

# Administering Rational ClearQuest

[support@rational.com](mailto:support@rational.com)  
<http://www.rational.com>

**Rational**  
the e-development company™

## **IMPORTANT NOTICE**

### **COPYRIGHT NOTICE**

ClearQuest, copyright © 1997–2000 Rational Software Corporation. All rights reserved.

THIS DOCUMENT IS PROTECTED BY COPYRIGHT AND CONTAINS INFORMATION PROPRIETARY TO RATIONAL. ANY COPYING, ADAPTATION, DISTRIBUTION, OR PUBLIC DISPLAY OF THIS DOCUMENT WITHOUT THE EXPRESS WRITTEN CONSENT OF RATIONAL IS STRICTLY PROHIBITED. THE RECEIPT OR POSSESSION OF THIS DOCUMENT DOES NOT CONVEY ANY RIGHTS TO REPRODUCE OR DISTRIBUTE ITS CONTENTS, OR TO MANUFACTURE, USE, OR SELL ANYTHING THAT IT MAY DESCRIBE, IN WHOLE OR IN PART, WITHOUT THE SPECIFIC WRITTEN CONSENT OF RATIONAL.

### **U.S. GOVERNMENT RIGHTS NOTICE**

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

### **TRADEMARK NOTICE**

Rational, the Rational logo, ClearQuest, ClearCase, Purify, PureCoverage, and Quantify are trademarks or registered trademarks of Rational Software Corporation in the United States and in other countries.

Visual Basic, Windows NT, and Microsoft are trademarks or registered trademarks of the Microsoft Corporation. All other names are used for identification purposes only and are trademarks or registered trademarks of their respective companies.

### **U.S. PATENT NOTICE**

U.S. Registered Patent Nos. 5,193,180 and 5,335,344 and 5,535,329. Licensed under Sun Microsystems Inc.'s U.S. Pat. No. 5,404,499. Other U.S. and foreign patents pending.

Printed in the U.S.A.

Part number: 800-023976-000

# Contents

<b>1</b>	<b>Before you begin</b>	
	Audience . . . . .	9
	Other ClearQuest documentation . . . . .	9
	ClearQuest documentation roadmap . . . . .	10
	Contacting Rational Software . . . . .	11
<b>2</b>	<b>Understanding ClearQuest administration</b>	
	Starting ClearQuest Designer . . . . .	13
	Overview of ClearQuest architecture . . . . .	14
	How components work together . . . . .	14
	How components are used . . . . .	15
	Overview of administrator tasks . . . . .	16
	Working with ClearQuest schemas and databases . . . . .	17
	Defining your workflow . . . . .	18
	Working with record types, states, and actions . . . . .	18
	Defining a state model . . . . .	18
	Customizing your workflow . . . . .	19
	Before you begin customizing . . . . .	19
	Getting ClearQuest users started . . . . .	20
<b>3</b>	<b>Managing databases</b>	
	ClearQuest required databases . . . . .	24
	Where to keep ClearQuest databases . . . . .	24
	Maintaining database integrity . . . . .	24
	Selecting a database vendor . . . . .	25
	Creating new databases . . . . .	26
	Backing up your data . . . . .	26

Overview of database maintenance .....	27
Moving a database to a new location or to a new vendor .....	29
Preparing to move a database .....	29
Moving a schema repository .....	30
Moving a user database .....	31
Moving a database to a different schema .....	34
Important considerations when moving to a new schema .....	35
Updating user databases .....	36
Viewing database properties .....	37
Using the ClearQuest Export Tool .....	38
Deleting a user database .....	42
Undeleting a user database .....	43
<b>4 Working with ClearQuest schemas</b>	
Overview of working with schemas .....	46
Working with a test database .....	47
Checking out a schema .....	49
Creating a new schema .....	50
Selecting a scripting language .....	51
Validating schema changes .....	52
Checking in a schema .....	53
Undoing a schema checkout .....	53
Upgrading a user database .....	54
Deleting a schema .....	54
Selecting a ClearQuest schema .....	55
Adding packages to a schema .....	56
<b>5 Customizing a schema</b>	
Planning schema customizations .....	62
Working with record types .....	64
State-based record types .....	64
Stateless record types .....	64
Adding a new record type .....	65

Selecting a unique key for a stateless record type . . . . .	.66
Selecting a default record type . . . . .	.66
Adding a new record type family . . . . .	.67
Renaming a record type . . . . .	.69
Deleting a record type . . . . .	.69
Working with fields . . . . .	.70
Adding a new field . . . . .	.71
Defining field behavior . . . . .	.73
Changing the name of a field . . . . .	.74
Deleting a field . . . . .	.74
Using fields to link records . . . . .	.75
Customizing fields by adding hooks . . . . .	.79
Defining your state model . . . . .	.80
Using the State Transition Matrix . . . . .	.81
Adding a new state . . . . .	.82
Mapping state types . . . . .	.82
Changing the name of a state . . . . .	.83
Working with actions and action types . . . . .	.84
Adding and modifying actions . . . . .	.86
Adding a new action . . . . .	.86
Creating a state transition . . . . .	.87
Customizing actions by adding hooks . . . . .	.88
Using default actions . . . . .	.89
Working with record forms . . . . .	.90
Creating a new form . . . . .	.90
Working with form controls . . . . .	.91
Adding a field to a form . . . . .	.93
Editing field control properties . . . . .	.94
Editing forms . . . . .	.94
Reusing forms . . . . .	.95
Deleting forms . . . . .	.95
Creating forms for multiple platforms . . . . .	.96

<b>6</b>	<b>Administering users</b>	
	Overview of user administration	98
	ClearQuest user privileges	99
	Working with users	100
	Adding new users	100
	Upgrading the user database	101
	Subscribing users to databases	102
	Modifying user profiles	103
	Making users inactive	104
	Working with user groups	105
	Creating user groups	105
	Adding users to a group	106
	Subscribing user groups to databases	107
	Making user groups inactive	108
	Controlling user access to actions	109
	Exporting and importing users and groups	110
<b>7</b>	<b>Using hooks to customize your workflow</b>	
	Overview of hooks	112
	Planning hooks for your workflow	113
	Using field hooks	114
	Creating a dependency between fields	115
	Using dependent fields on ClearQuest Web	116
	Working with choice lists	117
	Creating a dynamic list hook	118
	Defining choice list behavior	119
	Using action hooks	120
	Execution order of field and action hooks	122
	When an action begins	122
	When a field's value is set	122
	When the record is validated	123
	When the record is committed	124
	Using record scripts	125

Using global scripts	126
Writing external applications	126
Using the ClearQuest API	127
Working with sessions	127
Working with queries	127
Working with records	128
Common API calls	129
Finding hook script text	131
Examples of common hooks	133
Hook for creating a dependent list	133
Field choice list hook to display user information	135
Action initialization hook for a field value	137
Action hook for setting the value of a parent record	138
<b>8 Administering ClearQuest Web</b>	
ClearQuest Web considerations	144
Customizing ClearQuest Web	144
Limiting access to ClearQuest Web	145
Using hooks in ClearQuest Web	146
Enabling dependent fields for ClearQuest Web	146
Displaying messages on ClearQuest Web	147
Using hooks to detect a web session	147
<b>9 Administering ClearQuest E-mail</b>	
Enabling automatic e-mail	150
Setting up e-mail rules	150
Configuring ClearQuest clients to send e-mail	151
Submitting records by e-mail	153
Configuring Rational E-mail Reader	153
Formatting e-mail for submission	154
Using “round-trip” e-mail	156
<b>10 Importing data into ClearQuest</b>	

Preparing for a successful data import .....	158
Creating an import schema .....	158
Creating a database for imported data .....	160
Testing the import process .....	160
Creating a ClearQuest import file .....	161
Formatting the record import files .....	161
Formatting the history import file .....	164
Formatting the attachments import file .....	165
Performing the data import .....	166
Importing history .....	171
Importing attachments .....	172
Importing duplicate records .....	173
Importing records from the error file .....	174
Upgrading existing records .....	174
<b>11 ClearQuest schemas and packages</b>	
ClearQuest predefined schemas .....	A-2
ClearQuest packages .....	A-4
State model for the Defect record type .....	A-11
State model for the EnhancementRequest record type .....	A-12
State-type models for packages .....	A-13
Resolution package state type model .....	A-13
UnifiedChangeManagement package state type model .....	A-14
<b>12 Enabling ClearQuest for Unified Change Management</b>	
Adding UCM functionality to a schema .....	B-2
Installing the UCM packages .....	B-2
Setting default actions for the UCM package .....	B-4
<b>13 Index</b>	



# 1

## Before you begin

Rational ClearQuest® is a highly flexible defect and change tracking system that captures and tracks all types of change, for any type of project. This guide explains the concepts and initial steps required to begin administering Rational ClearQuest.

### Audience

This guide is for ClearQuest administrators. It assumes that you have read *Introducing Rational ClearQuest*, have experience administering relational databases, and know how to write scripts in VBScript or Perl.

For specific step-by-step instructions, see the ClearQuest Designer Help. Select Help > Contents and Index. The ClearQuest Designer Tutorial provides six lessons that cover the basics of ClearQuest administration. To take the tutorial, select Rational ClearQuest Designer Tutorial from the Start menu.

### Other ClearQuest documentation

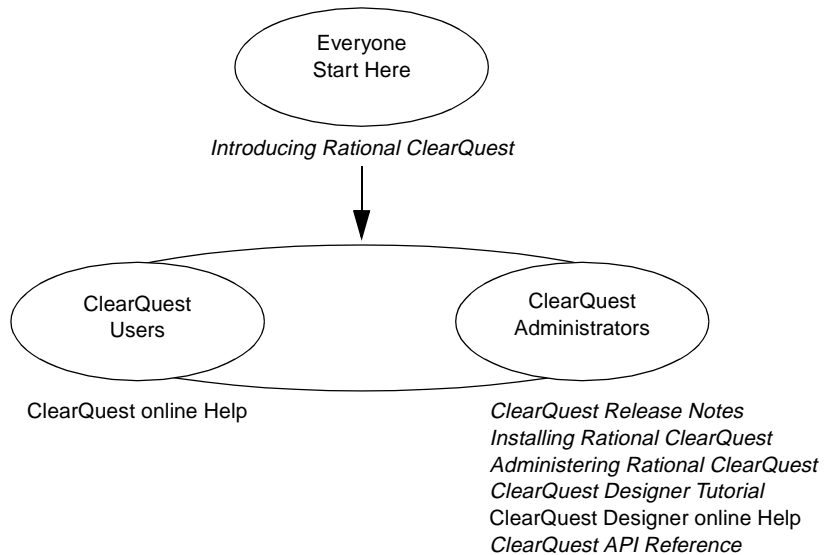
In addition to this guide, ClearQuest includes the following printed documentation:

- *Introducing Rational ClearQuest*: Provides an overview of how to use ClearQuest, and a brief example of how you can customize ClearQuest to fit your organization's workflow.
- *Installing Rational ClearQuest*: Explains how to install ClearQuest, ClearQuest Designer, vendor databases, and related tools.

