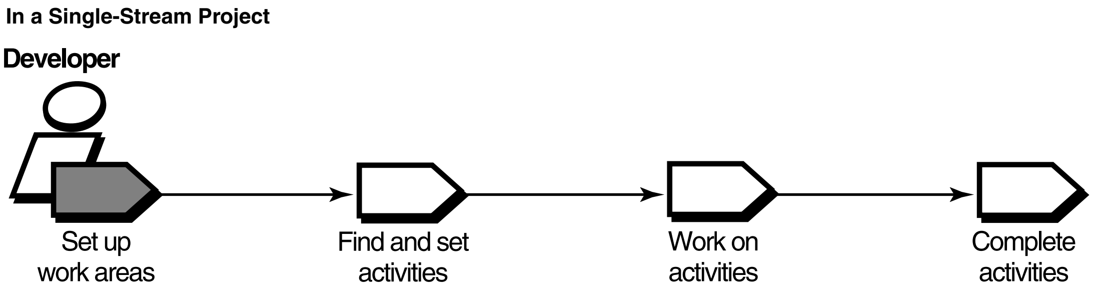
- **Waiting**. Creating an activity places it in a state based on this state type to indicate, for example, that the work is waiting for someone to resolve dependencies and that it is not ready to be scheduled. Activities in this state do not appear in MyToDoList; they may or may not contain a value 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 state 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 UCM-enabled record to the UCM activity. After the UCM activity is linked to the record, you can change the activity record owner but you cannot change the stream that contains the activity. For more information, see *Reassigning Activities* on page 75.

  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 the stream. The schema in Figure 11 bases its Complete state on the Complete state type.

Setting Up Work Areas

2





Before setting up work areas, consider adjusting your **umask** setting to control the level of access others have to your work. (See *Adjusting Your umask* on page 34 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 on to a Rational 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 the *Command Reference*.

For information about the differences in the tasks for single-stream projects, see *Setting Up Work Areas in a Single-Stream Project* on page 43.

Adjusting Your umask

Your **umask** 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, see a **umask(1)** man page.

The CCASE_BLD_UMASK Environment Variable

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

For more information about ClearCase environment variables, see the **env_ccase** reference page in the *Command Reference*.

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:

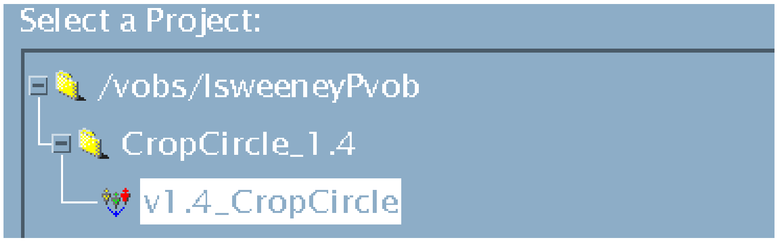
clearjoinproj

The Select a Project dialog box opens.

Choosing a Project

In this page of the Join Project Wizard, choose a project. Navigate the list of PVOBs, folders, and projects and click the project you want to join (Figure 12).

Figure 12 List of Projects



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

After you select a project to join, click **Next**. If you join a multiple-stream project, you next verify stream names. If you join a single-stream project, you next set up your integration view (see *Setting Up Your Views* on page 37).

Verifying Stream Names

After you choose a multiple-stream project, this step of the Join Project Wizard opens the Set Stream dialog box, which presents the two streams that Rational ClearCase uses to keep track of your work on this project.

Development Stream

If you do not have any streams defined in the selected project, the Join Project Wizard gives your development stream a name 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**. If you want to add a comment to describe the stream that you are creating, click **Advanced Options** and, in the **Comment** box, enter your description. If you want to work on activities in an existing development stream, click **Advanced Options**, click **Use existing development stream**, and select a development stream.

Although you can change the name 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. This is the stream to which you typically deliver your work and from which you update your work area. The Join Project Wizard presents the name of the project integration stream as a default. The integration stream becomes your parent stream for this project. It is the stream to which your integration view will be attached. This stream also is the default target to which you deliver your work and from which you update your work area.

After you verify your stream names, you set up your views (see *Setting Up Your Views* on page 37).

Choosing a Different Parent Stream

If your project uses feature-specific development streams, you can join the project at the development stream level rather than the integration stream level (see *Feature-Specific Development Streams* on page 19). To use one of these parent development streams, click the browse button to the right of the name of the default integration stream and navigate to and select one of the development streams listed.

This development stream, rather than the integration stream, becomes your parent stream for this project. Your integration view is attached to this parent stream. This stream also is the default target to which you deliver your work and from which you update your work area.

After you choose a different parent stream, you set up your views.

Setting Up Your Views

In the next steps of the Join Project Wizard, you set up your development view and integration view (multiple-stream project) or simply set up your integration view (single-stream project) 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 *Streams and Views* on page 15, 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 does not use the *build auditing* and *build avoidance* features of ClearCase.

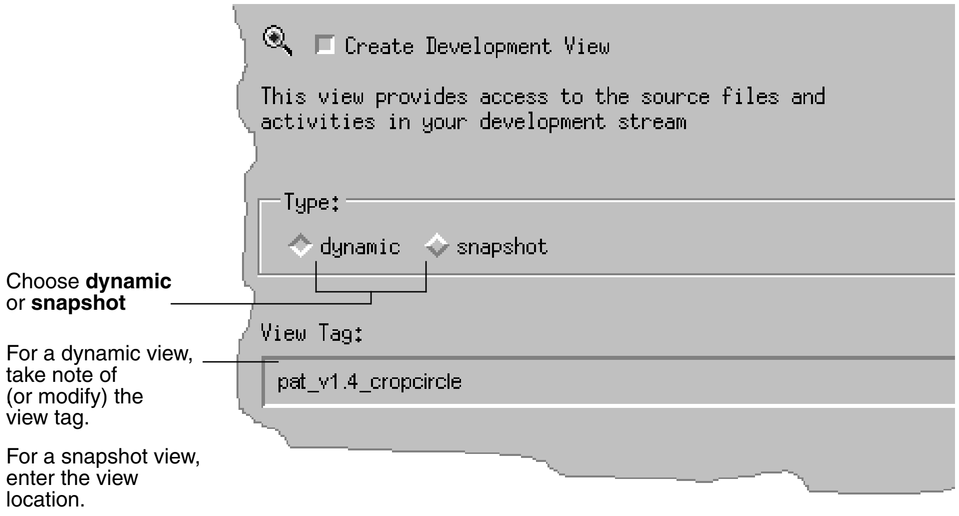
Work in a dynamic view when any of these conditions is true:

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

Because a dynamic view can show changes in its attached 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


Create Development View

This view provides access to the source files and activities in your development stream

Type: dynamic snapshot

View Tag: pat_v1.4_cropcircle

Choose **dynamic** or **snapshot**
For a dynamic view, take note of (or modify) the view tag.
For a snapshot view, enter the view location.

For a detailed comparison of snapshot views and dynamic views, see the Help and the *Administrator's Guide* for Rational ClearCase.

Locations for Snapshot Views

When creating a snapshot view, you must specify a view directory into which ClearCase loads, or copies, versions of source files from VOBs into the snapshot view at the directory you specify.

When choosing a directory, consider these constraints:

- The view root directory must be located on a disk with enough space for the files loaded into the view and any other files you add.
- 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 want to access the view from other workstations, it must be located in a directory that is accessible to other workstations; 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 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 location when you create a view.

If your ClearCase administrator does not set up storage locations, ClearCase sets a directory under the root directory of the snapshot view as the default location for the view storage directory. This is called a co-located view storage directory.

You can override the default location. If your administrator sets up multiple storage locations, you can select one explicitly. You can place the view storage directory under the root directory of the snapshot view.

If you co-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 *view_server* process is running.
- Make sure that the view storage directory is accessible to any data backup schemes your organization institutes.

  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 *To Register a Snapshot View* on page 123.

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 that you choose a name that indicates the view owner and the stream to which the view is attached.

Under the Hood: View Storage for Dynamic Views

The first time you create a dynamic view, the wizard provides a default path for the storage directory. Click **Advanced Options** to provide a different path 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, see the **clearmake** reference page in the *Command Reference*.

Valid Locations for Dynamic View Storage Directories

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

- The directory must be located on a ClearCase host or on a Network Attached Storage (NAS) device. View processes (specifically, *view_server* processes) run on the computer that physically stores the view storage directory or on the computer that can access a location on a supported NAS device which contains 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 that your organization institutes.

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 *Changing Which Elements Are Loaded into a Snapshot View* on page 119.

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.

Accessing Your Development View

After creating your development view, the Join Project Wizard offers to open your development view in a shell or File Browser (**xclearcase**), a graphic user interface 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, see the following sections:

- *Accessing a Snapshot View* on page 42
- *Accessing a Dynamic View* on page 42

Accessing a Snapshot View

Recall that when you create the view, ClearCase loads one version of each element in the project *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 path:

```
~/pat_v1.4_cropcircle
```

The view files are located in the ~/pat_v1.4_cropcircle directory. (See *Locations for Snapshot Views* on page 38 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. If you can access the other workstation and that the directory owner has set up the proper permissions, you can use the **cd** command to access the view.

To use someone else's snapshot view in a ClearCase operation (for example, in a deliver or rebase operation), register the view (see *Registering a Snapshot View* on page 122).

Accessing a Dynamic View

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

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. (Figure 13 on page 38 shows where to provide the view tag in the Join Project Wizard.)

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

Because ClearCase mounts public VOBs, you have to mount any private VOBs that you want to access (see *Mounting or Unmounting VOBs for Dynamic Views* on page 121).

Logging On 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, ClearCase prompts you to enter your ClearQuest user ID and password.

ClearQuest keeps a database of user IDs and passwords. (ClearQuest can use your standard network user ID and password.) Make sure that your ClearQuest user database administrator has added a user ID and password to the ClearQuest user database for you.

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

Setting Up Work Areas in a Single-Stream Project

The tasks that you perform while setting up work areas in a single-stream project are similar to those for a multiple-stream project. However, there are a few differences in the details.

- You do not verify stream names.

  You cannot create development streams in a single-stream project. Therefore, you create only one view that is attached to the integration stream, the only stream in the project.

- You create only an integration view when you run the Join Project Wizard.

  In single-stream projects, the integration stream is the sole project work area. Each developer has a view (an integration view) in which to work privately. Your integration view is attached to the integration stream, the shared work area for the project.

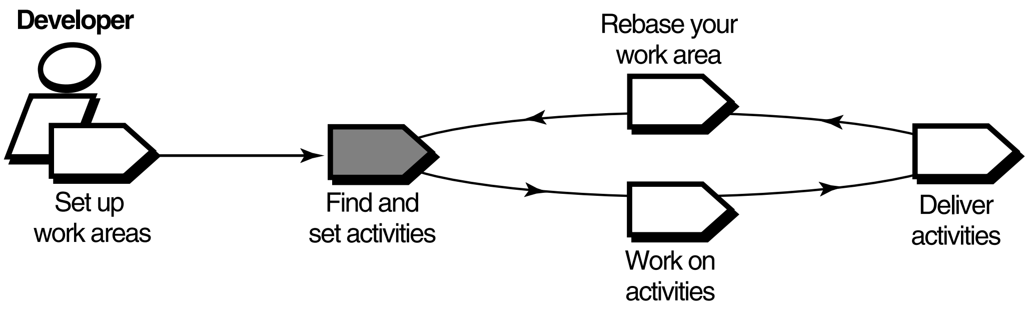
- You access your integration view rather than your development view.

  In a single-stream project, you have no development views. However, to access your integration view, you follow the same steps as you would to access development views (see *Accessing Your Development View* on page 41).

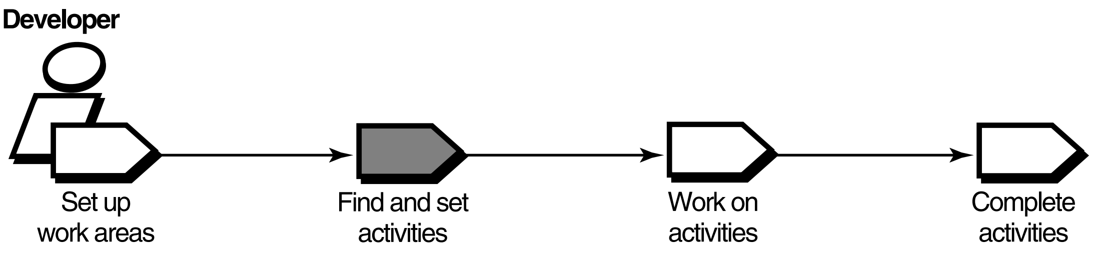
Finding and Setting Activities

3

In a Multiple-Stream Project



In a Single-Stream Project



Before you start work on the project, find any *activities* that your project manager or other team members 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

The descriptions in this chapter apply to both multiple- and single-stream projects. However, for single-stream projects, substitute the terms integration view and integration stream where the text refers to development view and development stream.

Note: Rational ClearCase provides a Web interface through which you can complete some of the tasks described in this and subsequent chapters (see *Introduction* for Rational ClearCase). Ask your system administrator how to access the Web interface. For more information, see the Web interface Help.

Finding Activities

UCM provides several locations from which to find activities:

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

Finding Activities with MyToDoList and Other Queries

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

To Start the ClearQuest Client

Enter the following command to start the client:

```
clearquest
```

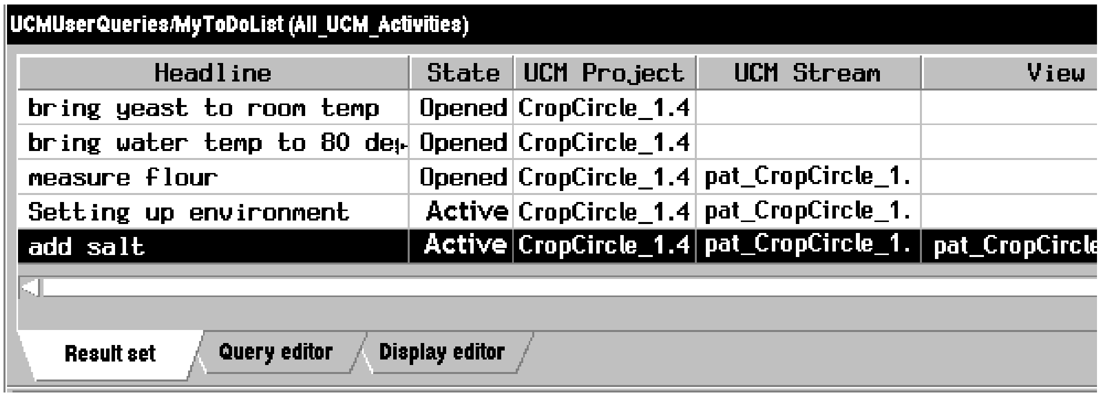
The login dialog box appears. For information about connecting to ClearQuest, see *Logging On to a ClearQuest User Database* on page 43.

When you are logged in to the ClearQuest user database, the ClearQuest client window appears. For information about the ClearQuest client, see *Display of ClearQuest User Database Information* on page 23 and Help.

Using MyToDoList

The MyToDoList query finds all UCM-enabled records that are assigned to you in a ClearQuest user database, 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 28 and *UCM-Enabled Record Types and UCM Activities* on page 26.

Figure 14 Linked Activities in the MyToDoList Query


UCMUserQueries/MyToDoList (All_UCM_Activities)

Headline	State	UCM Project	UCM Stream	View
bring yeast to room temp	Opened	CropCircle_1.4		
bring water temp to 80 de	Opened	CropCircle_1.4		
measure flour	Opened	CropCircle_1.4	pat_CropCircle_1.	
Setting up environment	Active	CropCircle_1.4	pat_CropCircle_1.	
add salt	Active	CropCircle_1.4	pat_CropCircle_1.	pat_CropCircle

The result set for MyToDoList shows the view in which an activity is currently set and lists the UCM stream if the activity is related to an activity object.