ClearQuest includes the following online documentation:

- *ClearQuest Release Notes*, located in `\Rational\doc\clearquest_readme.htm`
- *ClearQuest Help*: Online Help for ClearQuest.
- *ClearQuest Designer Help*: Online Help for ClearQuest Designer.
- *ClearQuest Designer Tutorial*: Introduction to the basic tasks of a ClearQuest administrator.
- *ClearQuest API Reference*: Online reference guide explaining the syntax for writing hooks and external applications.
- *ClearQuest Web Help*: Online Help for the ClearQuest web-based client.
- *ClearQuest Maintenance Tool Help*: Online Help for the ClearQuest Maintenance Tool.
- *ClearQuest Import Tool Help*: Online Help for the ClearQuest Import Tool.

## ClearQuest documentation roadmap



## Contacting Rational Software

If you have a technical problem and you cannot find the solution in this guide or in the online Help, contact Rational Software Technical Support. Select **Reference > Contacting Technical Support** in the ClearQuest Help for addresses and phone numbers of technical support centers.

Before contacting technical support, note the sequence of events that led to the problem and any program messages you see. If possible, have the product running on your computer when you call.

For technical information about ClearQuest, answers to common questions, and information about other Rational Software products, visit the Rational Software World Wide Web site at <http://www.rational.com>. To contact technical support directly, use <http://www.rational.com/support>.



# 2

## Understanding ClearQuest administration

As the ClearQuest administrator, your job is to set up, customize, and maintain ClearQuest for your end users. This chapter provides essential concepts to help you administer ClearQuest and to use this guide effectively. The topics covered include:

- Overview of ClearQuest architecture
- Overview of administrator tasks
- Working with ClearQuest schemas and databases
- Defining your workflow
- Getting ClearQuest users started

**Note:** The term *administrator*, as used in this guide, means any person with user-access privileges above that of Active User. For definitions of user-access privileges, see “ClearQuest user privileges” on page 99.

### Starting ClearQuest Designer

To perform most ClearQuest administration, you will use ClearQuest Designer. Select **Rational ClearQuest Designer** from the Start menu.

ClearQuest Designer comes with a default User Name (*admin*) that you can use to get started. You do not need to type a password. The *admin* user account has Super User privileges so you can perform all ClearQuest administrator functions.

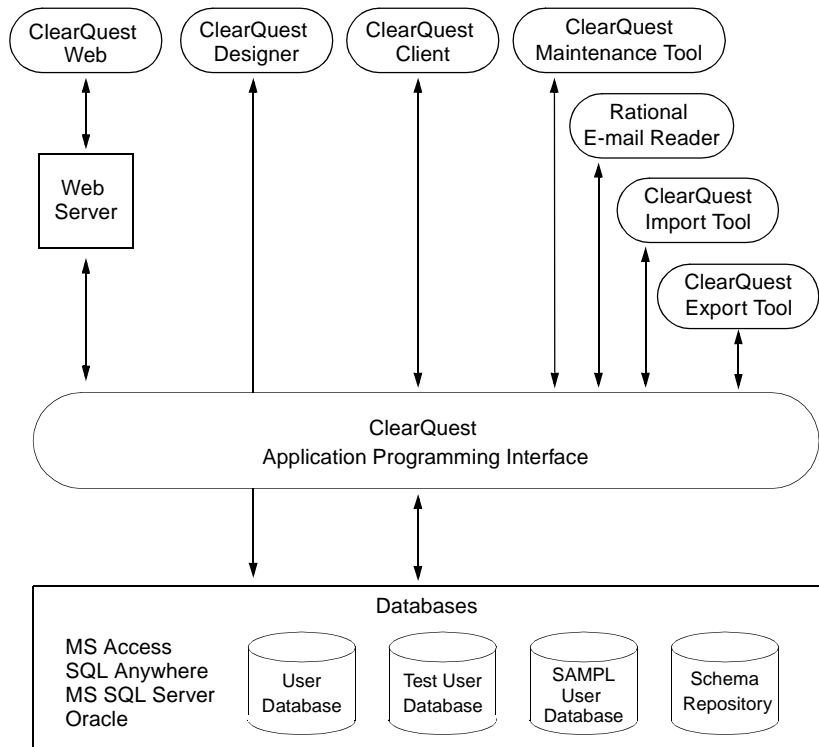
**Note:** The first thing you might want to do is add a password to the *admin* account or create a new user account to limit the access to ClearQuest Designer. See Chapter 6, “Administering users,” specifically “Modifying user profiles” on page 103.

## Overview of ClearQuest architecture

ClearQuest consists of several components that work together in a client-server environment.

### How components work together

As the ClearQuest administrator, you work with each of these components:



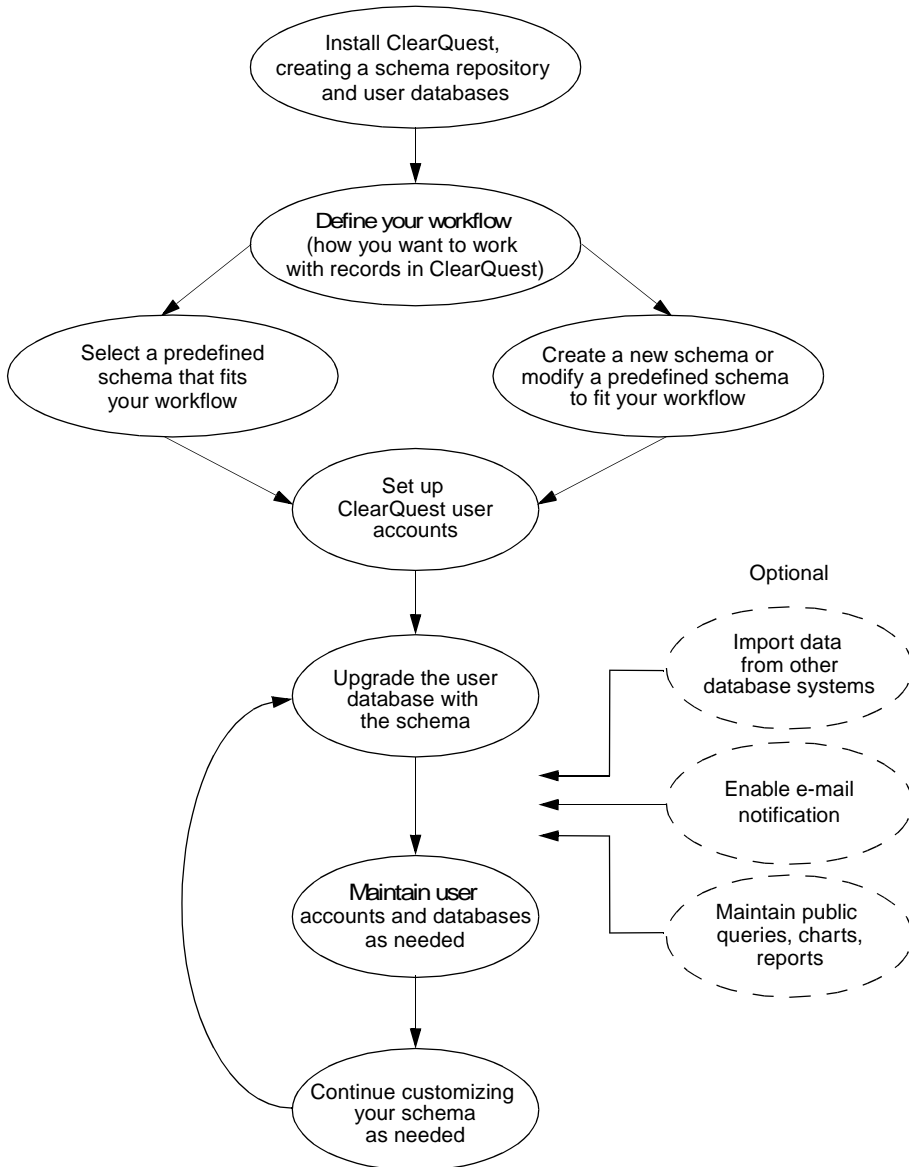
## How components are used

Here's how you use ClearQuest components:

Component	Used by	Use to
<b>Client tools</b>		
ClearQuest for Windows	Everyone	Submit, modify, and track change requests, and to analyze project progress by running queries, charts, and reports.
ClearQuest for UNIX	Everyone	Submit, modify, and track change requests, and to analyze project progress by running queries.
ClearQuest Web	Everyone	Access ClearQuest across multiple platforms through Netscape Navigator® or Microsoft's Internet Explorer.  Submit change requests and run queries. On Windows you can also run charts, and reports.
<b>Administrator tools</b>		
ClearQuest Designer	ClearQuest administrator	Customize ClearQuest, manage ClearQuest schemas and databases, and administer users and user groups.
ClearQuest Import Tool	ClearQuest administrator	Import data, including records, history, and attachments, from other change request systems.
ClearQuest Export Tool	ClearQuest administrator	Export ClearQuest data from one ClearQuest user database to another user database that uses a different schema.
ClearQuest Maintenance Tool	Everyone	Set up and connect to the schema repository during installation and when you upgrade to a new ClearQuest version.
Rational E-mail Reader	ClearQuest administrator	Enable ClearQuest users to submit and modify records by e-mail.  See Chapter 9, "Administering ClearQuest E-mail."

## Overview of administrator tasks

Below is an overview of the ClearQuest administrator tasks.





## Working with ClearQuest schemas and databases

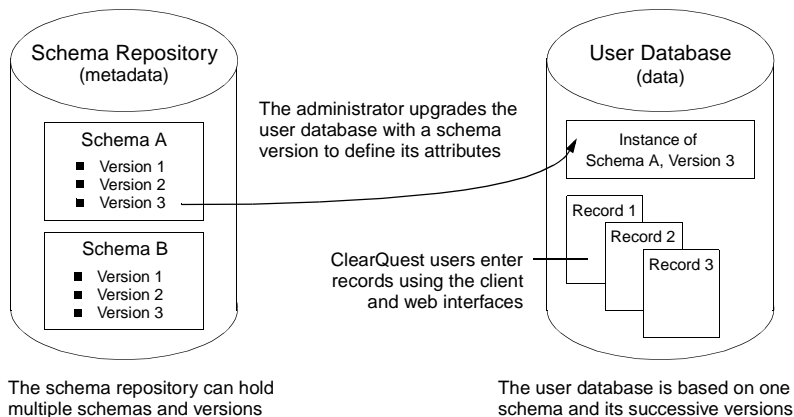
A ClearQuest schema contains the metadata that defines your workflow, that is, how your users work with records within ClearQuest. It includes record definitions, form and field definitions, record states and the actions that can be performed on records, and optional hook code that customizes your workflow.

ClearQuest includes several predefined schemas that provide common workflows and support integration with various Rational Software products. For a description of ClearQuest schemas, see Appendix A, “ClearQuest schemas and packages.”

ClearQuest uses relational databases to store the data about your process and forms (metadata) and the data (records) entered by ClearQuest users. ClearQuest requires at least two databases:

- A schema repository (master database) where ClearQuest stores schemas.
- One or more user databases to hold the data entered by the users of ClearQuest and ClearQuest Web. You can have as many user databases as you need.

The schema defines the attributes of the user database. As the ClearQuest administrator, you will use ClearQuest Designer to customize schemas and to apply schemas to user databases.



## Defining your workflow

Your most important tasks as the ClearQuest administrator are to define how you want your users to work with records within ClearQuest and then select the predefined ClearQuest schema that fits your workflow, or customize a schema to fit your needs.

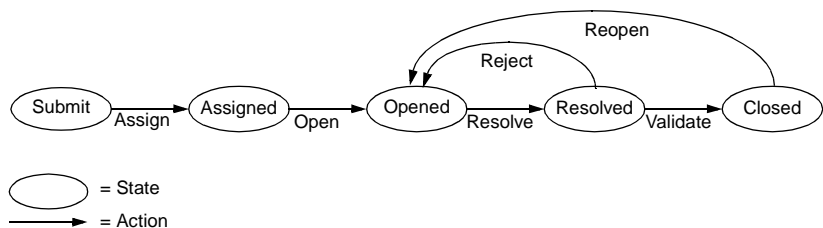
ClearQuest predefined schemas are made up of packages that contribute specific functionality to the schemas. For example, the Email package included in most schemas enables ClearQuest to send automatic e-mail notifications. The easiest way to add functionality to a schema is to add one or more ClearQuest packages. See “Adding packages to a schema” on page 56.

### Working with record types, states, and actions

ClearQuest allows you to track multiple types of records, such as defects and enhancement requests. ClearQuest users work with records by moving them through various stages, or “states,” usually beginning with when the record is submitted to the system and ending with when it is closed. Movement from state to state is initiated when a user performs an action such as Open or Resolve on the record.

### Defining a state model

Each record type has a state model that defines how it can be used. Below is a typical state model showing how a change request record moves from state to state as a result of the actions performed by ClearQuest users.



## Customizing your workflow

You can refine your state model by adding rules and permissions that support your workflow. You do this by using the predefined hook code and controls that are included in ClearQuest. You can also use the ClearQuest application programming interface (API) to write your own hook code or to write external applications that reinforce your process. You can:

- Control the type of data you collect by customizing the fields you use and how they behave.
- Define access controls that determine who can perform actions and when those actions can be performed.
- Define what happens when an action is performed. Actions can trigger other events, such as sending automatic e-mail notifications when a specific action is performed.

For example, you can restrict actions to specific users or user groups. You might allow everyone on the team to resolve a change request, but allow only a quality assurance engineer to validate the resolution and close the record. You can also designate a person to be responsible for supplying data before the request can move to the next state.

## Before you begin customizing

Before you begin customizing ClearQuest, be sure to read:

- Chapter 4, “Working with ClearQuest schemas.” This chapter is essential reading whether you are using a predefined schema or creating a schema from scratch. It details the process for working with schemas, from checking them out of the schema repository to upgrading the user database with the latest schema changes.
- Chapter 5, “Customizing a schema,” especially the section “Planning schema customizations” on page 62.
- Appendix A, “ClearQuest schemas and packages” for a complete list of ClearQuest’s predefined schemas and schema packages.

## Getting ClearQuest users started

Below is a list of tasks you must perform to get your ClearQuest users started:

<b>Task</b>	<b>Do this</b>	<b>To find out how, see</b>
Install ClearQuest and set up databases	Create a schema repository and one or more user databases	<i>Installing Rational ClearQuest</i>
Define your workflow	Plan states and actions; create a state transition model; define record types, forms and fields; plan hook code	This chapter and Chapter 4, "Working with ClearQuest schemas"  Chapter 5, "Customizing a schema"
Select a predefined schema that fits your workflow		Appendix A, "ClearQuest schemas and packages"
Customize the selected schema or build a new schema from scratch (optional)	Begin by adding predefined packages to build the functionality you need	"Adding packages to a schema" on page 56  Appendix A, "ClearQuest schemas and packages"
	Customize the schema as needed	Chapter 4, "Working with ClearQuest schemas" and Chapter 5, "Customizing a schema"
	Write hook code to further refine your workflow (optional)	Chapter 7, "Using hooks to customize your workflow"
Upgrade your user database with the selected/customized schema		Chapter 4, "Working with ClearQuest schemas"
Create ClearQuest login accounts and user groups and define user access permissions	Add ClearQuest users, subscribe them to a user database, create user groups, set permissions	Chapter 6, "Administering users"

<b>Task</b>	<b>Do this</b>	<b>To find out how, see</b>
Import data from other database systems (optional)	<b>Note:</b> You must set up user accounts before importing data from other database systems	Chapter 10, "Importing data into ClearQuest"
Enable e-mail notification (optional)	Have each ClearQuest user enable e-mail notification in ClearQuest	ClearQuest Help
	Set up e-mail rules	Chapter 9, "Administering ClearQuest E-mail"
Enable ClearQuest Web (optional)		<i>Installing Rational ClearQuest</i>
		Chapter 8, "Administering ClearQuest Web"
Customize ClearQuest queries, charts, and reports (optional)	Customize ClearQuest queries, charts, and reports and save them in the ClearQuest Public Queries folder	ClearQuest Help
	Alternatively, give permissions to other users, such as project managers, to do this	Chapter 6, "Administering users"



# 3

## Managing databases

ClearQuest uses relational databases to store the data about your process and forms (metadata) and the actual information you plan to track (records). This chapter describes how to maintain ClearQuest databases. The topics covered include:

- ClearQuest required databases
- Selecting a database vendor
- Creating new databases
- Overview of database maintenance
- Moving a database to a new location or to a new vendor
- Moving a database to a different schema
- Using the ClearQuest Export Tool
- Deleting a user database
- Undeleting a user database

## ClearQuest required databases

ClearQuest requires at least two databases:

- A schema repository: This is the master database where ClearQuest stores schemas.
- A user database: This is where data entered by ClearQuest client and web users is stored. You can have as many user databases as you need.

For a description of how ClearQuest databases and schemas work together, see “Working with ClearQuest schemas and databases” on page 17.

### Where to keep ClearQuest databases

Store ClearQuest databases on a machine dedicated solely to that purpose. Select a powerful machine with sufficient RAM and hard-disk space. The exact requirements depend on the number of users at your site, the number of records in your database, and whether you include large attachments with your records. Check to see if your database vendor has specific requirements.

### Maintaining database integrity

ClearQuest stores data using relational database functionality. However, the methods for storing and locating information in tables and for joining that information for presentation in the interface are specific to ClearQuest.

**Warning:** To preserve data integrity, use only ClearQuest tools to manipulate ClearQuest data. Do not use your database vendor tools to directly manipulate ClearQuest data or tables. If your data becomes corrupted, it will be extremely difficult to recover.



## Selecting a database vendor

ClearQuest supports the following databases:

- Entry-level database: Microsoft Access (supplied with ClearQuest)
- Mid-level database: Sybase SQL Anywhere (supplied with ClearQuest)
- High-end databases: Microsoft SQL Server and Oracle (NT and UNIX)

For specific supported versions, see *ClearQuest Release Notes* in `\Rational\doc\clearquest_readme.htm`.

To decide which database vendor is best for your needs, determine how many simultaneous users you expect to have. Use the following general guidelines:

- For initial testing and evaluation or for small groups (fewer than five users), you can use the entry-level database.
- For small- to medium-size groups (five to twenty users), you can use the mid-level database.
- For larger groups (over twenty users), large amounts of data, or to achieve better performance, use one of the high-end databases.

**Note:** You can use more than one database vendor for ClearQuest databases. For example, you might want to use high-end databases for your schema repository and user databases, then use an entry- or mid-level database for your test database.

## Creating new databases

Once you have selected a dedicated machine to use as the ClearQuest database server and have decided which database vendor to use, you are ready to create the required schema repository and user databases. For specific instructions on creating new databases, see *Installing Rational ClearQuest*.

**Note:** If you plan to use ClearQuest with a MS Access database, you do not need to have already set up this database; you can do so when you use ClearQuest.

**Note:** You must create your schema repository before you can begin customizing ClearQuest. You should also set up a test user database for testing new schema customizations. See “Working with a test database” on page 47.

Once your databases are set up, you can begin using and customizing ClearQuest. See Chapter 4, “Working with ClearQuest schemas”.

### Backing up your data

You never know when an equipment failure or software problems will occur that can cause data loss. Regular backups are the only way to ensure the safety of your ClearQuest data.

Develop a strategy for performing regular database backups, backing up your user databases and your schema repository at the same time. This preserves both your data and customizations.

You can use the database tool of your choice to perform backups. Most high-end databases come with a tool that you can use to establish regular backups.

**Note:** Although you can use the tool of your choice to back up ClearQuest databases, use only ClearQuest tools to directly modify ClearQuest data or tables.

## Overview of database maintenance

Below is an overview of database-maintenance activities and the ClearQuest tool or interface you should use to perform them.

Activity	Tool/Interface	Comments
Create a new schema repository or user database	Your database vendor tool, and	To create new empty databases, use the database vendor of your choice.
	ClearQuest Maintenance Tool	To connect and initialize the schema repository and SAMPL user database, use the ClearQuest Maintenance Tool.
	and ClearQuest Designer	To create new user databases, use ClearQuest Designer.  <b>Note:</b> You can create Microsoft Access and SQL Anywhere databases from within ClearQuest Designer.
Back up a database	Your database vendor tool	To preserve your data, perform regular backups.
Modify data	ClearQuest client, web interface, ClearQuest Designer, ClearQuest API	To maintain database integrity, <b>use only ClearQuest tools to directly modify ClearQuest data or tables.</b> Do not use your database vendor tools to modify ClearQuest data.
Use a newer version of a schema for a user database	ClearQuest Designer	See "Upgrading a user database" on page 54.
Change the schema for a user database	ClearQuest Export Tool and	Use the ClearQuest Export Tool to export the data from the user database. . .
	ClearQuest Import Tool	. . . then use the ClearQuest Import Tool to import the data into a new user database associated with a new schema.  See "Moving a database to a different schema" on page 34.

<b>Activity</b>	<b>Tool/Interface</b>	<b>Comments</b>
Move a database or change database vendors	ClearQuest Designer	To move a user database to a new location or to a different database vendor, use ClearQuest Designer.
	or ClearQuest Maintenance Tool	To move a schema repository to a new location or to a different database vendor, use the ClearQuest Maintenance Tool.  See "Moving a database to a new location or to a new vendor" on page 29.
Delete/undelete a user database	ClearQuest Designer	ClearQuest Designer removes the link between the schema repository and the database, but does not delete the physical database.  To delete the physical database, use your database vendor tool.  See "Deleting a user database" on page 42.
Upgrade to a new ClearQuest version	ClearQuest Maintenance Tool	To upgrade to a new version of ClearQuest, use the ClearQuest Maintenance Tool.

## Moving a database to a new location or to a new vendor

You can move a ClearQuest database to a new location or to a different database vendor. To move a schema repository, use the ClearQuest Maintenance Tool. To move a user database, use ClearQuest Designer.