To Open the MyToDoList Query

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

The query output displays in the Query results pane.

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

3 To see more information about an item on your to do list, select it. ClearQuest displays the record 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:

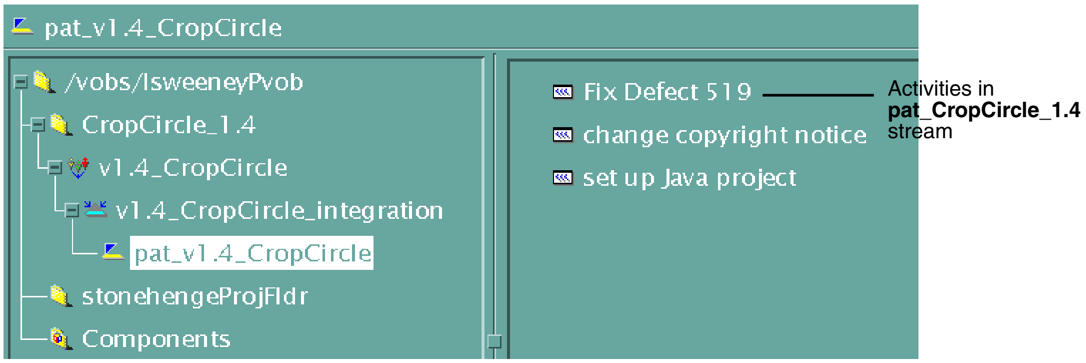
- To search across all UCM-enabled record types, in the Choose a Record dialog box, select **All_Ucm_Activities**. This selection limits your query to fields common to all UCM-enabled record types.
- You cannot use a query to find ClearCase change-set information.

For more information about creating a query, see Help.

Finding Activities with Project Explorer

The Project Explorer arranges activities hierarchically by stream and by project (Figure 15).

Figure 15 Activities in Project Explorer



To start the Project Explorer, enter the following command:

clearprojexp

The Project Explorer details pane shows only the UCM 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 the associated ClearQuest user database for the project. For more information, see *UCM-Enabled Record Types and UCM Activities* on page 26.

Finding Activities with cleartool lsactivity

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

To List the Activity Objects in Your Development Stream

1 Access your development view. (For more information, see *Accessing Your Development View* on page 41.)

2 Enter the following command:

cleartool lsactivity –cview

For example:

```
% cleartool setview pat_1.4_cropcircle
% cd /guivob
% cleartool lsactivity –cview
05-Aug.09:14:17  set_up_Java project  pat “set up Java project”
06-Aug.14:17:19  change_copyright_notice   chris “change copyright notice”
```

In this example, two activities are in the stream attached to **pat_1.4_cropcircle**: `set_up_Java_project` and `change_copyright_notice`. The string `05-Aug.09:14:17` indicates the activity creation date and time, the string `set_up_Java_project` provides the activity' s ID (used as the string in the activity-selector), and the string `pat` is the activity creator, and the string `"set up Java project"` is the headline associated with the activity.

In a project that is enabled for ClearQuest, each activity displayed is linked to a UCM-enabled record in an associated ClearQuest user database. For more information, see *UCM-Enabled Record Types and UCM Activities* on page 26.

Finding Activities with ClearCase Dialog Boxes

ClearCase provides a graphic user interface, the File Browser, from which you can browse the VOB namespace and issue **checkout** and **checkin** commands for the elements in your view.

To start the File Browser, enter the following command:

xclearcase

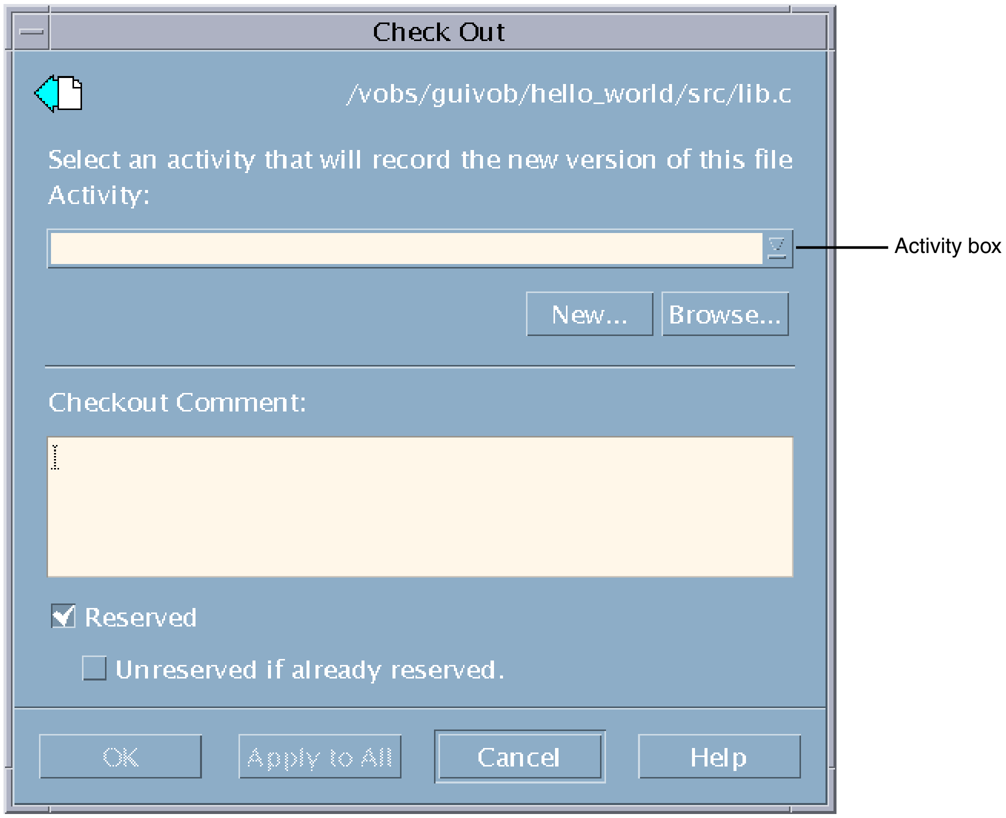
When you issue a **checkout** or **checkin** command from the File Browser, a dialog box opens for you to set an activity and optionally enter a version creation comment (Figure 16).

If the view from which you start the dialog box currently has an activity set, the Activity box shows the headline of the activity. If there is no activity set, the Activity box is blank. Use the list to select one, click **New** to create an activity to set (see *Creating UCM Activities* on page 50), or click **Browse** to find an activity to set.

If the project is enabled for ClearQuest, the dialog box displays the IDs of activities found by the UCMCustomQuery1 query. (For more information, see *UCMCustomQuery1* on page 29.)

In projects that do not use the UCM-ClearQuest integration, the list shows the IDs of UCM activity objects in the stream to which is attached the view from which you issued the command. Click **Browse** to show the activities that other team members created in your stream.

Figure 16 Activities in the Check Out Dialog Box


Check Out
/vobs/guivob/hello_world/src/lib.c
Select an activity that will record the new version of this file
Activity:
Activity box
New...
Browse...
Checkout Comment:
Reserved
Unreserved if already reserved.
OK
Apply to All
Cancel
Help

Creating UCM Activities

Your organization may have policies regarding the scope of a UCM 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 Enabled for ClearQuest

In a project that is enabled forClearQuest, an *activity* usually refers to two objects that are linked together: a UCM activity object and a UCM-enabled record in a ClearQuest user 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 records (based on a UCM-enabled record type) in a ClearQuest user database. You can create records only from the ClearQuest client.

2 When you set a view to an activity (which you can do by clicking **Actions > WorkOn** for a ClearQuest user database record), ClearCase does the following:

   - Creates an activity object in the stream to which the view is attached and
   - Links the activity object to the UCM-enabled record in ClearQuest.

   You do not create UCM activity objects directly.

To Create Activities from the ClearQuest Client

Note: You can create and use a template of default values for the Submit form in the ClearQuest user database. Such a template can save time and be helpful when you are entering multiple, related records. For information about using a template, see Help.

1 Start the ClearQuest client (see *To Start the ClearQuest Client* on page 46).

2 In the ClearQuest client window, click **Actions > New**.

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

   ClearQuest highlights in red the required fields 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 or explanation that your project manager may have included. If you have questions about what type of information a field requires, right-click the field and then click **Help**.

4 Complete the required fields in the ClearQuest user database 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 the ClearQuest user database, you must complete an additional procedure to make the activity appear in MyToDoList. (See *To Make Activities Appear in MyToDoList* on page 53.)

Creating Activities in a Project Not Enabled for ClearQuest

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 the **rmactivity** command to remove it.

To Create Activities

This procedure describes using the command line to add an activity to your development stream in a project that is not enabled for ClearQuest. (You can also use a dialog box to create a new activity; see *Finding Activities with ClearCase Dialog Boxes* on page 49). For information about adding activities to a different stream, see the **mkactivity** reference page in the *Command Reference*.

1 Access your development view. (For more information, see *Accessing Your Development View* on page 41.)

2 Enter the following command:

   cleartool mkactivity [–nset] [**–headline** *unique-string*] [*unique-*ID]

Use the **mkactivity** options as follows:

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

–**headline**
: Option to provide unique text for describing the work that you are going to do associated with the activity.

unique-string
: Provide a unique alphanumeric string used as the headline to identify the activity in the output.

If you do not provide a headline and unique identification, **mkactivity** prompts you to have a headline and an ID automatically generated. If you omit the headline but do provide an ID, **mkactivity** makes the headline equal to the ID.

It is best to provide a headline that you can later use to readily set an activity when you want to do work on your project. If you provide a headline but omit the ID, **mkactivity** prompts you to have one generated based on the string that you supplied for the headline. For example, use the following command to create an activity for changing copyright strings:

```
cleartool mkactivity -nset -headline "changing copyright strings"
Create activity with automatically generated name? [yes]
Created activity "changing_copyright_strings"
```

Use the generated ID (changing_copyright_strings) in the command line interface to refer to the activity when you are in the context of the current PVOB. For more information, see the **mkactivity** reference page in the *Command Reference*.

Assigning Activities in a Project 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 the development of an activity. 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 you begin work on an activity, the activity assignment usually does not change.

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

Note: UCM activity objects exist in a particular stream. They record the activity creator, but the creator information cannot be changed, and has no relationship to the concept of activity assignment that is used in ClearQuest.

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 the ClearQuest user database, 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 **Owner**, 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.

Setting Your View to an Activity

To work on an activity, you must set your development view to that activity. ClearCase associates all work in your view to the currently set activity, adding each version you create in your view to the activity change set.

Note: Only one user should set an activity in the same UCM view at the same time. If multiple users attempt to set an activity in the same UCM view at the same time, one user succeeds and other users encounter a set-activity retry error. We recommend that only one person have primary use (make modifications) in a UCM view and other users do read-only actions in the view.

You can set activities from the following locations:

- MyToDoList or the results of any other 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 the ClearQuest User Database

The first time you set an activity in a project that is enabled for ClearQuest, you must set it from the ClearQuest user database as described in this section. Because you cannot create UCM activity objects directly in these projects, you must use the ClearQuest client to create the activity object. For more information, see *Creating Activities in a Project Enabled for ClearQuest* on page 50.

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 56) if you need to set the activity subsequently.

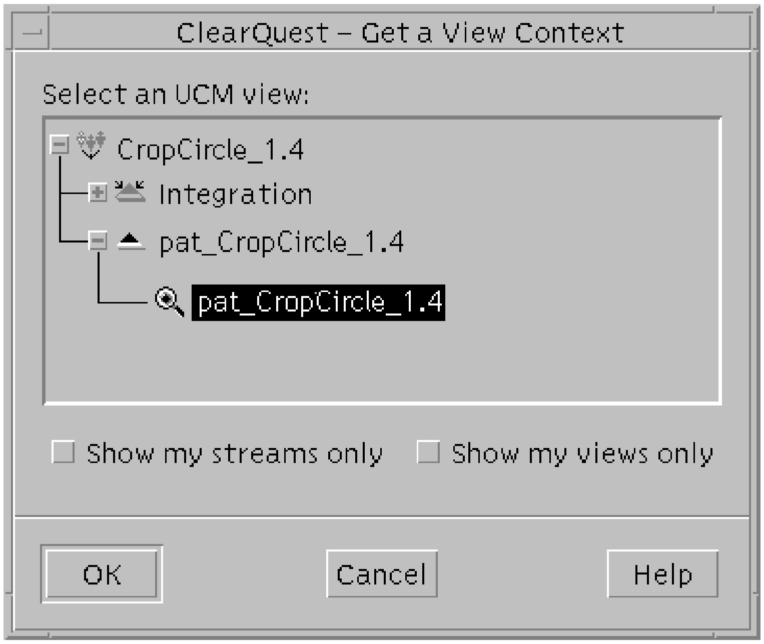
To Set Activities from the ClearQuest User Database

1. Start the ClearQuest client (see *To Start the ClearQuest Client* on page 46) and do one of the following:
   - Display MyToDoList and, in the left pane, click **Workspace > Public Queries > UCMUserQueries** and run the MyToDoList query.
   - Use another query to find the activity to set.
2. In the Query results pane, click the activity to set.
3. In the Record form pane, if it is not 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 ClearQuest - Get a View Context dialog box (see Figure 17), select Show my streams only.

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

Figure 17 Select Your View



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

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

Setting Activities with cleartool setactivity

In a project that is enabled for the UCM-ClearQuest integration, before you use **cleartool setactivity**, a UCM activity object must exist in the stream, and you must know the activity record ID. For information about creating an activity object, see *Setting Activities from the ClearQuest User Database* on page 54 and *Creating UCM Activities* on page 50.

To Set an Activity with cleartool setactivity

1. Access your development view. (For more information, see *Accessing Your Development View* on page 41.)
2. Find an activity selector to use. For information about finding an activity selector for a particular activity, see *Finding Activities with cleartool lsactivity* on page 48.
3. 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 project enabled for ClearQuest, the **setactivity** command takes an activity-selector that is an ClearQuest user database 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 (see *Unsetting Your View from an Activity*) or set another one.

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

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 headlines of activities found by the UCMCustomQuery1 query. (For more information, see *UCMCustomQuery1* on page 29.)