The examples in this section show how to move a database from Microsoft Access to Microsoft SQL Server, but the same steps apply to converting from any one supported database to another.

### Preparing to move a database

Follow these procedures to prepare to move a schema repository or a user database to a new location or to a different database vendor.

- 1 Create a new empty database using your vendor tool (you can create Microsoft Access and SQL Anywhere databases from within ClearQuest Designer).

**Note:** See *Installing Rational ClearQuest* for information on creating new databases.

- 2 Notify all users to log off from ClearQuest.

ClearQuest prevents new users from accessing the databases during the process, but it cannot detect or log off users who are currently logged in when the procedure begins.

**Note:** Some high-end databases provide a tool for logging users off the database. Check to see if your database vendor has such a tool.

- 3 Back up all your databases.

**Warning:** Always back up your database before moving it or changing database vendors. The conversion process locks the original databases during conversion.

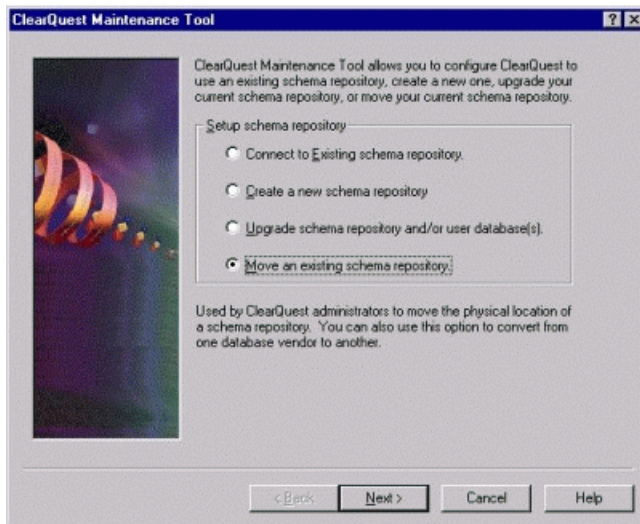
## Moving a schema repository

This example shows how to use the ClearQuest Maintenance Tool to move a schema repository called “CQMaster” to a new database vendor. You can follow the same procedure to move any schema repository to a new location.

**Note:** You must have Super User privileges to move a schema repository. See “ClearQuest user privileges” on page 99.

To move a schema repository to a new location:

- 1 Complete the steps in “Preparing to move a database” on page 29.
- 2 Select Rational ClearQuest Maintenance Tool from the Start menu.
- 3 In the Maintenance Tool, select **Move an existing schema repository** and click **Next**.



- 4 Enter your ClearQuest administrator login name and password, then click **Next**.

- 5 Provide the new schema repository properties, then click Next.

Move Schema Repository

Enter the properties of the new schema repository you wish to move/convert to. Note that once a database is moved, the previous version will be locked.

Source Schema Repository

Vendor: MS\_ACCESS Physical Database Name: \\cc3m\ChangeRequests\CQMast Server Name:

New Schema Repository Properties

Vendor: SQL\_SERVER

Physical Database Name: CQMaster

Database Server Name: cc3m

Administrator Name: admin

Administrator Password: \*\*\*\*\*

Read/Write User Name: admin

Read/Write User Password: \*\*\*\*\*

Read Only User: admin

Read Only User Password: \*\*\*\*\*

< Back Next > Cancel Help

- 6 When the conversion is complete, exit the ClearQuest Maintenance Tool.
- 7 Have all ClearQuest users run the ClearQuest Maintenance Tool to connect to the new schema repository. Be sure to provide the information they need to connect. See “Installing ClearQuest for end users” in the *Installing Rational ClearQuest* guide for details on the information needed to connect.

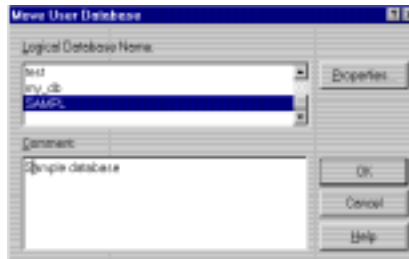
### Moving a user database

This example shows how to use ClearQuest Designer to move a user database called “SAMPL” to a new database vendor. You can use the same procedure to move any user database to a new location.

**Note:** If you have already moved the database outside of ClearQuest and only need to change the database properties, use the tasks described in “**Updating user databases**” on page 36.

- 1 Complete the steps in “Preparing to move a database” on page 29.

- 2 Select Rational ClearQuest Designer from the Start menu.
- 3 In ClearQuest Designer, select Database > Move User Database.
- 4 In the Move User Database property dialog box, select the user database and click Properties. (Note that if you click **OK** at this point you will only be updating any changes you have made in the comment area of the dialog box; you will *not* be moving the database.)



—— Click to open the Properties dialog

- 5 In the Properties dialog, select the new database vendor and fill in the properties for the new database, including whether the database is production or test. When you finish, click Move.





- 6 In the Moving Database dialog box, the message in the Status area informs you whether the database move is successful.



If you want to save the status message into an output file, click the **Save Status As...** button and specify a file name and path for the output file. Click **Save** and continue to the step 7.

If you do not want to save the Status message to an output file, continue on to step 7.

- 7 In the Moving Database dialog box, click **OK**. An update confirmation message appears:



- 8 In the Move User Database dialog box, click **OK**.

## Moving a database to a different schema

Once a user database is associated with a schema, you can only upgrade that database with newer versions of the same schema. You cannot apply an entirely different schema to the database.

For example, if a user database uses version 1 of the DefectTracking schema, you can upgrade that database to version 2 of the DefectTracking schema. You cannot upgrade that same database to use the Enterprise schema.

To move a database to a new schema, you must first create a new database and associate it with the desired schema. Then you use ClearQuest tools to export the data from the old database and import it into the new one.

To move a database to a new schema:

- 1** Before you begin, read “Important considerations when moving to a new schema” on page 35.
- 2** Create or modify the schema you want to start using.  
See Chapter 4, “Working with ClearQuest schemas” and Chapter 5, “Customizing a schema.”
- 3** Create a new user database and associate it with the new schema.  
Select Database > New Database in ClearQuest Designer to create a new user database. ClearQuest Designer prompts you to associate the new database with a schema.
- 4** Use the ClearQuest Export Tool to export the data from your current user database.  
See “Using the ClearQuest Export Tool” on page 38.
- 5** Use the ClearQuest Import Tool to import the data into the new database.  
See Chapter 10, “Importing data into ClearQuest” for information on using the Import Tool.

## Important considerations when moving to a new schema

Consider the following issues and requirements when moving a database to a new schema:

- After you create a new schema, be sure to test it on a test database before upgrading your new user database. Import a small subset of data and test the schema to make sure it works correctly. See “Working with a test database” on page 47.
- If your new schema has dynamic lists, be sure to populate those lists before importing your data. See “Working with choice lists” on page 117.
- If your data has dependencies for reference or reference\_list fields, you should import the data in the proper order. For example, you might need to import the project record type first, followed by the main data record type, then attachments, and finally history.
- If you are moving to a new schema repository, you must export user information from the old schema repository and import it into the new one before moving other data. You can use ClearQuest Designer to export user data. See “Exporting and importing users and groups” on page 110.
- If the new schema uses different state names than the old schema, you must edit the export file to change the state names. For example, if the old schema uses the state name *submitted* but the new schema calls that state *new*, you must edit the export file and replace *submitted* with *new* before importing that data into the new database.
- The ClearQuest Export Tool exports the record ID into the display\_name field for the history and attachments record types. (See “Working with record types” on page 64.) Remember to map the display\_name field to the original “old” ID field when importing data.

## Updating user databases

Use the **Update User Database Properties** menu option to change a selected database's properties if you have moved your user database outside of ClearQuest. (If you have *not* moved your database outside of ClearQuest, use the tasks described in "Moving a database to a new location or to a new vendor" on page 29 to change properties prior to moving the database.)

- 1 Click **Database > Update User Database Properties**.
- 2 Click the desired database name.

**To change the Comments field only:** use the **Comment** editor window for make your changes and click **OK**.

**To modify database properties:** click the **Properties** button to display the Database Properties dialog box, and make your changes. Click the **Update** button to save your changes.

## Viewing database properties

- 1 In ClearQuest Designer, click Database > View Database Properties.
- 2 From the Database Property dialog box, highlight the desired database name and click the Properties button to display the properties of the database.

**Note:** This windows read-only. To move and/or edit database properties, see “Moving a database to a new location or to a new vendor” on page 29 or “**Updating user databases**” on page 36.

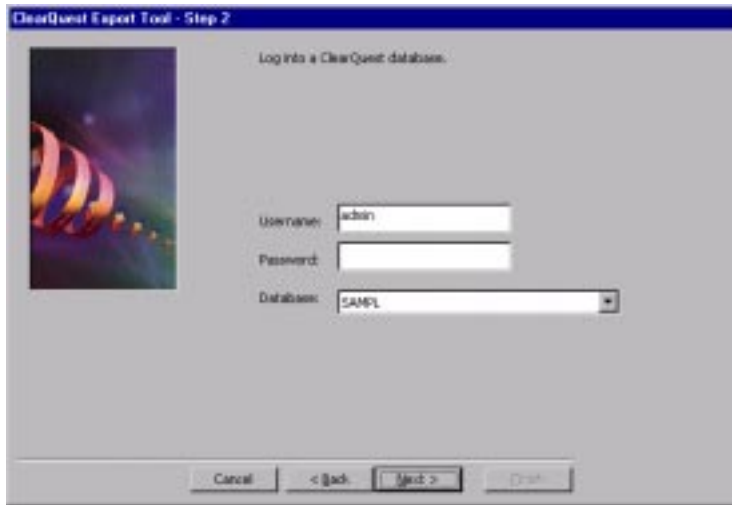
## Using the ClearQuest Export Tool

The ClearQuest Export Tool exports data from a ClearQuest database into an ASCII format optimized for the ClearQuest Import Tool. To run the Export Tool:

- 1 Select Rational ClearQuest Export Tool from the Start menu.
- 2 In the Export Tool, select the schema repository containing the database from which you want to export data. The default is the schema repository associated with the current version number of ClearQuest.



- 3 Log in and select the database you want to export.



ClearQuest Export Tool - Step 2

Log into a ClearQuest database.

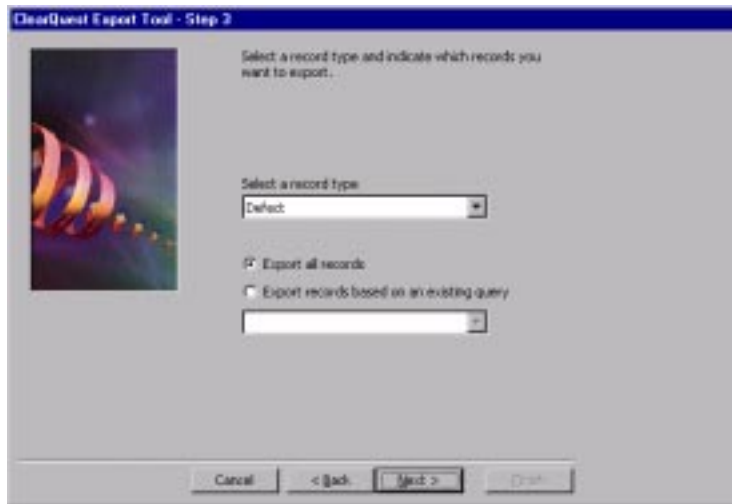
Username: admin

Password:

Database: SAMPLE

Cancel < Back Next > Finish

- 4 Select a record type and indicate which records you want to export. You can export all records or select a ClearQuest query to export specific records.