In projects that do not use the UCM-ClearQuest integration, the dialog boxes show the headlines of activity objects in the stream to which is attached 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

Use a command from a view context:

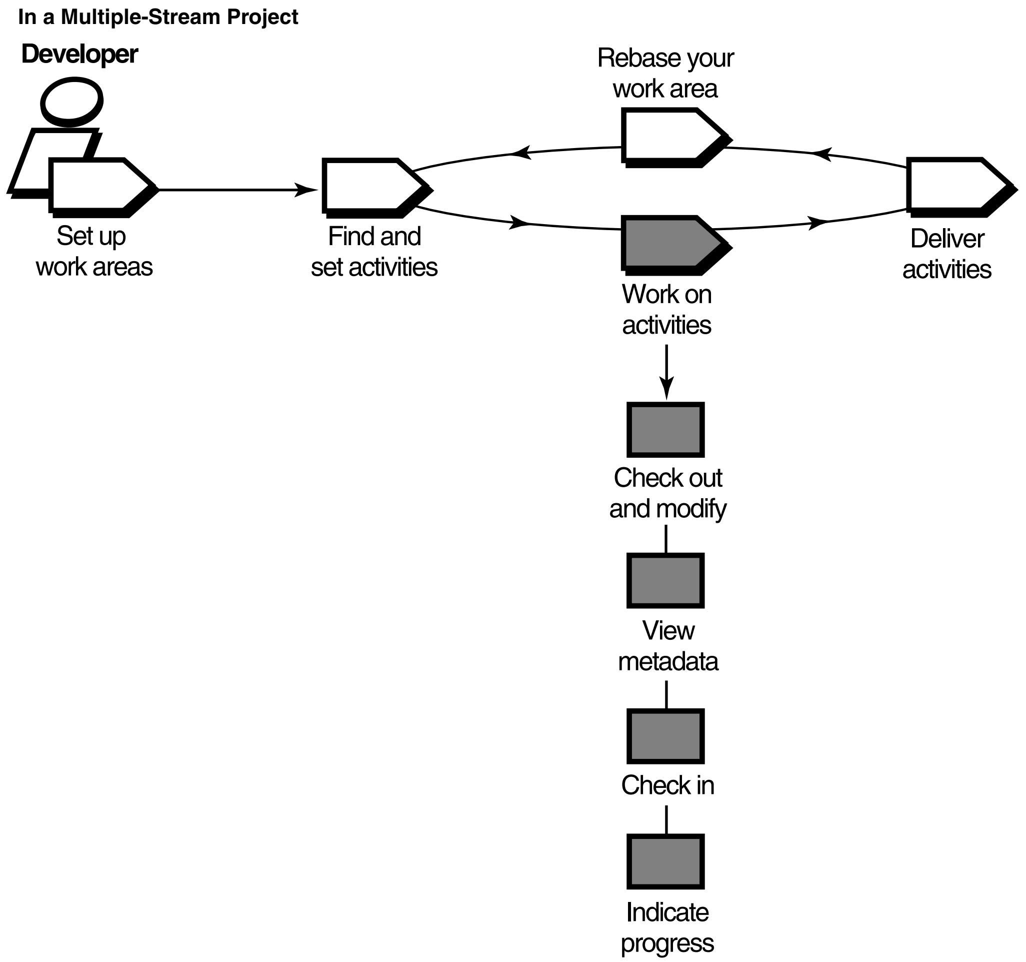
1. Access your development view. (For more information, see *Accessing Your Development View* on page 41.)
2. Enter the following command:

   cleartool lsactivity –cview

A message displays the date of creation, the activity ID, and owner of the current activity.

Working on Activities

4



Working on activities involves the following tasks:

- Checking out elements
- Working with checkouts
- Canceling checkouts
- Checking in elements

- Testing your work
- Indicating your progress

In a multiple-stream project, the work in your development stream is unavailable to other team members until you deliver it (see Chapter 5, *Delivering Activities*).

In a single-stream project, the work in your work area is available to other team members when you check in. For information about working in a single-stream project, see *Working in a Single-Stream Project* on page 77.

Checking Out Elements

To modify files and directories under source control, you must check them out. Checking out adds versions to the activity that is currently set in the view. The checked-out versions are displayed as part of the activity change set.

Placing files and directories under source control is a separate procedure described in *Adding Files and Directories to Source Control* on page 113.

> **Note**: For information about checkout treatment of VOB symbolic links in a snapshot view, see *Under the Hood: VOB Links* on page 61.

If your project uses the UCM-ClearQuest integration, your project manager may enable a policy called Perform ClearQuest Action Before Work On. This policy runs a customized ClearQuest user database script when you attempt to work on an activity. For example, your project manager may create a script that checks whether your user name matches the name in the ClearQuest user database record Owner field. Before you can work on the activity, you have to meet all criteria established by the Perform ClearQuest Action Before Work On policy.

To Check Out Elements

To check out files and directories:

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 the *Command Reference*.

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. For more information, see the **ln** reference page in the *Command Reference*.

You cannot successfully link to a component that is not part of your configuration and therefore not accessible to your VOB namespace. The link is created, but you cannot see the target that is outside your configuration. You should ask your project manager to add that component to the project and rebase to that configuration.

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 alternative 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 specific path, ClearCase considers other paths for the element as “checked out but removed.” That is, to prevent you from modifying the element from multiple paths, ClearCase executes standard checkout behavior at only one path (the one from which you entered the **checkout** command), but does not create view-private files at other paths. For information about standard checkout behavior, see the **checkout** reference page in the *Command Reference*.

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 src/util.h.

2 ClearCase removes the read-only permission from util.h in the src directory only (which is the location from which you issued the **checkout** command).

3 You modify src/util.h 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.h in the temp directory.

5 You check in temp/util.h. Even though you checked out and modified src/util.h, ClearCase does not prevent you from checking in temp/util.h; with a VOB hard link, temp/util.h is just another name for src/util.h.

6 Any changes you made to src/util.h 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.h to be hijacked (even if you change permissions), because you checked out the element in the VOB.

Resolving Checkout Problems

You can encounter problems when attempting to check out an element. Table 2 suggests ways to resolve common problems.

Table 2 Resolution of Checkout Problems (Part 1 of 2)

Problem	**Likely cause**	**Suggested resolution**
Snapshot view is out of date	You are checking out a nonexplicit version, and the snapshot view is not up-to-date with the VOB. If the element being checked out were updated, a different version would be loaded.	Update the element and then check out the updated version.
Target branch is already reserved	Another view holds a reserved checkout of the target branch.	Check out the branch as unreserved.
Non-**LATEST** version selected	You are checking out a nonexplicit version, the config spec selects a version that is not the **LATEST** on the branch, and there is no **mkbranch** rule. This can occur because a label or a time rule selects elements.	Check out the **LATEST** version.
File is hijacked and does not correspond to the selected version	The file is in the `hijacked/nocheckout` state or the checkout explicitly specifies a different version.	Check out the hijacked file; or merge the hijacked version and the selected version, and use the merge result as the checkout data.

Table 2 Resolution of Checkout Problems (Part 2 of 2)

Problem	Likely cause	Suggested resolution
A **mkbranch** rule in the view config spec fails and automatic branch creation fails	The branch exists. The config spec does not select the branch before specifying that the branch should be created (as it should). Or, for a dynamic view, a time rule prevents it from seeing the branch. Or, for a snapshot view, the branch was created after the view was lasted updated.	Check out the **LATEST** version on the branch; for an out-of-date snapshot view, this means an update of the element followed by a checkout of **LATEST**.

Working with Checkouts

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

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

Viewing Element History

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

To View Element History

In a view, enter this command:

cleartool lshistory –graphical *path*

You can use this command from a snapshot view whether or not the element specified by *path* 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 that another developer is using?

To Compare with a Predecessor

In a view, enter this command:

cleartool diff –graphical –predecessor *path*

The **–diff_format** option causes both the headers and differences to be reported in the style of the UNIX **diff** utility, writing a list of the changes necessary to convert the first file being compared into the second file.

To Compare with a Version Other Than the Predecessor

1. In a shell, enter this command:

   cleartool lsvtree –graphical *path*
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 to use the command line:

1. Use **cleartool lsvtree** to list element versions.
2. Use the **cleartool diff** command with a *version-extended path*. For example, to compare the current version of prog.c with version 4**:**

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

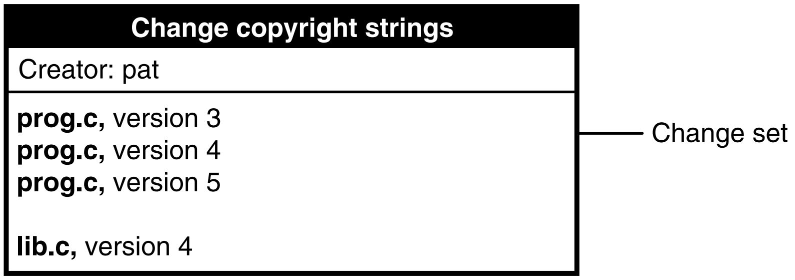
You can use the **lsvtree** and **diff** commands from a snapshot view whether or not the element specified by *path* is loaded into the view. For more information, see the **diff** and **pathnames_ccase** reference pages in the *Command Reference*.

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 specific activity. For example, for the activity **Change copyright strings** you check in a version of prog.c and realize later that you missed a string in the file. You then set your view to the activity **Change copyright strings**, check out prog.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 change set with the version that immediately precedes the change set. In Figure 18, the change set predecessor of prog.c is version 2. You can compare any version of prog.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 properties sheet.
   a. Type **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 properties sheet, click **Change Set**.
3. Right-click the version you want to compare, and, on 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 the *Command Reference*.

Viewing the Change Set of an Activity

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

To View the Change Set of an Activity from ClearQuest

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

  a Start the Clearquest client (see *To Start the ClearQuest Client* on page 46).

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

  ClearQuest displays MyToDoList list in the Query builder pane. (Figure 14 on page 47.)

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 type for the record.

To View the Change Set of an Activity 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

The **lsactivity** output provides the UCM activity ID and versions in the change set. For example:

```
% cleartool lsactivity –long –cact
activity SAMPL051111
05-Aug-01.09:14:17 by Pat (pat.user@bread)
"Created for the cropcircle_1.4 project on 09/17/00 13:25:24."
owner: pat
group: user
stream: pat_1.4_CropCircle
title: changing_copyright_strings
change set versions:
/guivob/prog.c@@/main/stream990805.090736/5
/guivob/spices.c@@/main/stream990805.090736/2
/guivob/onions.c@@/main/stream990805.090736/9
/guivob/tomatoes.c@@/main/stream990805.090736/1
/guivob/flour.c@@/main/stream990805.090736/3
```

In a project that is enabled for ClearQuest, the ClearQuest record ID is the same as the UCM activity ID.

To View the Change Set of an Activity 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 **Change Set**.

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 Within a ClearCase Context

1. Display an activity change set. (For more information, see *Viewing the Change Set of an Activity* on page 67.)
2. On the **Change Set** tab, select one or more elements.
3. Right-click a selected element and click **Move to Activity**.

To Move Versions Within a ClearQuest User Database Context

1. Display an activity change set in the ClearQuest user database. For more information, see *Viewing the Change Set of an Activity* on page 67.
2. Select an element from the change set list.
3. Right-click the element and click **Move to Activity**.
4. Follow the ClearCase prompts.

Canceling Checkouts

If you check out a file but do not 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 *path*

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, type **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 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 path of the new element) to the VOB lost+found directory under names of the form: *element-name.UUID*.