ClearQuest Export Tool - Step 2

Select a record type and indicate which records you want to export.

Select a record type: Default

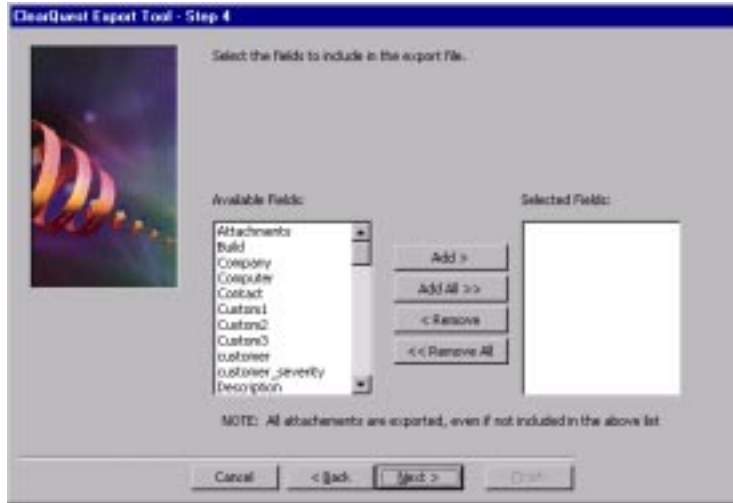
Export all records

Export records based on an existing query

Cancel < Back Next > Finish

The history, attachments, and duplicates data are automatically exported for the selected record type.

**5 Select the fields to export.**

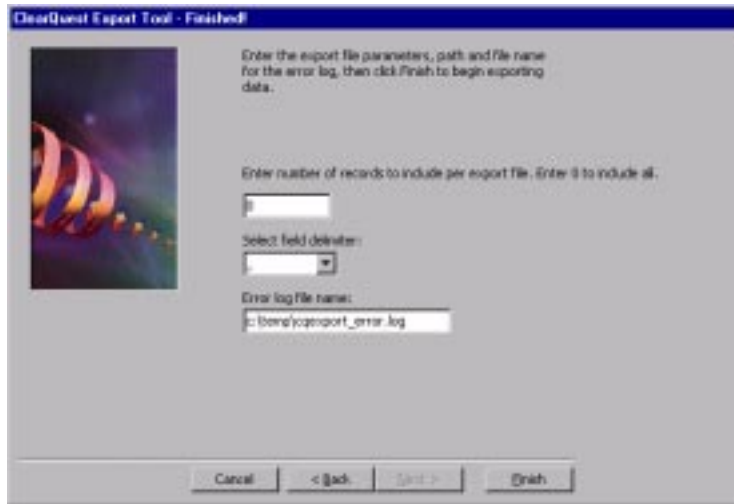


**6 Enter names for the export files. ClearQuest creates separate files for records, history, attachments, and duplicates. These are the files required by the ClearQuest Import Tool.**





**7** Enter the final export parameters.



- You can choose to include all records in one file or enter the number of records you want per file.
  - Select the field delimiter for the export files. The most common format is comma-delimited, but you can also use colons, semi-colons, pipes, or tabs.
  - Enter the path and name for the Error log.
- 8** Click Finish to begin the export. ClearQuest displays a dialog box when the export is complete and lists any detected errors.

## Deleting a user database

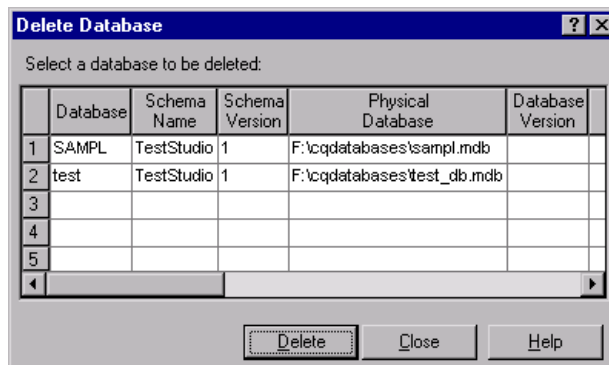
Deleting a user database from ClearQuest removes the link between the schema repository and the user database. It does not actually delete the physical database, nor does it delete the schema associated with that database.

You can restore a deleted database at a later date by restoring the link between the database and the schema repository. See “Undeleting a user database” on page 43.

**Note:** The link to a deleted database is permanently lost if you delete the schema version associated with the database or if you upgrade to a newer version of ClearQuest.

To delete a user database from ClearQuest:

- 1 Select Rational ClearQuest Designer from the Start menu.
- 2 Select Database > Delete Database to open the Delete Database dialog.



- 3 Select the database you want to delete and click Delete.

**Note:** You can physically delete the user database after you remove the link; however, if you do this you will not be able to undelete the database. To physically delete an Oracle or SQL Server database, use your vendor’s tools to remove the user database after deleting the link. For Microsoft Access and SQL Anywhere databases, you can manually remove the database files.

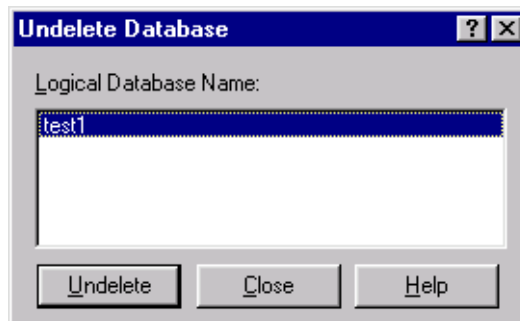
## Undeleting a user database

You can restore the link between a user database and the schema repository by undeleting the database. You must restore the link to the same schema version to which the user database was previously linked. The physical database must be in the same location that it was in when it was deleted.

**Note:** Undeleting a database should be done as soon as possible after you delete the database to avoid permanently losing its link to the schema repository. If you delete the schema version associated with the database or if you upgrade to a new version of ClearQuest, you cannot restore the link. Undeleting a database is possible only if the physical database still exists. You cannot undelete a database if you used your database vendor's tools to delete the physical database.

To undelete a user database:

- 1 Select **Rational ClearQuest Designer** from the Start menu.
- 2 Select **Database > Undelete Database** to open the Undelete Database dialog.



- 3 Select a deleted database from the list and click **Undelete**.



# 4

## Working with ClearQuest schemas

A ClearQuest schema contains the metadata (the record types, actions, states, fields, forms, and hooks) that define how you work with records in ClearQuest. ClearQuest includes several predefined schemas that provide common workflow models.

This chapter provides an overview of how to work with schemas, whether you are modifying an existing schema or creating a schema from scratch. The topics covered include:

- Overview of working with schemas
- Selecting a ClearQuest schema
- Adding packages to a schema

**Note:** To work with schemas, you need Schema Designer or Super User privileges. To learn how to set user privileges, see Chapter 6, “Administering users.”

**More information?** For a list of predefined schemas and packages, see Appendix A, “ClearQuest schemas and packages.” For information on how to customize a schema, see Chapter 5, “Customizing a schema.”

## Overview of working with schemas

To customize a schema, or to create a new schema, use the ClearQuest Designer and follow these basic procedures:

- 1 Create a new user database to be used as a test database and associate it with the schema you want to customize. See “Working with a test database” on page 47.
- 2 Check out a schema from the schema repository (see “Checking out a schema” on page 49), or create a new schema (see “Creating a new schema” on page 50).
- 3 Set the test database for the schema by selecting Database > Set Test Database. See “Working with a test database” on page 47.
- 4 Choose the scripting language to use for hook code. See “Selecting a scripting language” on page 51.
- 5 Customize the schema, validating often, especially when making complex changes. See “Validating schema changes” on page 52.  
**Note:** A quick way to customize a schema is to add ClearQuest packages to get the functionality you need. See “Adding packages to a schema” on page 56.
- 6 Save your work in progress by selecting File > Save Work. You can save your work and log off while a schema is checked out to you, then resume work when you log back in. This does not create a new schema version or affect your user database.
- 7 Test your work by selecting File > Test Work. This is a safe and easy way to test your work in the ClearQuest client without affecting your user database. See “Working with a test database” on page 47.
- 8 Repeat steps 5 through 7 until you are satisfied with your schema.
- 9 Check the schema into the schema repository to save the new version of the schema. See “Checking in a schema” on page 53.
- 10 Upgrade the user database to the new schema version. Select Database > Upgrade Database and select the user database. See “Upgrading a user database” on page 54.

## Working with a test database

You should set up a test user database for each schema you plan to customize. By using a test database, you can quickly and safely test your schema customizations in the ClearQuest client without affecting your production user database.

To work with a test database:

- 1 Create a test user database for the schema you plan to customize. In ClearQuest Designer, select **Database > New Database**.

The New Database Wizard prompts you through the process. Create a new database, giving it a name you can recognize as a test database, then associate the test database with the schema you want to customize.



**Note:** You can select **Visible to the designer only** so that the test database is not visible to ordinary ClearQuest client users. If you select **Visible to the designer only**, the test database will not be visible from ClearQuest UNIX, ClearQuest Web, or from the ClearQuest Import Tool.

- 2 Check out the schema you want to customize, then select **Database > Set Test Database** and select the test database you associated with this schema.
- 3 At any time while you are customizing the schema, select **File > Test Work**. ClearQuest saves and validates your schema changes, reporting any errors in the Validation pane at the bottom of the ClearQuest Designer window. See “Validating schema changes” on page 52.

ClearQuest then upgrades your test database with the current schema version. On Windows, ClearQuest automatically starts the ClearQuest client so you can test how the schema performs. To test the schema from the UNIX or Web clients, log into the test database manually. (Remember, the test database will not be visible to the UNIX or Web clients if you selected **Visible to the designer only** when you created the database.)

**Note:** You must keep your test database current with your latest schema version by selecting **File > Test Work**. If you attempt to check in a schema without testing your work, ClearQuest prompts you to test your work first.