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

```
cleartool: Warning: Object "prog.c" no longer referenced.
cleartool: Warning: Moving object to vob lost+found directory as
"prog.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 about starting the Update Tool, see 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 permanently delete an element in the lost+found directory, take note of the name of the orphaned element and use this command:

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

For example, from a dynamic view:

cleartool rmelem /guivob/lost+found/prog.c.5f6815a0a2ce11cca54708006906af65

From a snapshot view:

cd ~/pat_v1.4_cropcircle_sv
cleartool rmelem guivob/lost+found/prog.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 parent directory of the element 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.

Canceling the Checkout of a Deleted File

If you check out a file element in a dynamic view and then delete that file from your view, you should cancel the checkout.

1. In the view in which the file is checked out, use the following command:

   ```
   cleartool lscheckout -me -cview -all -short
   ```

2. In the list, note the path for the deleted file.

3. Use the following command with the path for the deleted file to cancel the checkout:

   `cleartool uncheckout` *path*

   You see the Checkout cancelled message for the file.

In a snapshot view, deleting checked-out elements executes an update operation on the affected elements, after which you cannot cancel the checkout.

Checking In Elements

Check in an element 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 project team until you deliver them.

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 either 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 the *Command Reference*.

Try to enter meaningful comments. Comments are a valuable part of keeping track of your work. You can search for text in comments to identify a specific version.

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 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 the directory tree of your view.)

Testing Your Work

To reduce the amount of merging needed when you deliver your work, you should build and test the modifications in your work area. If you work in a multiple-stream project, do this especially after you rebase the stream and before you deliver your work. If you work in a single-stream project, do this before you check in your changes.

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

For more information, see *Building Software* and the **clearmake** reference page in the *Command Reference*.

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
- Reassigning an activity
- Deleting an activity

You may need to reassign or delete an activity whether or not your project is enabled for ClearQuest.

Modifying Information in a ClearQuest User Database

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 the symptoms of a defect. Adding or modifying such information does not change the activity state or have any effect on the corresponding activity in your stream.

For information about adding information to multiple activities, see Help for ClearQuest.

To Modify ClearQuest User Database Information

1. Use a query to find the activity. For example, to use MyToDoList:
   - **a** Start the ClearQuest client (see *To Start the ClearQuest Client* on page 46).
   - **b** In the left pane of the client window, 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**.
4. In the Record form, make your changes. (Changing the Owner field reassigns the activity. For information about reassigning an activity, see *Reassigning Activities* on page 75.)
5. To save your changes, click **Apply**. To clear changes you made without saving them, click **Revert**.

Closing an Activity

If you work in a multiple-stream project, your project manager may have set up a policy for your project that closes activities in ClearQuest when you deliver them.

If these policies are 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 appear in MyCompletedWork, but they do not appear on MyToDoList. However, completed activities remain in your stream and appear in your stream properties even after you deliver them. For information on queries, see *Queries* on page 28.

If you work in a single-stream project that is enabled for ClearQuest, your project manager can set policies to have you perform other actions to finish activities (see *Finishing Your Activity* on page 79).

To Close a ClearQuest Activity

Your project manager may have customized your ClearQuest user 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. Start the ClearQuest client (see *To Start the ClearQuest Client* on page 46).
   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** 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**.
4. Click **Apply**.

Using Schema-Specific Actions

Your ClearQuest user database schema, which your project manager can customize (see *The UCM-ClearQuest Schema* on page 25), 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 form pane of the record.

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 change set to another activity in the same stream (see *To Move Versions Within a ClearCase Context* on page 68) or to an activity in a different stream (see *Moving Work from a Development Stream* on page 121).
- 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 properties sheet, displays the activity 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* on page 121.)

Reassigning Activities

You can assign an activity on MyToDoList to another project team member by changing the activity owner, unless your project manager has enabled the UCM policy Perform ClearQuest Action 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 UCM 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 owner in ClearQuest, and then do one of the following:

- If you have not created versions for the activity (that is, the activity change set is empty), you can remove the activity in your development stream. (See *To Delete a UCM Activity* on page 76.) 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 **Change Set** list, which is available from the ClearQuest Record form pane or from the properties sheet of the activity, displays its change set.
- If the activity change set is not empty, the new owner can create and work in a view attached to the stream containing the activity. For information about attaching a new view to an existing stream, see Help.

To Change an Activity Owner

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

  a Start the ClearQuest client (see *To Start the ClearQuest Client* on page 46).

  b In the left pane of the client window, 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 you can delete ClearQuest records of any specific record type.

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

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

To Delete a UCM 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 rmver** to remove the versions from the VOB. 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 graphic user interface.

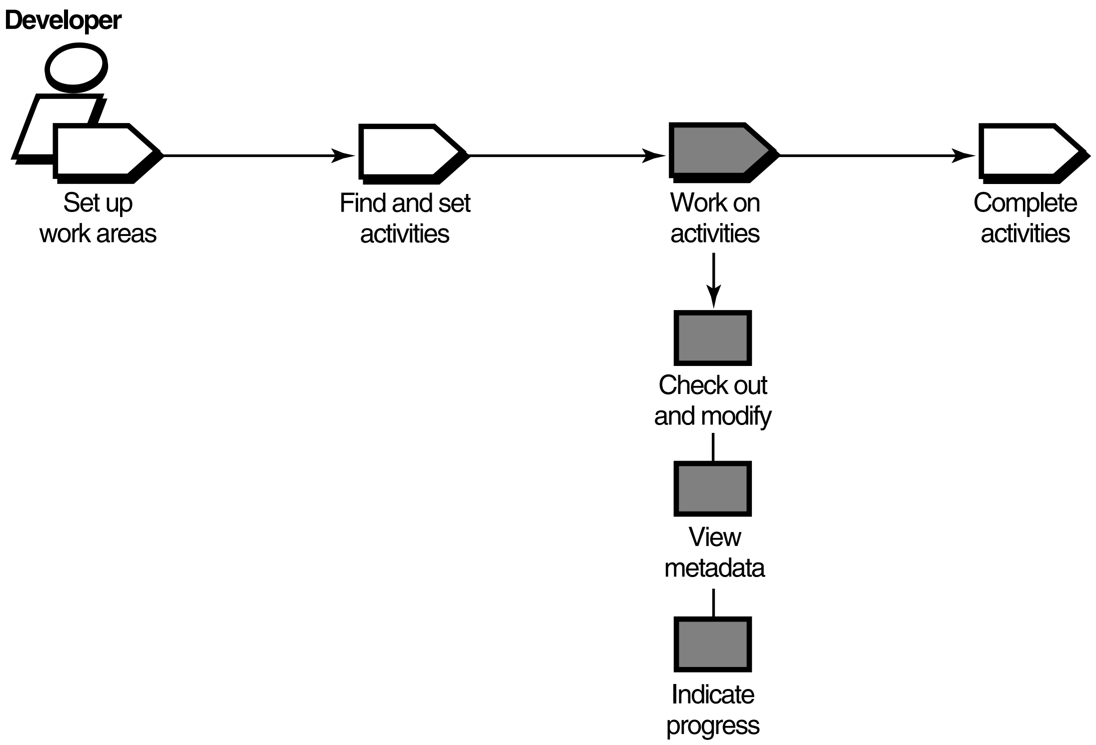
To delete an activity from a shell, use **cleartool rmactivity**. For more information, see the **rmactivity** reference page in the *Command Reference*.

To Delete a ClearQuest Activity

1. Use a query to find the activity. For example, to use MyToDoList:
   a. Start the ClearQuest client (see *To Start the ClearQuest Client* on page 46).
   b. In the left pane of the client window, click **Workspace > Public Queries > UCMUserQueries** and run the MyToDoList query.

      ClearQuest displays the records in the **Result set** tab 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 UCM activity (and the activity change set is empty), ClearQuest deletes the activity in the stream as well.
4. Click **Apply**.

Working in a Single-Stream Project



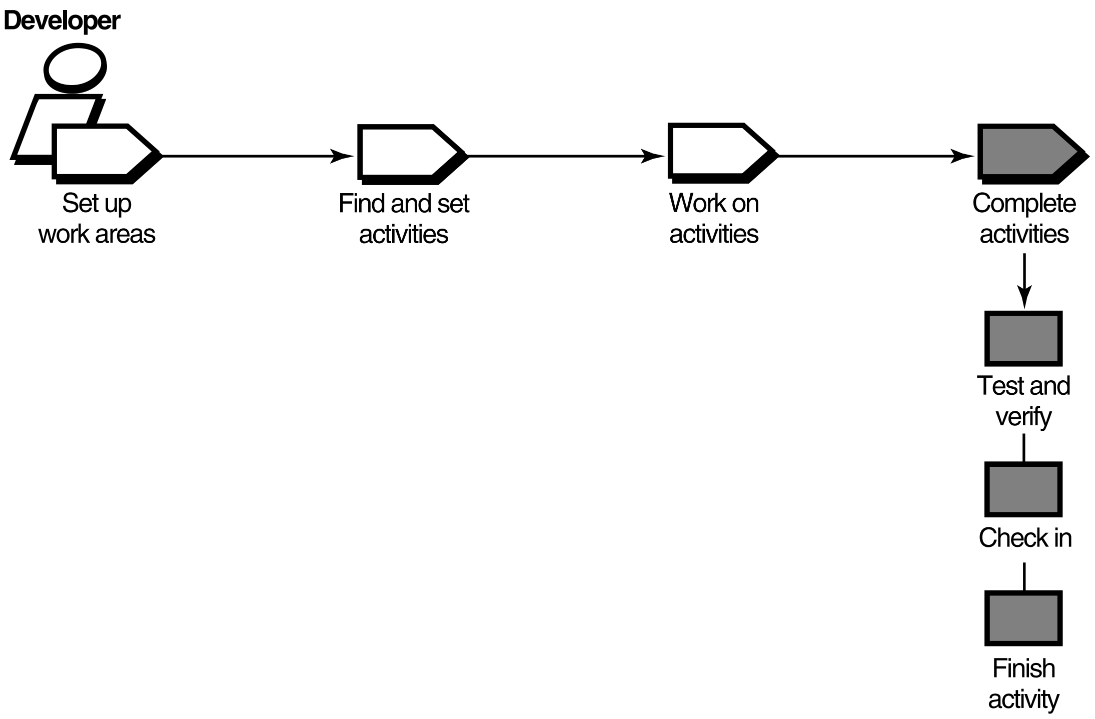
Working on activities in single-stream project involves the following tasks:

- Checking out elements (see *Checking Out Elements* on page 60)
- Working with checkouts (see *Working with Checkouts* on page 64)
- Canceling checkouts (see *Canceling Checkouts* on page 68)
- Indicating your progress (see *Indicating Your Progress* on page 72)

Tasks While Working in a Single-Stream Project

The tasks for working in a single-stream project are similar to those one does in a multiple-stream project. However, you do not check in your work until you are ready to complete activities in the project (see *Completing Activities in a Single-Stream Project*).

Completing Activities in a Single-Stream Project



In a single-stream project, when you are ready to make work in the project integration stream available to the project team (for example, to contribute to official project builds), you complete your activity. Completing your activity involves the following tasks:

- Testing your work (see *Testing Your Work* on page 72)
- Checking in elements (see *Checking In Elements* on page 71)

- Finishing your activity (if your project is enabled for ClearQuest)

In a single-stream project, you start work from the latest version in your integration stream, which you access through your integration view. When you check in your work, those versions are the latest versions in the integration stream. Team members working in dynamic views see your work immediately. Team members working in snapshot views see your work after the update their views.

Because the work is available to others on the project when you check in, you must test your work *before* you check in. Because checking in makes your work available to others, there is no need to deliver or rebase.

Also, if your project is enabled for ClearQuest, you must finish activities (see *Finishing Your Activity* on page 79).

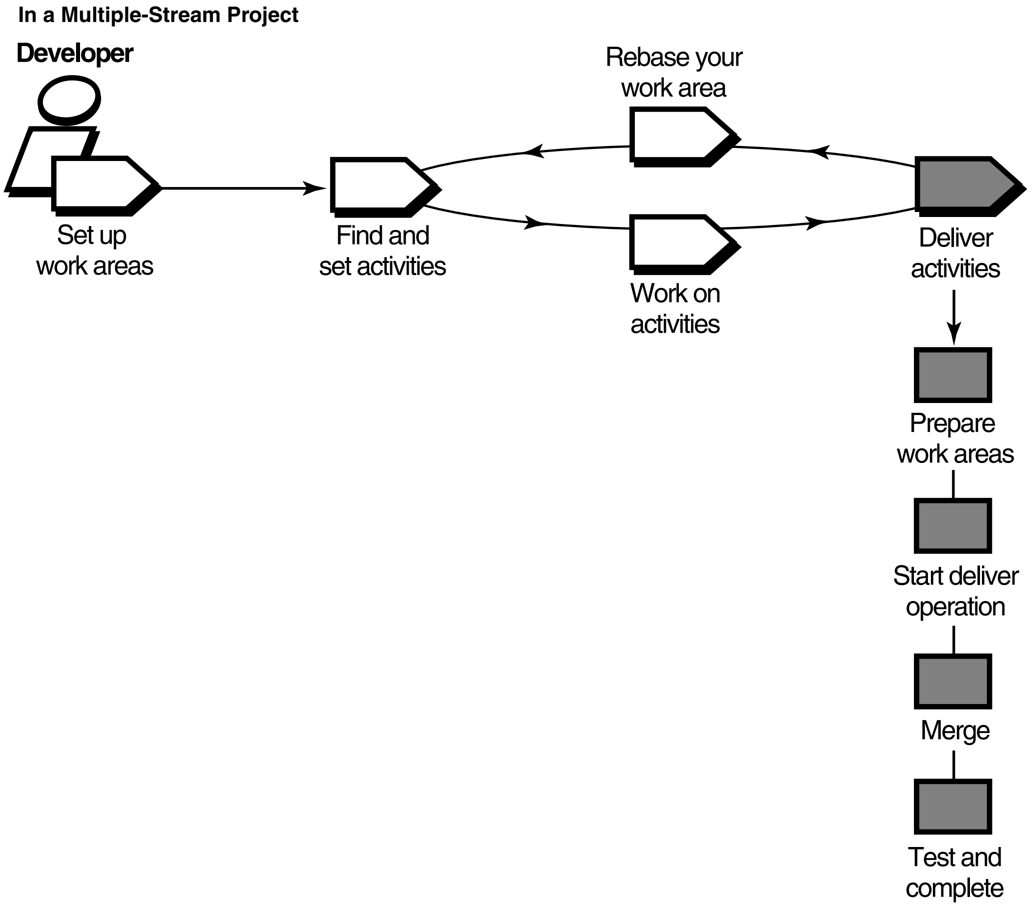
Finishing Your Activity

In a single-stream project that is enabled for ClearQuest, you must indicate the completion of your work by finishing the activity. If your project uses the UCM-ClearQuest integration, your project manager may enable a policy that runs a customized ClearQuest user database script when you finish an activity. For example, your project manager may create a script that checks whether your user name matches the name in the ClearQuest user database record Owner field or asks you to enter other information. When you successfully complete the script, another policy can automatically transition your activity to the complete state in the ClearQuest user database.

To finish your activity, in Project Explorer, right-click your activity and click **Finish Activity**, or, in the File Browser, click **Project > Finish activity**.

If any versions in the activity are checked out, the Check Out dialog box appears for you to check in your changes. After you check in, this transitions the activity to the Complete state type (see *State Types and State Transitions in a Schema Enabled for UCM* on page 32).

Delivering Activities 5



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

- Preparing your work areas
- Starting the deliver operation
- Selecting activities and baselines

- Merging versions
- Testing your work
- Completing the deliver operation

Note: If your project uses Rational ClearCase MultiSite to share source data with developers in other geographical locations and if the project 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 101.

Preparing Your Work Areas

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

- If your project integrator 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 about 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

Rational ClearCase delivers only checked-in changes. In addition, your project manager may have set a project policy that requires you to 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 the *Command Reference*.

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 specific file, check it in by typing this command:

  cleartool checkin *filename*

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

  cleartool uncheckout *filename*

  During the cancel checkout operation, you can choose to save your changes in a *view-private file* (see *Canceling Checkouts* on page 68).

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 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 to the target stream.

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 target stream.

To Update a Snapshot View and Resolve Hijacked Files

1 Enter the following command:

  cleartool update –graphical *snapshot-view-path*

2 In the Update dialog box, click **Advanced** and select an option for resolving *hijacked files*. (For information about each option, see the Help.)

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, see *Using the Update Tool* on page 133.

Starting the Deliver Operation

In UCM, you can deliver your work as activities or as baselines. Typically, in a single project environment, you work on activities and deliver your completed activities. Some UCM environments manage parallel work on the same components. The project integrators in such environments deliver baselines to the project integration stream to incorporate changes in components. For information about delivering baselines, see *Managing Software Projects*. This section covers delivering activities.

You can deliver your work to either the default target (the parent stream of your development stream) or a nondefault target. The policies on the project and on the target stream determine what target is allowed. When you start the deliver operation, the target stream policy settings are checked to determine whether the deliver operation is allowed. The project policy determines the settings for streams in the project; the project policy may defer the setting to the target stream policy. For information about project and stream policies, see *Managing Software Projects*.

If your organization uses ClearCase MultiSite and your site does not master the target of your delivery, you can only post your work (see *MultiSite: Posting Work to Deliver* on page 101).

Deliveries and the UCM-ClearQuest Integration

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

If your organization uses ClearCase MultiSite and the UCM-ClearQuest integration, the activity object and its ClearQuest user database record must be mastered locally before you can work on, set, or change an activity.

Delivering to the Default Target

If you deliver your work to the default target, the target of the delivery is your parent stream (see *Your Parent Stream* on page 20).

To Start the Deliver Operation to the Default Target

Start the delivery from either the command line or Project Explorer.

From the Command Line with a View Context

1 Access your development view. (For more information, see *Accessing Your Development View* on page 41.)

2 Enter the following command:

cleartool deliver –graphical

The **Deliver Preview** dialog box opens for you to select activities.

From the Command Line Without a View Context

1 From a shell prompt, enter the following command:

clearmrgman –deliver

2 UCM Deliver starts. In the Deliver from Stream dialog box (Figure 19), navigate to your project and stream.

3 Select your stream and click **OK**.

The Deliver Preview dialog box opens for you to select activities.

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.

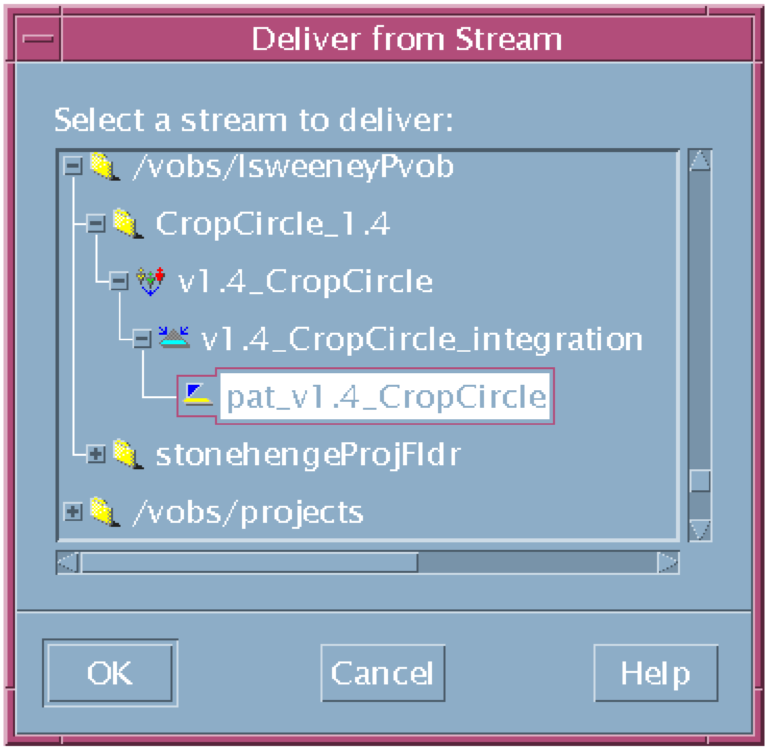
2 Click **+** next to any folder, next to your project, and next to your parent stream.

3 Click your development stream.

4 Right-click to display the shortcut menu. Then select **Deliver from Stream > To Default**.

When the delivery starts, you have to specify what to deliver (see *Selecting Activities* on page 87).

Figure 19 Select Your Stream



Delivering to a Nondefault Target

If you are delivering your work to a nondefault target, you can choose a stream in the same project (intraproject delivery) or a stream in another project (interproject delivery). The policies on the target stream (intraproject) or on the target project (if you are doing an interproject delivery) determine what target is allowed. You can attempt to deliver from any stream in a project to any other stream in the same project, including the integration stream. Whether the operation can proceed depends on the applicable policy settings. See *Managing Software Projects* for details on policies.

To Start the Deliver Operation to a Nondefault Target

Use Project Explorer to deliver your work to a nondefault target.

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 Navigate to and click **+** next to your project.

3 Navigate to and click your development stream.

4 Right-click to display the shortcut menu. Then click **Deliver from Stream > To Alternate Target**.

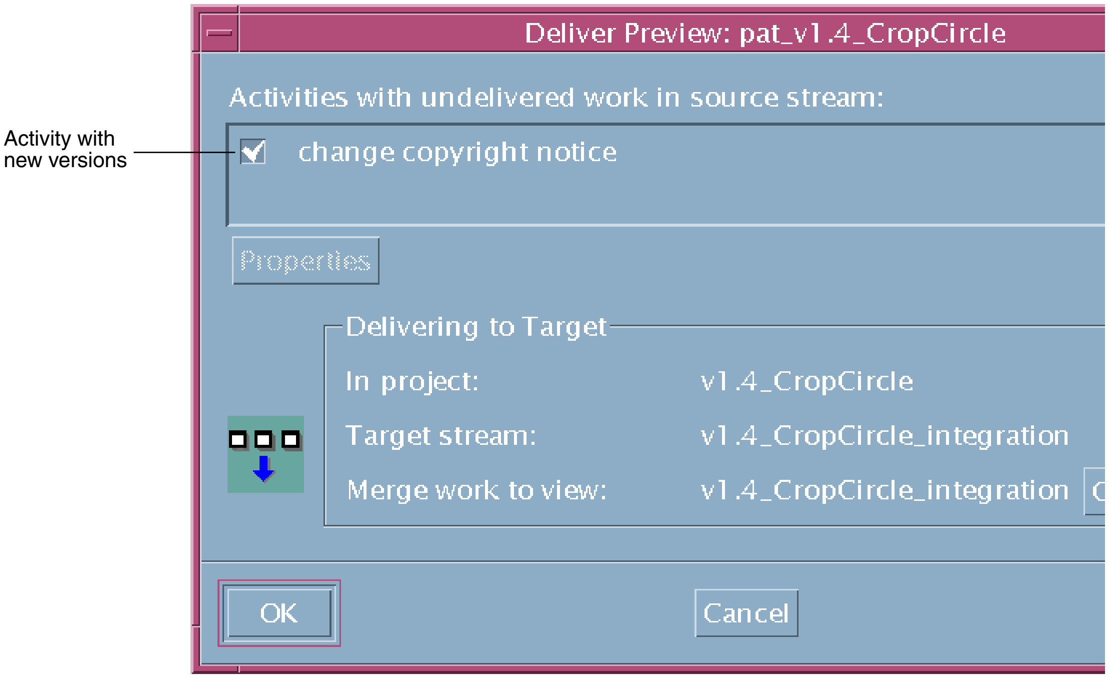
5 In the Deliver from Stream dialog box, select the target stream and click **OK**.

The Deliver Preview dialog box appears. Now you select activities to deliver (see *Selecting Activities* on page 87).

Selecting Activities

After you start the deliver operation and, optionally, specify the target, ClearCase opens another Deliver 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


Activity with new versions
Deliver Preview: pat_v1.4_CropCircle
Properties
OK
Cancel

Activities with undelivered work in source stream:

✓ change copyright notice

Delivering to Target

In project: v1.4_CropCircle

Target stream: v1.4_CropCircle_integration

Merge work to view: v1.4_CropCircle_integration

The Delivering to Target box shows the target context of your delivery.

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 (see *Activity Dependencies in the Deliver Operation*) and baseline considerations (see *Baseline Considerations in the Deliver Operation* on page 89), ClearCase prompts you to change your selection.

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

Controlling Merge Behavior

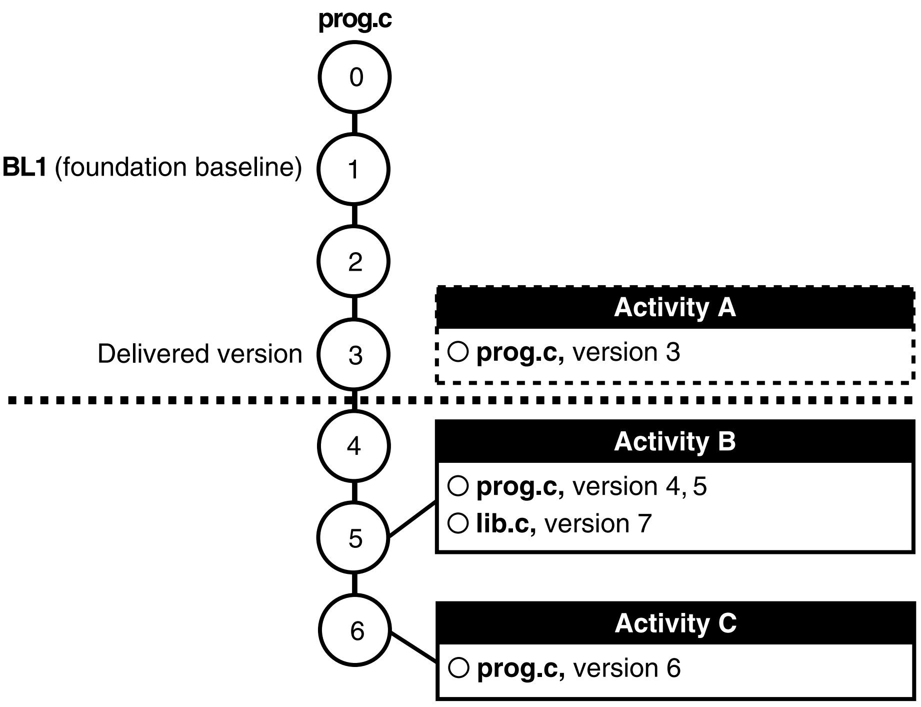
ClearCase tries to do the merge automatically. However, you can perform the merge on each element manually. To handle such a need, in the Deliver Preview dialog box, select **Merge elements graphically** to turn off all automated merging. If you select this option, the merge operation prompts you for confirmation before proceeding with each change.

Activity Dependencies in the Deliver Operation

For any given element in an activity 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 *foundation baseline* (whichever is later). For example, in Figure 21, because version 3 of prog.c was delivered in a previous deliver operation, the dependencies between Activities B and C, which contain versions of prog.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 prog.c.
- Because Activity B also contains versions of lib.c, you may be required to deliver other activities to satisfy dependencies for lib.c. For example, if Activity D (not shown in Figure 21) contained an undelivered version 6 of lib.c, delivering Activity B would require you to deliver Activity D as well.

Figure 21 Activity Dependencies

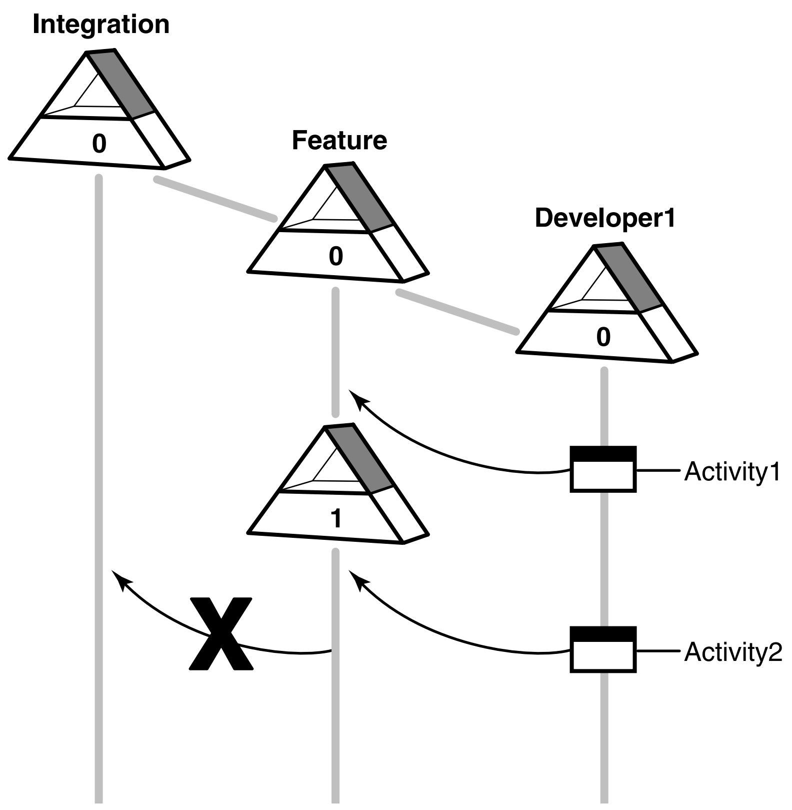


Baseline Considerations in the Deliver Operation

As with dependencies between activities in the deliver operation (see *Activity Dependencies in the Deliver Operation* on page 88), in some situations, baselines can affect the deliver operation. If the foundation baseline of the source stream is different than that of the target stream, the deliver operation includes activities in the foundation baseline plus all activities created after the foundation baseline. This prevents you from delivering only selected changes.

For example, the Feature stream in Figure 22 is configured with baseline 0. The developer working in the Developer1 stream delivers Activity1 to the Feature stream.

Figure 22 Baseline Consideration in Deliver

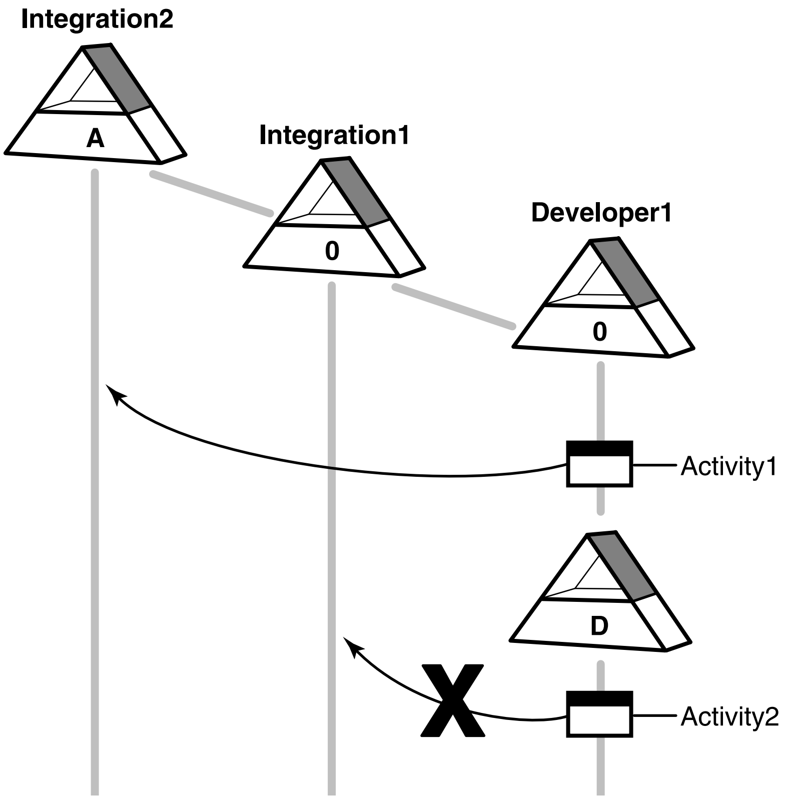


The owner of the Feature stream explicitly creates baseline 1 in that stream. Activity Activity1 is included in baseline 1. The developer in the Developer1 stream continues working on a new activity and delivers Activity2 to the Feature stream.

If the integrator of the Feature stream delivers work to the Integration stream, the delivery includes activity Activity2 plus the activities included in baseline 1. The integrator cannot deliver only Activity2.

Another baseline consideration occurs when you deliver to an alternative target stream, as shown in Figure 23.

Figure 23 Baseline Consideration in Alternative Target Deliver



The Developer1 stream in one project delivers Activity1 to an alternative target, the integration stream (Integration2) in another project. As an intermediate step, deliver creates a hidden baseline D (referred to as a deliver baseline) in the Developer1 stream. If the Developer1 stream next tries to deliver only Activity2 to the integration stream (Integration1) in its project, deliver requires that the changes associated with the deliver baseline (Activity1) be delivered. In this case, you cannot deliver only activity Activity2, but must also deliver activity Activity1 which is included in the deliver baseline.

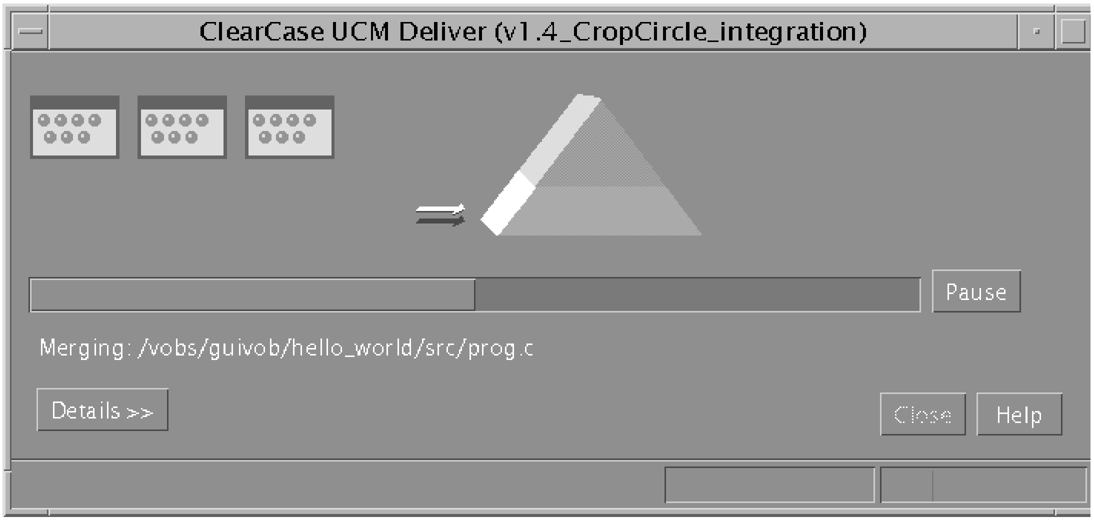
Merging Versions

Note: If your project uses Rational ClearCase MultiSite to share source data with developers in other geographical locations, and if the project streams are mastered at a different site, you do not merge versions. For more information, see *MultiSite: Posting Work to Deliver* on page 101.

After selecting activities, click **OK** in the Deliver 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 ClearCase UCM Deliver window.

For each new version delivered to the integration view, ClearCase merges all nonconflicting differences. To see progress of the merge, click **Details** and an expansion box lists the versions in the merge (Figure 24).

Figure 24 Deliver Progress

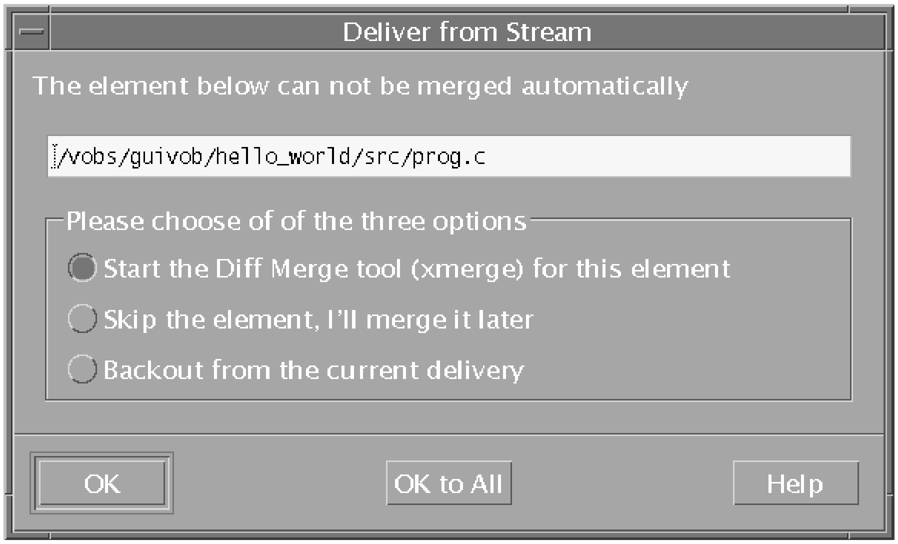


Merging: /vobs/guivob/hello_world/src/prog.c

As it merges versions, ClearCase changes the value in the Merged column of the Details expansion box 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 shows **No**. For each version that has conflicts, a Deliver from Stream dialog box appears (Figure 25), giving options to handle the version.

Figure 25 Merge Conflict Options


Deliver from Stream

The element below can not be merged automatically

/vobs/guivob/hello_world/src/prog.c

Please choose of of the three options

[● Start the Diff Merge tool (xmerge) for this element]

○ Skip the element, I'll merge it later

○ Backout from the current delivery

To merge a version that contains merge conflicts, select the version and click **Start the Diff Merge tool (xmerge) for this element** and click **OK**. To apply the same option to all remaining files with conflicts, click **OK to All**. ClearCase starts Diff Merge, a tool that helps you resolve conflicting differences. (**clearmrgman** is the main window of the Diff Merge, a ClearCase tool used for merging files and directories.)

Under the Hood: Concurrent Deliver Operations

ClearCase allows developers to deliver work concurrently. However, if another team member's deliver operation has checked out an 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 target stream when needed. If it encounters a reserved directory checkout in the target 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 it encounters an element that it cannot check out (for example because another team member's deliver operation has checked out the element), it skips the element and attempts to check out the remaining elements.

   If it 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. For more information, see *To Undo a Deliver Operation* on page 99.

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 the integration activity and adds versions to it; you do not need to create or maintain it. You may want to view the properties of an integration activity to see exactly which versions were merged and created during a deliver or rebase operation. You can view integration activity properties from the Project Explorer; the activity name is of the form *deliver.stream-name.date-stamp*.

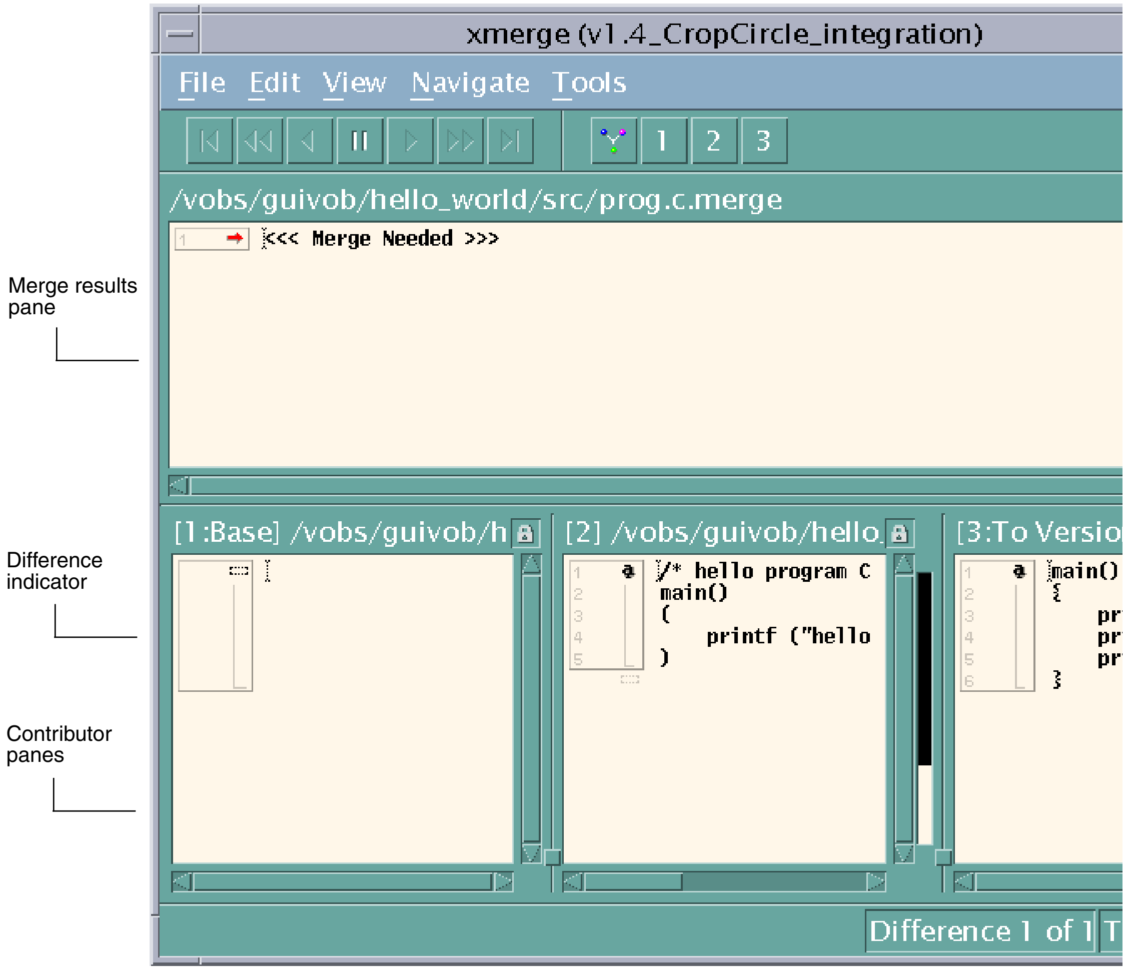
If your project uses the UCM-ClearQuest integration, ClearCase links a UCM-enabled record based on the UCMUtilityActivity record type with the UCM activity. During the deliver operation or the rebase operation, the activity is in a state based on the Active state type. After the operation, ClearCase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

As an intermediate step in the delivery, 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*.

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 26).

Figure 26 Diff Merge Window