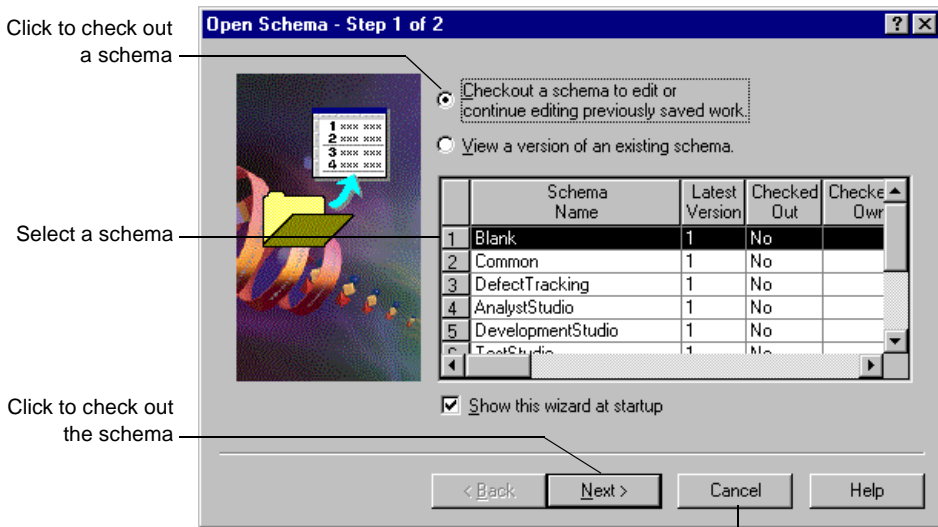
**More information?** Look up *test databases*, *setting* and *schemas*, *testing* in the ClearQuest Designer Help index.



## Checking out a schema

ClearQuest maintains schemas under version control in the schema repository. To customize a schema, you must first check out the schema from the schema repository. When you check out a schema, ClearQuest creates a new version of the schema for you to edit.

After you log into ClearQuest Designer, the Open Schema dialog is automatically displayed:



Or, click to start ClearQuest Designer without checking out a schema

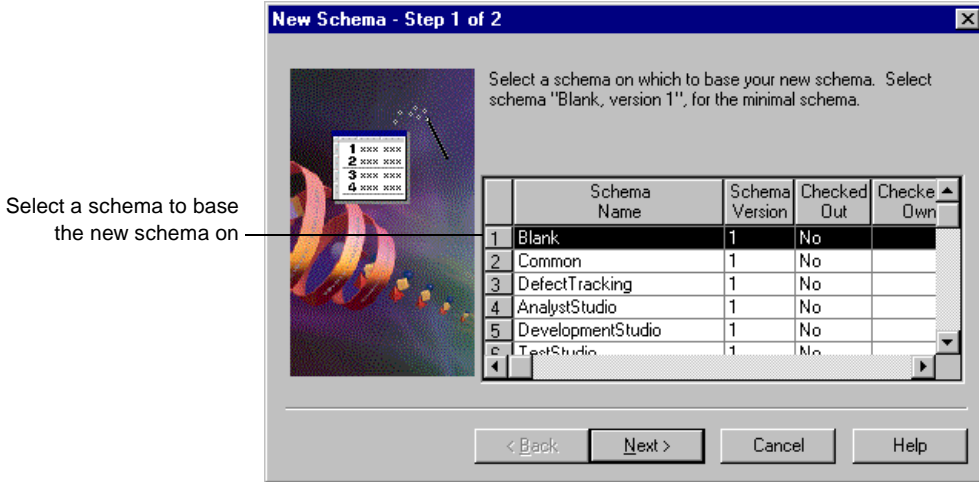
**Note:** If you open a schema for viewing only, the schema is read-only; you cannot customize it .

You can also check out a schema from within ClearQuest Designer by selecting **File > Open Schema**.

## Creating a new schema

To create a new schema, you must base the new schema on an existing schema.

In ClearQuest Designer, select **File > New Schema** and select a schema to base the new schema on.



To create a new schema from scratch, select the **Blank** schema, which contains only system fields.

## Selecting a scripting language

Hooks are triggers for pieces of code that ClearQuest executes at specified times to more fully implement your workflow. You can write hooks in VBScript (for Windows) and Perl (for Windows and UNIX).

After checking out a schema, you can select the scripting language that you want ClearQuest to use to run scripts on Windows. The default is BASIC (VBScript).



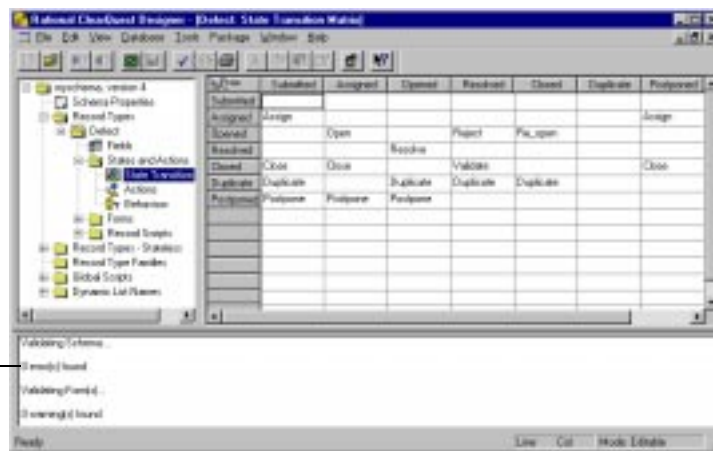
**More information?** See Chapter 7, “Using hooks to customize your workflow.” For information on the scripting languages themselves, see the following resources on the Internet:

- <http://msdn.microsoft.com/scripting/>
- <http://www.perl.com/>

## Validating schema changes

You cannot check in a schema if there are any validation errors. When you attempt to check in a schema, ClearQuest automatically validates your schema to ensure that it does not contain errors. If you are making complex changes to a schema, it's a good idea to check for errors often by validating your schema manually.

To validate a schema, select **File > Validate**. ClearQuest displays the validation results in the Validation pane at the bottom of the ClearQuest Designer window.



Review the validation results and modify the schema as needed to correct any errors. If you cannot validate all of your changes, you can save your schema changes and continue editing at a later time. You can also undo a schema checkout to revert the schema to its previous version. See “Undoing a schema checkout” on page 53.

**Note:** Validation does not guarantee that a schema will function as desired. Be sure to test a schema with a test database before checking it in, and back up your user database before upgrading it. See “Working with a test database” on page 47.

**More information?** Look up *schemas*, *validation* in the ClearQuest Designer Help index.

## Checking in a schema

After editing and testing a schema, you must check the schema back into the schema repository. When you check in a schema, ClearQuest saves the new version of the schema in the schema repository.

Checking in a schema does not affect the user database; you must upgrade the user database with the new schema version to have your changes take effect. Once the new schema version is checked in, you can use it to upgrade an existing database or apply it to a new database. ClearQuest keeps a history of each version, so you can view a prior version and use it in a new database.

To check in a schema, select **File > Check In**. ClearQuest validates the schema, displaying any errors in the Validation pane at the bottom of the ClearQuest Designer window.

**Note:** You cannot check in a schema if there are validation errors. See “Validating schema changes” on page 52.

## Undoing a schema checkout

You can return a schema to its previous version even if you have changed the schema and saved your changes. Undoing a schema checkout reverts the schema to the last checked-in version, removing all changes made since the schema was checked out, even if they were made and saved over multiple sessions.

To revert to the last checked-in version of a schema, select **File > Undo Check Out**.

**Note:** If you test your work by selecting **File > Test Work**, ClearQuest automatically upgrades your test database to the new schema version. If you then want to undo the schema checkout, you must delete the test database before ClearQuest will allow you to undo the schema checkout. You will then need to create a new test database.

## Upgrading a user database

After checking in a schema, you must upgrade the user database with the new schema version for your changes to take effect.

**Note:** Before upgrading, back up your user database.

To upgrade a user database, in ClearQuest Designer select Database > Upgrade Database.

Keep in mind the following restrictions when you upgrade a user database:

- Once a user database is associated with a schema version, it can only be upgraded to newer versions of the same schema. It cannot use an earlier version of the schema or to a different schema.
- Before upgrading a user database, make sure that there are no users connected to the database. ClearQuest prevents new users from connecting to a database while you are upgrading it, but it does not disconnect users who are currently connected.

**More information?** Look up *databases, upgrading* in the ClearQuest Designer Help index. Also see Chapter 3, “Managing databases.”

## Deleting a schema

You can delete a single version of a schema or all versions of a schema:

- To delete a single version of a schema from the schema repository, select File > Delete Schema Version and select the schema version you want to delete.
- To delete *all* versions of a schema from the schema repository, select File > Delete Schema and select the schema to delete.

**Note:** You cannot delete any schema or schema version that is currently associated with a user database. You must first delete the user database. All schema deletions are final. You cannot retrieve a deleted schema.

## Selecting a ClearQuest schema

ClearQuest provides predefined schemas that you can use as is or customize to suit your workflow. ClearQuest schemas are made up of predefined schema *packages*. A package is a piece of schema functionality. For example, the Email package enables ClearQuest to send automatic e-mail notifications.

You can add predefined packages to your own customized schema, or use them to enhance ClearQuest predefined schemas. See “Adding packages to a schema” on page 56.

ClearQuest includes the following predefined schemas:

Schema	Description
AnalystStudio	Compatible with Rational Suite AnalystStudio. Contains customization for use with Rational RequisitePro.
Blank	Contains only system fields. Use the Blank schema to create a schema from scratch.
Common	Contains metadata that is common to all of the ClearQuest schemas.
DefectTracking	Contains the fields necessary to start using ClearQuest to track defects in a software-development environment.
DevelopmentStudio	Compatible with Rational Suite DevelopmentStudio. Contains fields and rules that work with Rational’s Purify, Quantify, and Pure Coverage.
Enterprise	For use with Rational Suite Enterprise. Contains fields and hooks that work with all Rational products.
TestStudio	Compatible with Rational Suite TestStudio. Contains fields and rules that work with Rational’s TeamTest, RequisitePro, Purify, Quantify, and Pure Coverage.
UnifiedChangeManagement	Supports the UCM process by providing integration with Rational ClearCase. See Appendix B, “Enabling ClearQuest for Unified Change Management”.

**More information?** See Appendix A, “ClearQuest schemas and packages” for a complete description of schemas and packages.

## Adding packages to a schema

ClearQuest's predefined schema packages contain metadata such as records, fields, and forms that define specific functionality. Installing a package into a schema is the easiest way to add functionality to the schema.

A package might contain one or more new record types and might modify one or more record types that already exist in your schema. For example, the Email package adds the stateless Email\_Rule record type to the schema and adds the Send\_Email\_Notif action to an existing record type in the schema.

For a complete list of packages and the record types and fields they modify, see "ClearQuest packages" on page 4.

Keep in mind the following when adding packages to schemas:

- You can add one or more packages to any schema.
- A schema *cannot* already contain the metadata that the package adds. For example, attempting to install a package containing a Project record type into a schema that already contains a Project record type will cause the installation to fail.
- If you are upgrading to a new version of ClearQuest, install the latest version of the package(s).

**Note:** You cannot create your own packages to use with the Package Wizard. However, you can export schemas or schema versions and then import them into another schema repository using the CQLOAD command line utility. For more information, look up *cqload* in the ClearQuest Designer Help index.