```
<<< Merge Needed >>>
```

```
/* hello program C
main()
(
      printf ("hello
)
```

```
main()
{
      pr
      pr
      pr
}
```

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, in Figure 26, 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 about 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.

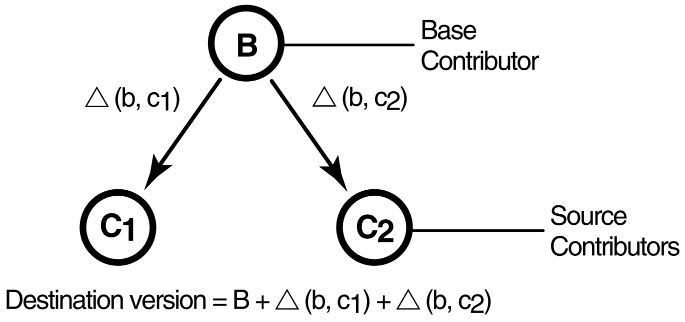
Under the Hood: How ClearCase Merges Files and Directories

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

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

As illustrated in Figure 27, ClearCase starts with the base contributor and adds changes that are present in the source contributors.

Figure 27 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 Diff Merge so you can choose which change to add to the destination version.

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 target 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 the ClearCase system not to attempt merging the file. For more information, see *Managing Software Projects*.

To Stop and Resume the Deliver Operation

You can stop the deliver operation temporarily by clicking **Cancel** in the clearmrgman dialog box.

To resume the deliver operation, type **cleartool deliver –resume –graphical** from your development view.

Delivering Changes to a Read-Only Component

Deliver cannot allow changes to a read-only component. This situation can result from someone changing the component modifiability in your project. What actions you take depend on whether you are doing an intraproject or an interproject deliver.

For Intraproject Deliver

If you are performing an intraproject deliver for a component that was previously modifiable, you receive a warning message. Decide what can be delivered from your stream. If the changes in the other components are independent of the changes in the read-only component, move the versions from the read-only component into a new activity. Deliver all activities in your stream except the new one.

If any of the changes in the other components are dependent on changes to the read-only component, decide whether those versions can be separated into a new activity. This may involve not delivering a subset of activities in your development stream.

If there are dependencies among the versions, your stream may no longer be deliverable to its default target. Decide whether there is a project to which you can deliver the changes. The project integrator may have to create a new development stream in this project and selectively find and merge the changes (findmerge -insert) from the versions into the new stream. Deliver cannot be used.

As a last resort for an intraproject deliver, you can remove all the versions (**rmver**) that should not have been made. Also, remove all changes dependent on that component. You must use **rmver -xhlink** to remove the versions from the change set. You may need to manually recode some changes to remove the dependence on the changes to the read-only component.

For Interproject Deliver

If you are performing an interproject deliver, you can also receive a warning message. The project from which you are delivering and the target project can have differences in component modifiability at any time. You have the following choices:

- Set the policies on the target stream and target project so that deliver can ignore the changes in both the read-only component and in other components that are dependent on the changes to the read-only component.
- Isolate the changes in the read-only component and changes in other components that are dependent on the changes to the read-only component. Deliver only the activities that contain changes to modifiable components.

Testing Your Work

At this point in the delivery, when all versions are merged, the deliver operation is still in progress. The destination versions (the new versions resulting from the merges) are checked out and visible only in your integration view.

Your integration view shows your delivered work and the work that other team members have delivered. To make sure that your work is compatible with other delivered work, we recommend that you test the work in your integration view before you complete the deliver operation.

For information about accessing your integration view, see *Accessing a Dynamic View* on page 42 or *Accessing a Snapshot View* on page 42.

Your organization can use **clearmake** as ClearCase build tools; however, the *build auditing* and *build avoidance* features are available only from dynamic views. For more information about **clearmake**, see *Building Software* and the reference page in the *Command Reference*.

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 During Testing

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

Undoing a Deliver Operation

At any time before you complete the deliver operation, you can undo changes made as a result of the deliver operation. This undo procedure is valid only if you have started a deliver operation, have not checked in modifications in the target view, and have not completed the deliver operation.

To Undo a Deliver Operation

1. If you started Diff Merge, 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 –xhlink** to remove the versions you checked in before canceling the operation. Because **rmver** erases part of your organizational 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 –**xhlink** option. For more information, see the **rmver** reference page in the *Command Reference*.

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.

> **Note:** If a view in which a deliver operation is in progress has been deleted, you are prevented from completing (and resuming or canceling) the deliver. This can block all other deliveries for the project. To complete the deliver, use the **deliver** command as described in the *Command Reference*. For more information, see the **deliver** reference page.

To Complete a Deliver Operation

Open the Deliver Progress dialog box (Figure 24 on page 92) by entering the following command from your development view:

```
cleartool  deliver  –complete –graphical
```

The Deliver Progress dialog box can remain open during the deliver completion to report status as ClearCase attempts to place the deliver operation in the complete state.

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 target stream.
2. If your project is enabled for ClearQuest and your project manager has enabled a policy, other actions occur.
   - If the Perform ClearQuest Action After Delivery policy is enabled, a ClearQuest user database script executes. If the script returns an error, ClearCase stops the deliver operation. You must correct the error before the deliver can continue.
   - If the Transition to Complete After Delivery policy is enabled, the delivered activities move to a Complete-type state.
3. It changes the state of the operation to Complete.

MultiSite: Posting Work to Deliver

If your organization uses ClearCase 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 site can modify them.

If your site does not master a stream, you cannot complete deliver operations to that stream. UCM provides a variation of the deliver operation called a *remote deliver*. You must post your work to deliver. If the target stream is mastered by the local PVOB replica, the deliver operation is local.

After you deliver your work and the delivery is in the posted state, the project integrator at the site that masters the target stream completes the remote deliver operation.

To Deliver to a Nonmastered Target Stream

To make your work on a remote stream available in the target stream:

1 Prepare your work areas (see *Preparing Your Work Areas* on page 82).

2 Start the deliver operation (see *Starting the Deliver Operation* on page 84).

   In a remote deliver operation, the source stream must be a development stream.

3 Select activities (see *Selecting Activities* on page 87).

4 Deliver the work so that it can be posted.

The deliver operation determines whether your development and target streams are mastered at different replicas. If they are mastered at different replicas, a remote deliver operation is put into effect. ClearCase changes mastership of the development stream object to match that of the target stream object, and it notifies you that it has assigned the development stream the posted status.

When your development stream has the posted status, notify the project integrator at the site that masters the target stream replica. The project integrator can then find posted deliver operations and either 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. The delivery completes when the posted deliver operation is merged with the target stream.

Remote Deliveries with the UCM-ClearQuest Integration

In a project that uses the UCM-ClearQuest integration, the project manager may set the Transition to Complete After Delivery policy on a project (see *Deliveries and the UCM-ClearQuest Integration* on page 84). This policy transitions activities in an associated ClearQuest user database to a Complete state when the deliver operation completes successfully. For this policy to work in a MultiSite environment, all activities being delivered must be mastered by the same PVOB replica that masters the target stream.

In a MultiSite environment that uses the UCM-ClearQuest integration, the project manager sets the Transfer ClearQuest Mastership Before Delivery policy on a project to have the software change the mastership of all activities being delivered. If the deliver operation is local and all activities being delivered are not mastered locally, the deliver operation fails.

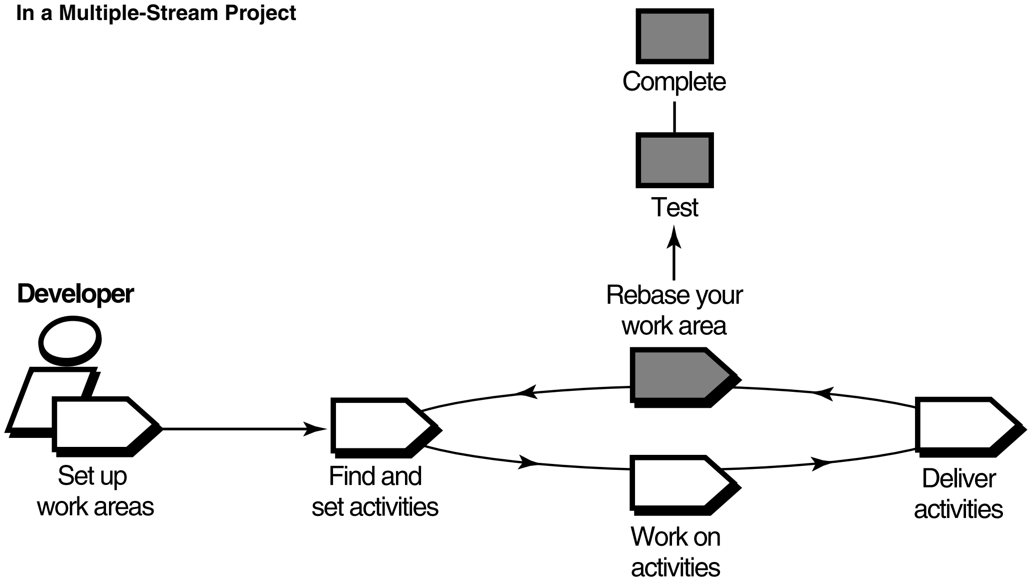
For a remote deliver operation, the Transfer ClearQuest Mastership Before Delivery policy causes the following behavior:

- If all activities are mastered by the remote replica, the deliver operation proceeds.
- If the deliver operation contains activities that are mastered by the local replica, MultiSite transfers mastership of those activities to the remote replica. The project integrator at the remote site completes the deliver operation.

  If the project manager sets the Transfer ClearQuest Mastership After Delivery on the project, MultiSite transfers mastership of the activities back to the local replica.
- If the deliver operations contains activities that are mastered by a third replica, the deliver operation fails.

Rebasing Your Work Area

6



Periodically, your project manager or project integrator incorporates delivered work into new baselines. Some of these baselines constitute a stable and significant source configuration, and your project manager recommends the baseline.

Rebasing has several benefits:

- You stay up-to-date with other developers' work.
- You reduce the number of merges and minimize the time required to merge versions.
- You identify integration problems early.

Each time your project manager recommends a new baseline for your parent stream, rebase your development work area.

Rebasing involves these tasks:

- Preparing your development view
- Starting the rebase operation

- Testing your development work area
- Completing the rebase operation

Preparing Your Development View

Before you start a rebase operation, 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 the *Command Reference*.

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 write to the VOB the changes for a given file, check it in. Type this command:

     cleartool checkin *filename*

   - To undo your changes for a file, cancel its checkout. Type this command:

     cleartool uncheckout *filename*

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

Starting the Rebase Operation

After you prepare your development view (see *Preparing Your Development View* on page 104), you can start the rebase operation. By default, in a multiple-stream project, you rebase your development stream either to baselines recommended by the parent stream or to baselines created by the parent. Typically, these recommended baselines are created by other streams in the project. In a single-stream project, an integrator rebases the integration stream to identify a major project milestone (see *Rebasing in a Single-Stream Project* on page 109).

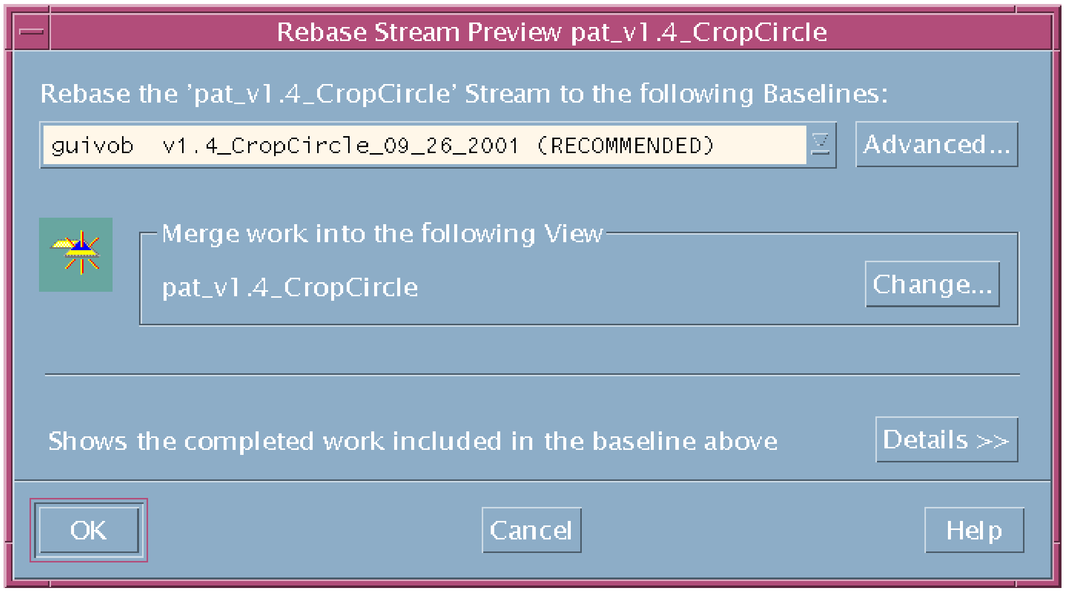
To Start the Rebase Operation

To start the rebase operation, enter the following command from your development view:

cleartool rebase –graphical

The Rebase Stream Preview dialog box opens (Figure 28), showing the name of the stream that you are using.

Figure 28 Rebase Stream Preview Dialog Box



Rebase Stream Preview pat_v1.4_CropCircle

Rebase the 'pat_v1.4_CropCircle' Stream to the following Baselines:

guivob v1.4_CropCircle_09_26_2001 (RECOMMENDED)

Merge work into the following View

pat_v1.4_CropCircle

Shows the completed work included in the baseline above

By default, the Rebase Stream Preview dialog box presents the parent stream recommended baseline.

You can also do the following in the Rebase Stream Preview dialog box:

- Verify that the dialog box displays your development view under **Merge work into the following view**. If you attached multiple views to your development stream, click **Change** and choose one view to use for the rebase operation.

  You cannot deliver or rebase from other views attached to the development stream until you complete the operation. However, you can check out, check in, and run most other ClearCase commands.

- To see the changes that will be available in your development stream after rebasing to the displayed baseline, click **Details**. The dialog box expands to show the list of activities in the baseline. To see more information about an activity, select it and click **Properties**.

  In the expanded area, select **Show integration activities** to see the baseline integration activities. An integration activity records a change set for a deliver or rebase operation.

- Control merge behavior (see *Merging Versions* on page 108)

  ClearCase tries to do the merge automatically. However, you can perform the merge on each element manually. To handle such a need, select **Merge elements graphically** to turn off all automated merging. If you select this option, the merge operation prompts you for confirmation before proceeding with each change.

To continue the rebase operation, click **OK**. To rebase to baselines other than recommended, see *Rebasing to Baselines Other Than Recommended*.

Rebasing to Baselines Other Than Recommended

The rules for rebasing to baselines other than recommended are different for integration streams and development streams. For an integration stream, you can only rebase to baselines from streams in other projects. The rules differ slightly for modifiable and read-only (nonmodifiable) components.

For a development stream, you can rebase to a baseline that is other than recommended and that is for a modifiable component if the baseline is compatible with your parent stream. You can rebase to a baseline from:

- The parent stream
- The parent stream foundation
- Other streams in the same project if the baselines were delivered to (and are contained in) your parent stream and contain the foundation baseline of your current stream.

The rebase operations ensures that you deliver your changes only from your stream.

If a baseline is for a nonmodifiable component, you are not restricted to selecting a baseline that is compatible with your parent stream. However, a restriction is imposed that you cannot modify elements in the component of that baseline. When you deliver from your stream, this restriction ensures that you deliver only changes made in your stream and that you do not strand changes in your stream foundation baseline.

The restriction allows you to configure your stream to a baseline of a nonmodifiable component that is not compatible with your parent stream or to a baseline of a component for which the parent stream does not configure a baseline. If you rebase your stream to an incompatible baseline and the component becomes modifiable, you still cannot modify the component until the baseline is made compatible with your parent stream (see *Delivering Changes to a Read-Only Component* on page 97).

To Rebase to Baselines Other Than Recommended

To rebase to baselines other than recommended, start the rebase operation (see *To Start the Rebase Operation* on page 105).

1. In the Rebase Stream Preview dialog box, click **Advanced**. In the Advanced Baseline Configuration dialog box, specify the list of baselines.
2. Click **Add**; in the Add Baselines dialog box, select baselines to be added to the list.
3. Select a listed baseline and do **one** of the following:
   - To remove it from the list, click **Remove**.
   - To modify the baseline, click **Change**. And, in the Change Baseline dialog box, select another baseline.

When you complete the list of baselines, click **OK** to quit the Advanced Baseline Configuration dialog box and click **OK** to continue the rebase operation.

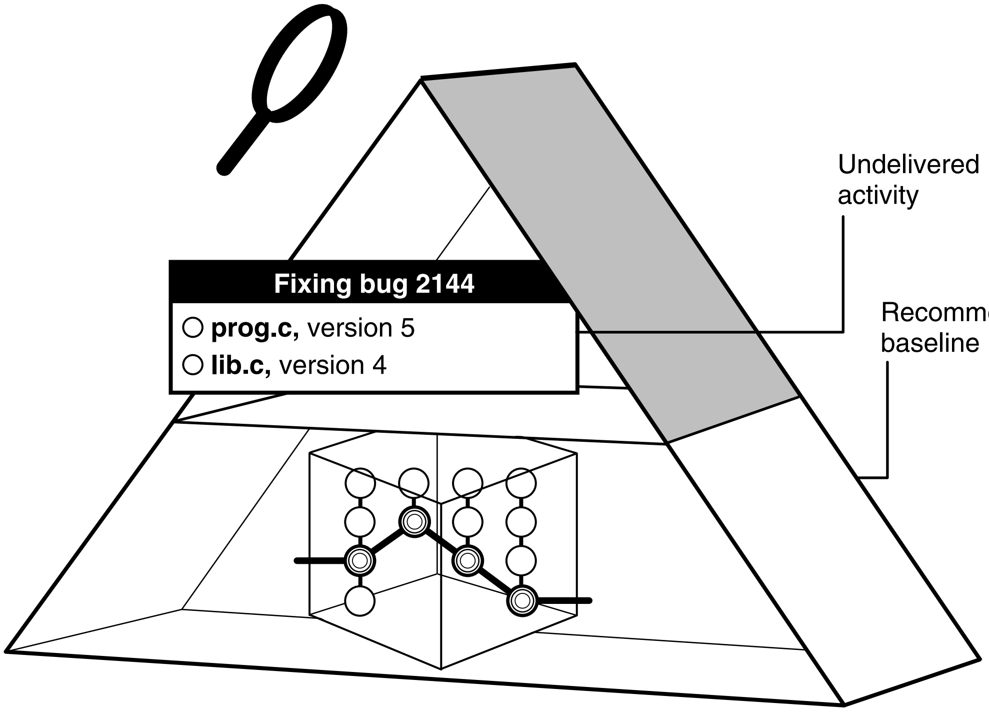
Rebasing Your Development Work Area

In rebasing your work area to the recommended baseline, Rational ClearCase takes the following actions:

- It changes your development stream configuration to select the new recommended baseline. The stream configuration continues to select any activities created in the development stream (Figure 29).
- For development views that are *snapshot views*, it starts an update operation to copy versions in the new baseline into your view.

  If other snapshot views are 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 29 Development Work Area After Rebasing



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

To Stop and Resume the Rebase Operation

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 modified and delivered a version of the same element, you must merge when you rebase to a baseline that contains the delivered version. The following circumstances involve merging:

- As part of working on an activity, you check in a version of prog.c from your development view.
- Another developer checks in and delivers a new version of prog.c to the parent stream.
- Your project manager adds the new version to the recommended baseline.
- 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 so that you can resolve the conflicts. For information about using Diff Merge to resolve conflicting differences, see *Using Diff Merge to Resolve Differences* on page 94.

Rebasing in a Single-Stream Project

Special considerations apply to rebasing in a view belonging to the integration stream of a single-stream project.

If you start the rebase operation from the graphical user interface and rebase detects a checked-out element, a pop-up appears warning you of the conflict. To continue the rebase operation, click **Continue** to resume the rebase or click **Close** to stop the rebase and resume later.

The element logs display additional information about the conflicting checkouts, for example, the view and the name of the user who checked out the file.

If you perform the rebase operation from the command line and rebase detects any checked-out element, the rebase warns you of the conflict. Rebase continues processing the remaining elements and merges all directories and files possible. Then the rebase interrupts to allow you to resolve any checkout and merge issues, after which you need to resume the rebase (see *To Stop and Resume the Rebase Operation* on page 108).

Testing Your Development Work Area

After ClearCase changes your stream 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 this ClearCase build tool for your builds; however, the *build auditing* and *build avoidance* features are available only from dynamic views.

For more information, see *Building Software* and the **clearmake** reference page in the *Command Reference*.

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 94, 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 operation. 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 undo 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. Close the **clearmrgman** window, if open.
3. From your development 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 by using the **cleartool rmver** command. Because **rmver** erases part of your organizational development history, your organization may not allow individual developers to use it. Be very conservative in using this command. For more information, see the **rmver** reference page in the *Command Reference*.

Completing the Rebase Operation

As with the deliver operation, ClearCase imposes the following 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. During the rebase completion, the **clearmrgman** window remains open.

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

cleartool rebase –complete –graphical

The rebase operation is placed in the Complete state.

Other Development Tasks 7

Chapter 4, *Working on Activities*, describes tasks that you perform daily or weekly. You may need to perform some of the tasks described in this chapter 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
- Registering a snapshot view
- Regenerating the .ccase_svreg file
- Moving views
- Accessing views and VOBs across platform types
- Regenerating the .view.dat file

Adding Files and Directories to Source Control

You can add files or directories to source control (version control) at any time during the development cycle (see *Components and Baselines* on page 14).

To Add Files and Directories to Source Control

You can add view-private files and directories to source control or make placeholders for nonexistent files and directories. Do the following:

1. In your development view, set an activity (see *Setting Your View to an Activity* on page 53).
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 this command:

   cleartool checkout –nc *directory-name*

We suggest using the **–nc** option because Rational 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 in the following format:

  cleartool mkdir *directory-name*

- To add a file to source control, enter this command in the following format:

  cleartool mkelem *file-name*

- To make placeholders for nonexistent objects, enter one of these commands in the following format:

  cleartool mkdir *directory-element-path*

  cleartool mkelem *file-element-path*

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

For more information about the **mkelem** command, see *Under the Hood: What Happens When You Add to Source Control* on page 114 and the **mkelem** reference page in the *Command Reference*.

Under the Hood: What Happens When You Add 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:

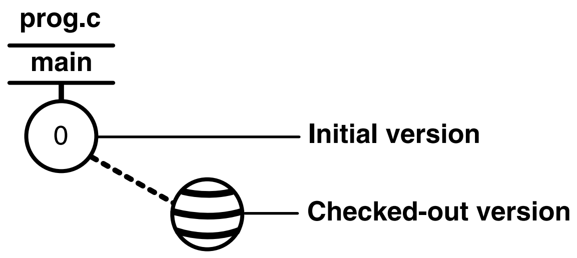
- Using **mkelem** with no arguments checks out the element. Any view-private data that corresponds to the element path remains in your view only and is added to version 1 in the VOB when you check in (Figure 30).
- Using **mkelem –ci** checks in the element, using any existing view-private data that corresponds to the element path as the content for version 1. Your view 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-path* exists, it is moved to a .keep file.

- (Replicated VOBs only) Using **mkelem –master** or **mkdir –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** or **mkdir** 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 parent directories of the element and check in the file or directory.

Figure 30 Creating an Element



Importing Files

If you add a large number of files and directories to source control, use the **clearfsimport** command (or **clearexport** command) and **clearimport** command. For more information, see the **clearfsimport** and **clearimport** reference pages in the *Command Reference*.

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 prog.c. If you remove prog.c from the design directory, ClearCase creates version 5 of /gui_vob/design, which does not contain the prog.c file element. The element prog.c itself is not modified.

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

Moving an Element to Another VOB

You cannot move an element in a UCM component VOB to another VOB.

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

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
  prog.c@@/main/2
  lib.c@@/main/10
cleartool checkout –nc design
   Checked out “design” version “/main/4”
cleartool mv prog.c  msg.cat
  Moved “prog.c” to “msg.cat”
  cleartool checkin design
  Default:
  Added file element "msg.cat".
Removed file element "prog.c".
Checkin comments for ".":  ("." to accept default)
.
  Checked in “design” version “/main/5”
cleartool ls design@@/main/5
  msg.cat@@/main/2
  lib.c@@/main/10
cleartool ls design@@/main/4
  prog.c@@/main/2
  lib.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 *current-path target-path*

3. Check in the parent directory.

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 may want to check periodically whether the functional specifications have been modified by reviewing the element 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 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

You can see all elements in a directory. Enter this command:

cleartool ls *path...*

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

Viewing the Contents of a Nonloaded Version

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. Use the **cleartool get** command to access a version of a file not loaded into your view. The command copies the version you specify into your view.

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

To Copy a Nonloaded Version of a File Element into Your View

Type a command in this format:

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

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

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 the scope of your view 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 stream to which the view is attached 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 scope.

Changing Which Elements Are Loaded into a Snapshot View

A set of load rules determines 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 config spec

  A config spec is a set of rules that written and maintained by the stream to which the view is attached and 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 config spec, ClearCase updates the entire view. This is an appropriate course of action when you deliver or rebase the stream to which the view is attached, 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 scope.

- By using **update –add_loadrules**

  The **–add_loadrules** option of **cleartool update** adds load rules to your view 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 config spec for editing:

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

b Enter the following command:

cleartool edcs

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

In your text editor, your view 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* [*path* ...]

For example, the rule **load** /gui_vob loads the /gui_vob component and all the files and directories it catalogs. To limit the scope of this load rule to a subordinate tree within the component, add the subordinate tree path. For example, the rule **load** /gui_vob/batch/calc loads the subordinate tree starting from the calc directory. You can use a load rule to specify a single file. For example, the rule **load** /gui_vob/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** /gui_vob and **load** /gui_vob/batch/calc, ClearCase loads the entire /gui_vob 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-path* [...]

The argument *element-path* identifies an element in the component namespace at a path that is relative to a snapshot view. For example, the following command loads all elements in a component named /gui_vob into the view **pat_v1.4_cropcircle_sv**:

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

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

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

The following commands load 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 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. To mount a private VOB, specify its VOB tag in the **mount** command. 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 about mounting VOBs, see the **mount** reference page in the *Command Reference*.

To Unmount VOBs

From a shell, enter this command:

cleartool umount *VOB-tag*

For more information about unmounting VOBs, see the **umount** reference page in the *Command Reference*.

Moving Work from a Development Stream

Under normal circumstances, project managers and integrators use baselines 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; doing so 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-selector of the activity 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 integration stream or to another development stream in the new project (see *Integration Stream* on page 36).
4. From your view in the original project, deliver the work to a nondefault target (the stream in the new project in which you intend to complete the work). (See *Delivering to a Nondefault Target* on page 86.)

   To specify the work to be delivered, use the activity selector that you obtained in Step 1.

5. Lock, in the obsolete state, your development stream for the original project.
6. Remove your development view for the original project.

Registering a Snapshot View

If you need to perform ClearCase operations in a snapshot view that you did not create or have not updated, you must register that view.

When you create or update a snapshot view, ClearCase registers it in the file .ccase_svreg in your home directory. When ClearCase needs to access a snapshot view, it looks in that file in your home directory for the path for the snapshot view root directory. If you did not create or have not updated the snapshot view that ClearCase is trying to access, you must explicitly register the view.

If a snapshot view is not registered under your home directory and you try to use the view to perform ClearCase operations (for example, as the target of a deliver operation or as the source for a rebase operation), ClearCase cannot find the view. To use this view to perform ClearCase operations, register the view (see *To Register a Snapshot View* on page 123). If you inadvertently delete or corrupt the .ccase_svreg file, you must regenerate the information in the file (see *To Regenerate the .ccase_svreg File* on page 123).

To Register a Snapshot View

To register a snapshot view that you did not create or have not updated, perform the following steps:

1 Change your working directory to the root directory of the snapshot view.

2 From that directory, use either **cleartool update** or **cleartool update -print**.

This registers the view for your use on any UNIX workstation. The preview form of the command (**update -print**) is quicker than update and does not change the loaded files.

Note: You cannot register a snapshot view that was created from a Windows host (see *Accessing Views Across Platforms of Different Types* on page 126).

To Regenerate the .ccase_svreg File

When you create a snapshot view or register a snapshot view, ClearCase creates or modifies the file .ccase_svreg in your home directory. Some ClearCase operations use information from this file. If the .ccase_svreg file in your home directory has been deleted or corrupted, you must regenerate the file. To regenerate information in .ccase_svreg, follow the steps in *To Register a Snapshot View* for each snapshot view that you use.

Moving Views

This section discusses the following tasks:

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

For information about changing a view tag, see the **mktag** reference page in the *Command Reference*.

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 it. 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 path 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_server* process of the view 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. The *Administrator's Guide* for Rational 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 the *Administrator's Guide*.

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 views and VOBs across platform types
- Developing software across platform types

Creating Views Across Platform Types

Your ClearCase 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 the *Command Reference*.

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 ClearCase host that runs Windows software and 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 the Operating System

For snapshot views, the operating system from which you create the view determines view characteristics; the operating system that hosts the files and processes related to a snapshot view do not affect the view 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). Although 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:

- Case-sensitive file lookups are not performed in the view.
- The view does not create symbolic links if the load rules encounter a VOB symbolic link.
- You can issue ClearCase commands for the view only from Windows hosts. (ClearCase hosts on UNIX do not recognize the directory tree as a snapshot view.)

Accessing Views Across Platforms of Different Types

This section describes support for accessing a view residing on a platform that differs from the platform from which it is being accessed.

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. On the UNIX system, create the UNIX view with the proper text mode. For more information, see *Developing Software Across Platforms of Different Types* on page 127.
2. On the Windows system, use Region Synchronizer to import the UNIX view tag of the view into your Windows network region.
3. In ClearCase Explorer, refresh view shortcuts. ClearCase Explorer detects the view tag, starts the view, and creates a view shortcut for you.

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 Platforms of Different Types

Your ClearCase administrator sets up VOBs on hosts running Windows or UNIX software and creates *VOB tags* in each ClearCase network region that needs to access the VOBs. (For information about registering UNIX VOB tags on a Windows computer, see the *Administrator's Guide* for Rational ClearCase.) Then, from any ClearCase host running Windows or UNIX software, you can create snapshot views to load elements from VOBs that have tags in your network region.

From dynamic views and snapshot views on a ClearCase host on a Windows computer that supports dynamic views, you can access VOBs on Windows computers and UNIX workstations. To access VOBs on UNIX workstations from dynamic views on Windows computers, you must use third-party software that provides access to UNIX file systems from Windows computers.

From dynamic views on a ClearCase host on a UNIX workstation, you cannot access VOBs on Windows computers. Table 3 summarizes your options for accessing VOBs across platforms of different types.

Table 3 ClearCase VOB Access Across Different Platforms

ClearCase host platform	Platform on which VOB is located	Type of 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 Platforms of Different Types

To use a UNIX view on a Windows host to access text files, you need an interop text-mode view. Before a VOB can be accessed from an interop text-mode view, the system administrator must enable the VOB for interop text-mode support.

If developers check in source files from views created on hosts running either Windows and UNIX, 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 choosing a text mode for a view, see the Rational ClearCase Help.

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-path*

   For example:

   Perl /usr/rational/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.

Working in a Snapshot View Without the Network

A

If you need to work with your source files from a computer that is disconnected from the network of Rational 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
- Reconnecting 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 Web interface for ClearCase. Ask your ClearCase administrator whether the Web interface for ClearCase 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 Help.

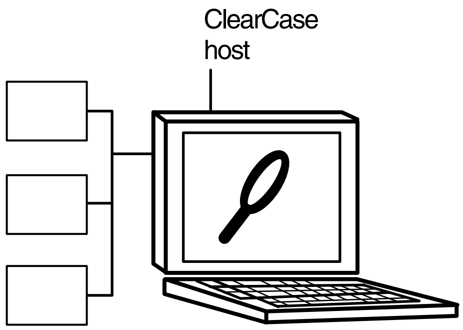
Setting Up a View for Your Hardware Configuration

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 31).

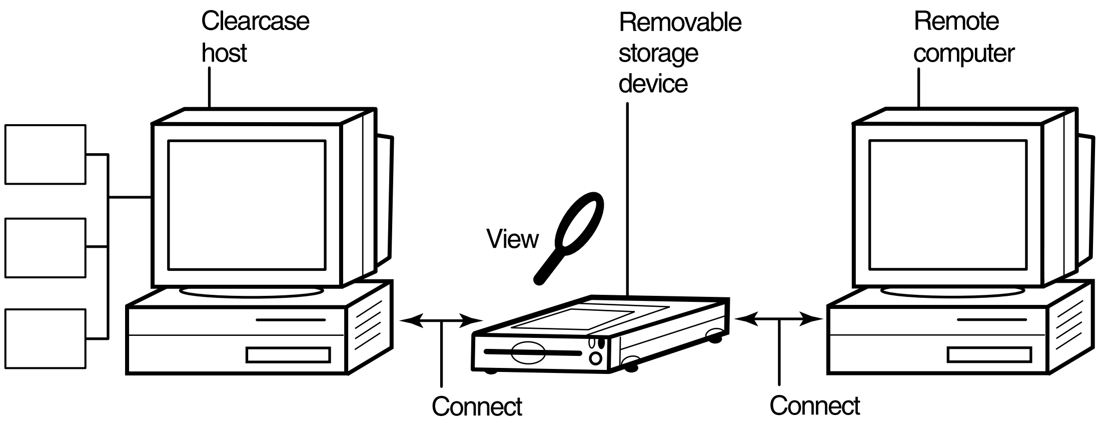
Figure 31 View on a Laptop



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 32).

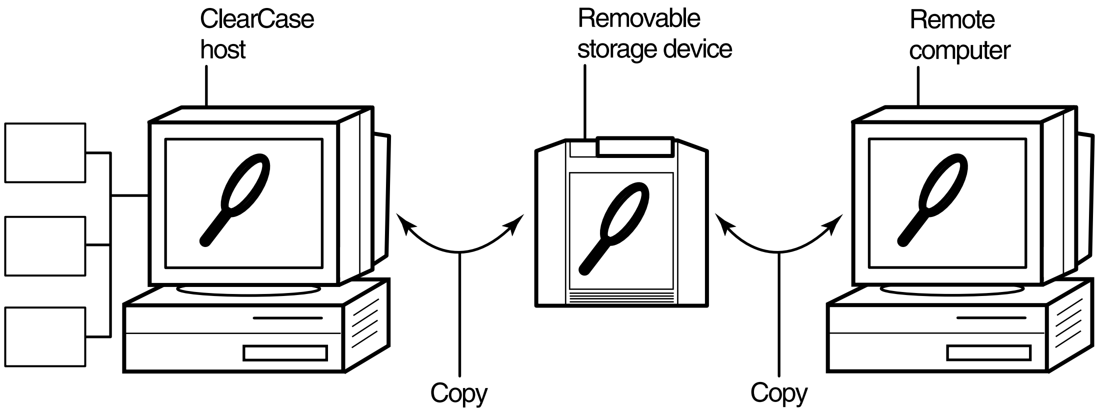
Figure 32 View On a Removable Storage Device



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 33).

Figure 33 Copy the View



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

Under the Hood: View Storage 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 the *view_server* process of a view runs on the host that contains the view storage directory or on a host that can access a location on a supported Network Attached Storage (NAS) device which 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.

Preparing the View

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

- Update the view to establish a checkpoint. (For information about updating the view, see *Updating the View* on page 137.)
- Check out the files you expect to modify. After you are disconnected from the network, you cannot check out files, although there are workarounds. (See *Hijacking a File* on page 132.)

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.

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
```