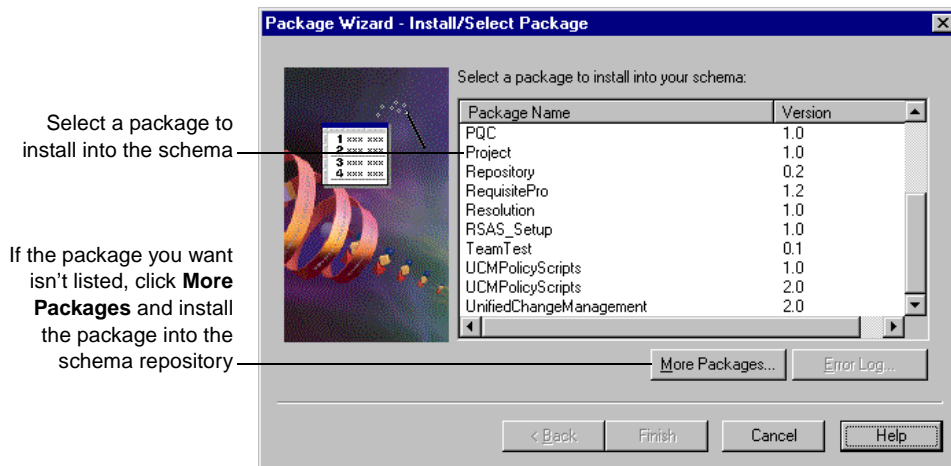


To add a package to a schema, you use the Package Wizard. You can begin with a schema that is checked in or checked out of the schema repository. (The Package Wizard automatically checks out the schema if necessary to add the package.)

The following instructions assume that you are beginning with a schema that is checked in:

1 Add the package.

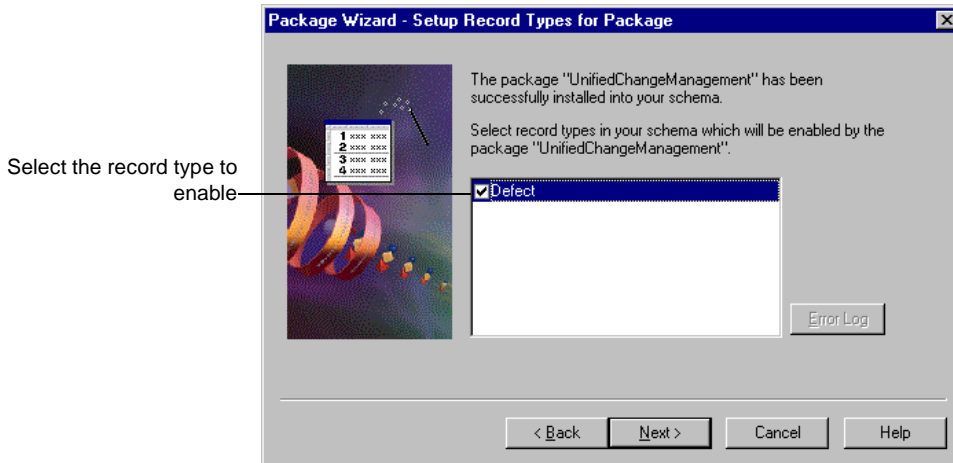
Select **Package > Package Wizard** and select a package to install.



**Note:** Packages that are already part of your current schema do not appear in the list of packages. Also, packages that are not already installed in the schema repository are not listed. Click **More Packages** to first install packages into the schema repository.

## 2 Enable record types.

Some packages modify record types that already exist in your schema. You can select the record types to enable for the package.



For descriptions of packages, including the record types they modify, see “ClearQuest packages” on page 4.

**More information?** Look up *record types, setting up for a package* in the ClearQuest Designer Help index.

**3** Some packages prompt you to map state types.

A state type is a label that packages such as the Resolution package and the UnifiedChangeManagement package use to define a state's role in your state model. When you add a package that uses state types, you are prompted to map each of your existing states to a state type in that package. You can map more than one state to a state type, but each state type requires at least one state.



State-type models are listed in “State-type models for packages” on page 13. To learn how to map state types, see “Mapping state types” on page 82.

**More information?** Look up *state types, setting up for a package* in the ClearQuest Designer Help index.

**4** If you are installing the UnifiedChangeManagement package, you must also create default actions. See “Using default actions” on page 89.



# 5

## Customizing a schema

This chapter describes how to use ClearQuest Designer to customize a schema. The topics covered include:

- Planning schema customizations
- Working with record types
- Working with fields
- Defining your state model
- Working with actions and action types
- Working with record forms

ClearQuest includes several predefined schemas that you can use without modification. For a complete list of ClearQuest predefined schemas and packages, see Appendix A, “ClearQuest schemas and packages.”

Before you can customize a schema, you must check out the schema from the schema repository. For instructions on how to work with schemas, see Chapter 4, “Working with ClearQuest schemas.”

To customize a schema, you need Schema Designer or Super User privileges. For more information, see Chapter 6, “Administering users.”

## Planning schema customizations

Customizing a schema is a multistep process involving these basic procedures:

### 1 Define your data.

To define the data you want to collect and manage in ClearQuest, you work with record types and fields:

- Decide what type of records you need. You may want separate record types for hardware defects and software defects, each of which would have its own set of fields. See “Working with record types” on page 64.
- List the fields you need on each record form. You can use fields to link records and you can customize your record type through field data types and hooks. See “Working with fields” on page 70.

**Note:** A quick way to add record types and fields to a schema is to add one or more ClearQuest predefined packages to the schema. See “Adding packages to a schema” on page 56.

### 2 Define your state model.

Each record type has its own state model. A state model consists of states, state types, actions, and field behaviors. To define your state model:

- List the states that you need and the actions that can be performed on the record in each state. Draw a state-model diagram showing how records will move through your system as the result of actions performed on them. For example, in a simple defect tracking system a record might move from the submitted state to opened, to fixed, and then to the verified state. See “Defining your state model” on page 80.
- Create the states and the actions you need. You can customize your workflow by adding action hooks. See “Working with actions and action types” on page 84 and “Customizing actions by adding hooks” on page 88.

- Use field behaviors to associate fields with specific record states. You can define whether the field is mandatory, optional, or read only in each state. See “Defining field behavior” on page 73.

**Note:** Some ClearQuest packages, such as the Resolution package and the UnifiedChangeManagement package, use state types to define a state’s role in your state model. If you add these packages to your schema or add a new state to a schema that uses state types, you must map the new state to a state type. See “Mapping state types” on page 82.

### 3 Build your record forms.

Create forms for your users to submit new records and to work with records. Decide on the tabs you will need on the forms and where to locate each field. Using form controls, add the fields you’ve created to the record form. See “Working with record forms” on page 90.

### 4 Add hooks if desired.

- Define the relationships between fields, determining whether the value in one field affects the values in other fields. Understanding these relationships will help you decide how to use hooks. See “Customizing fields by adding hooks” on page 79.
- Plan additional controls or permissions to enhance how your record form is used. For example, decide who should be notified when a record is submitted or changed and who can perform certain actions. See “Customizing actions by adding hooks” on page 88.

**More information?** See Chapter 7, “Using hooks to customize your workflow.”

## Working with record types

A record type is a category of change request to be managed by ClearQuest. A record type contains everything you need to define the data you want to collect and track: the states a record can be in, the actions that can be performed on the record, its fields, forms, and hooks.

You can have one or more record types in a schema. For example, you can create a simple schema that has a single Defect record type, or you can create a schema that has multiple record types such as Hardware Defects, Software Defects, and Enhancements.

ClearQuest supports state-based and stateless record types.

**Note:** ClearQuest packages can add and modify record types. For more information, see “ClearQuest packages” on page 4.

### State-based record types

State-based record types use states such as Submitted, Assigned, and Resolved to track their status as users work with them. A record of this type is always associated with a state. The record moves from state to state through the actions performed on it.

**Note:** You can group two or more state-based record types as a record type family to allow ClearQuest users to query across multiple record types. See “Adding a new record type family” on page 67.

### Stateless record types

Stateless record types are typically used for reference, such as a list of users, projects, or other information. Records of this type do not move from state to state; the only actions you can perform on a stateless record are Submit, Modify, Delete, and Import.

ClearQuest maintains four stateless system record types: History, Attachments, Groups, and Users. You cannot delete system record types.

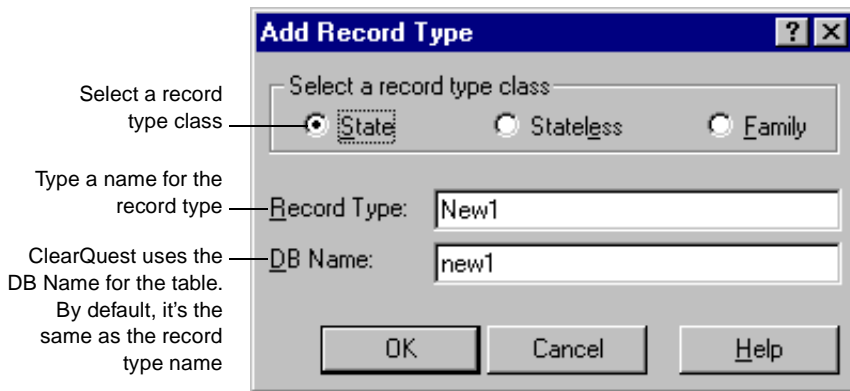


## Adding a new record type

You can add new record types to a schema. You must create each record type as either state-based or stateless; once you create the record type, you cannot change whether it is state-based or stateless.

To add a new record type:

- 1 In ClearQuest Designer, select **Edit > Add Record Type/Family**.
- 2 In the Add Record Type dialog, select the class for the record type and give the record type a name.



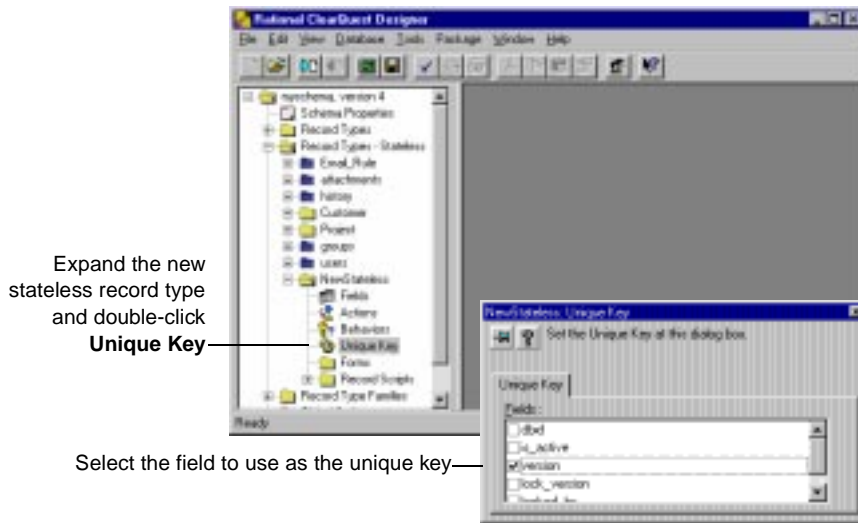
- 3 If you are adding a stateless record type, you must select one or more of its fields to be the unique key. See “Selecting a unique key for a stateless record type” on page 66.

**Note:** You can use ClearQuest predefined packages to add record types to a schema; packages can add record types and enhance existing record types by adding fields, hooks, forms, and states. See “Adding packages to a schema” on page 56.