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: Determining Whether a File Is Hijacked* on page 136.

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 are ready, check in your changes.
- Undo the hijack. For more information, see *To Undo a Hijack* on page 136.

To Hijack a File

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

chmod +w prog.c

To Find 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-path* **–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.

Reconnecting 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.

Using the Update Tool

When you are 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
- Merging changes to a hijacked file
- Undoing a hijack
- Choosing other ways to handle hijacked files

To Find Hijacked Files

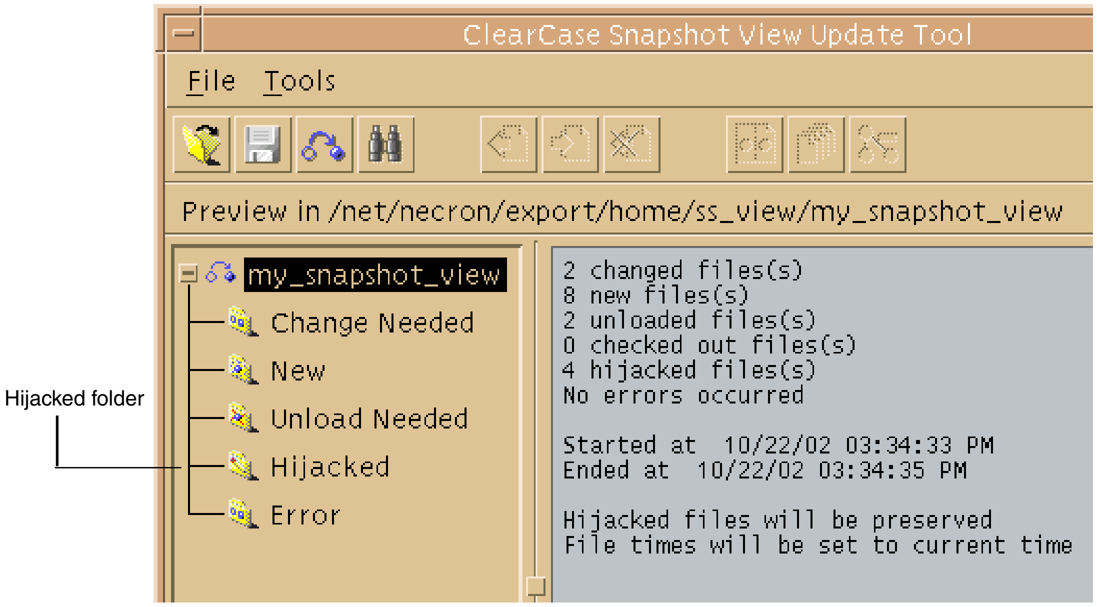
1 Enter the following command:

  cleartool update –graphical *snapshot-view-path*

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

3 If any hijacked files are in your view, the Snapshot View Update window displays a folder in the left pane titled **Hijacked** (Figure 34). Select **No** for the option asking whether you want to check out the hijacked files now.

Figure 34 Hijacked Files in the Update Window


ClearCase Snapshot View Update Tool
File
Tools
Preview in /net/necron/export/home/ss_view/my_snapshot_view
my_snapshot_view
Change Needed
New
Unload Needed
Hijacked
Error
Hijacked folder

```
2 changed files(s)
8 new files(s)
2 unloaded files(s)
0 checked out files(s)
4 hijacked files(s)
No errors occurred

Started at  10/22/02 03:34:33 PM
Ended at   10/22/02 03:34:35 PM

Hijacked files will be preserved
File times will be set to current time
```

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 Snapshot View Update window, click a hijacked file.

2 Click **Tools > Compare with old**. For information about using the Diff Merge tool, see the Help.

To Check Out a Hijacked File

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

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

2 Click **Tools > Checkout**.

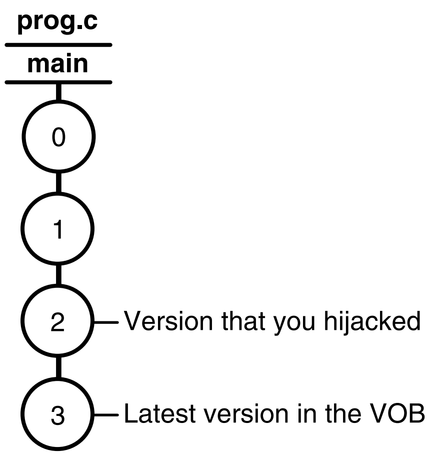
3 ClearCase treats a checked-out hijacked file as it does any other checkout.

When you are ready, you can check in the file.

Merging the Latest Version to a Hijacked File

If you are 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 35), ClearCase prevents you from checking in the file.

Figure 35 Hijacked Version May Not Be the Latest Version



You have to merge the latest version in the VOB with the checked-out file before ClearCase allows the checkin.

To merge the latest version in the VOB to the checked-out 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 path*. For more information, see *The Version-Extended Path* on page 70.

For example:

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

Using the **–graphical** option starts the Diff Merge tool. For information about using the Diff Merge tool, see the Help for ClearCase. After merging, save the results and check in the version by entering the **cleartool checkin** command from the view.

To Undo 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 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, see Step 2 in *Updating the View* on page 137.

Under the Hood: Determining Whether a File Is Hijacked

To keep track of file modifications in a snapshot view, ClearCase stores the size and last-modified time stamp of a loaded file (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 the read-only permission of a non-checked-out file 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

For more information, see *Updating the View*.

Updating the View

1. Enter the following command:

   cleartool update –graphical *snapshot-view-path*

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 these choices:

   - Leave hijacked files in place
   - Rename the hijacked files and load the selected version from the VOB
   - Delete hijacked files and load the selected version from the VOB

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

Index