## Selecting a unique key for a stateless record type

When you add a new stateless record type to a schema, you must select one or more of its fields to be the unique key. ClearQuest uses the unique key to differentiate among stateless records of a given type.

To select the unique key for a stateless record type:



**More information?** Look up *unique keys* in the ClearQuest Designer Help Index.

## Selecting a default record type

Each schema must have a default record type. Default record types can be state-based or stateless. ClearQuest uses the default record type to create a shortcut button in the ClearQuest client that can be used for submitting records of that type. ClearQuest also uses the default record type in situations where no other record type is explicitly specified.

To make a record type the default, right-click the record type in the Workspace and select **Default Record Type** from the shortcut menu.

## **Adding a new record type family**

You can create a record type family to group two or more state-based record types together. This allows ClearQuest users to query across multiple record types.

Keep in mind the following when creating a record type family:

- The record types included in a family must contain one or more common fields, that is, fields that are the same in each record type. These common fields are the only fields that can be used to query the record type family.

Common fields must have the same name and be of the same data type. For example, if you group a Defect record type and an Enhancement record type, you can use the Description field in each record type as the common field as long as that field is the same data type (perhaps short string) in both record types.

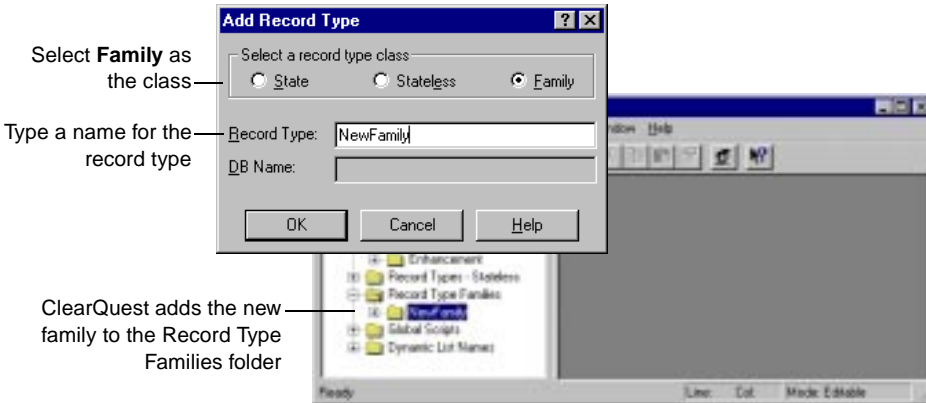
To learn how to add fields to a record type and to define their data type, see “Working with fields” on page 70.

- Since record types and record type families appear in the same window when ClearQuest users select **Query > New Query**, use a naming convention that will help users distinguish individual record types from record type families.

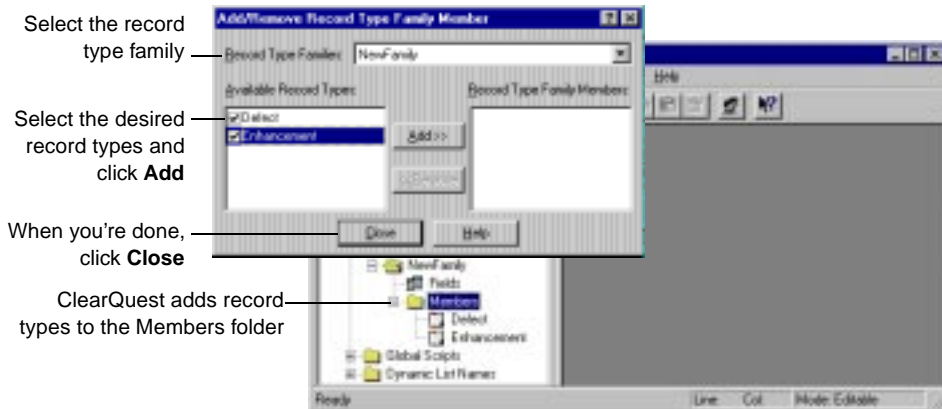
To add a new record type family to a schema:

1 In ClearQuest Designer, select **Edit > Add Record Type/Family**.

2 In the Add Record Type dialog:



3 Select **Edit > Record Type Family Members** and add the desired record types to the family.



4 Add common fields to the member record types to allow ClearQuest users to query the new family. See “Working with fields” on page 70.

**More information?** Look up *record type families, adding* in the ClearQuest Designer Online Help index.

## Renaming a record type

To rename a record type or family, select **Edit > Rename Record Type/Family** and type a new name.

**Note:** When you change the name of a record type, you must also change its name in any hooks that reference the record type. If you do not edit your hooks, your scripts will not work as intended.

**More information?** Select **Defining record types > Renaming record types** in the ClearQuest Designer Help.

## Deleting a record type

To delete a record type, select **Edit > Delete Record Type/Family** and select a record type or family to delete.

The following restrictions apply to deleting record types:

- You cannot delete system record types.
- You cannot delete the default record type. You must first assign another record type to be the default.
- If you have explicitly referred to the name of a record type in a hook, you must modify your script code to remove any references to that name.
- Record types added by read-only packages cannot be deleted.

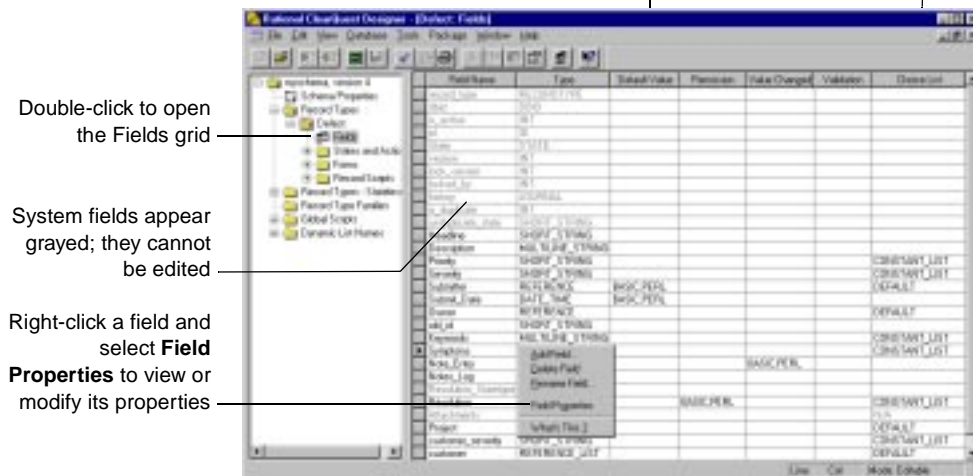
## Working with fields

You use fields to control the type of data that users can add to a user database. You can do the following with fields:

- Define field behavior
- Add Help text to a field
- Link related records
- Customize a field by adding hooks

Each record type has a Fields grid that shows the fields associated with that record type. To display the Fields grid, expand a record type in the Workspace and double-click Fields. Each field is displayed in a row, and its properties are displayed in columns.

Use these columns to add hooks to fields



Double-click to open the Fields grid

System fields appear grayed; they cannot be edited

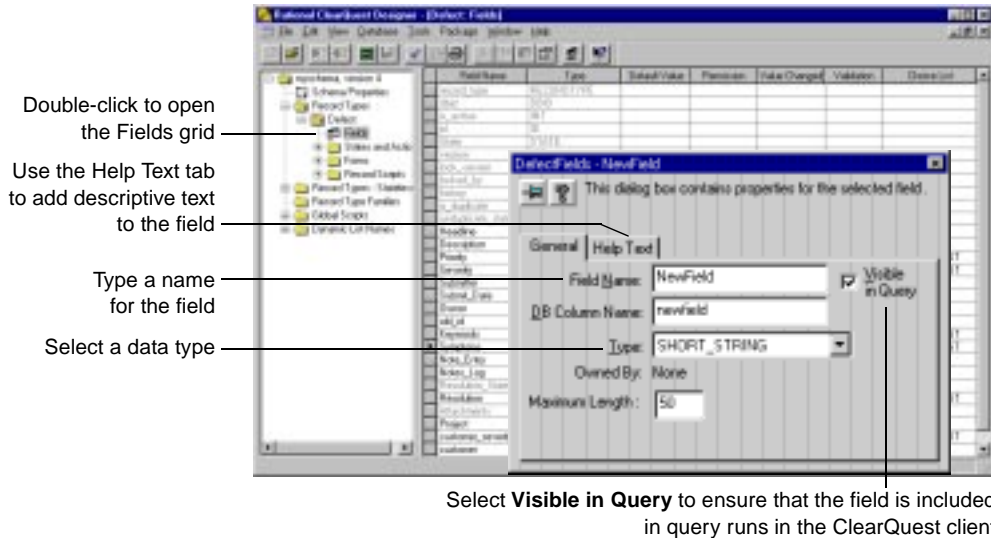
Right-click a field and select **Field Properties** to view or modify its properties

You can use the Fields grid to add new fields to the record type and to modify the properties of existing fields.

Each ClearQuest record type includes system fields. These fields are required for every record of that type. System fields appear grayed in the Fields grid.

## Adding a new field

To add a new field, open the Fields grid and select **Edit > Add Field**.



The DB Column Name is the name ClearQuest uses for the table column. By default it is the same as the field name.

**Note:** To make a new field available to users, you must add the field to the record form. See “Adding a field to a form” on page 93.

### *Adding Help text to the field*

Use the Help Text tab to add text to the field that describes the field or instructs your ClearQuest client users how to use the field. The Help text limit is 254 characters, approximately two-thirds of the space available on the Help Text tab.

ClearQuest users can right-click the field on the record form and select **Help** from the shortcut menu to view the Help text.

## **Selecting a field data type**

When you add a field, you must select its data type. The data type defines the type of data that can be entered in the field.

ClearQuest supports the following field data types:

<b>Data</b>	<b>Description/Comments</b>
ATTACHMENT_LIST	Allows records to store files related to the record.
DATE_TIME	SQL date and time.
INT	SQL integer.
MULTILINE_STRING	A variable-length string of unlimited size.
REFERENCE	<p>A reference to a unique key in a record type.</p> <p>For REFERENCE type fields, you must select a state-based or stateless record type to refer to. You can also enter an optional back-reference field to create a link from the referenced record back to this field's record.</p>
REFERENCE_LIST	<p>Multiple references to unique keys in record types. Reference-list fields allow you to reference multiple records within a field. You can use reference-list fields in conjunction with a parent/child control to link related records.</p> <p>For REFERENCE_LIST type fields, you must select a state-based or stateless record type to refer to. You can also enter an optional back-reference field to create a link from the referenced record back to this field's record.</p>
SHORT_STRING	A variable-length character string with a 254-character maximum. You set the length in the Properties dialog box when defining the field. Enter a value between 1 and 254 in the Maximum Length field.

**Note:** You cannot modify the data type of a field after you check in the schema. To change the data type, delete the existing field and create a new field with the data type you want.



## Defining field behavior

Each field has one or more behaviors associated with it. Fields in a state-based record type can have a different behavior for each state. For example, a field can be optional in the Opened state and mandatory in the Resolved state. Fields in a stateless record type need only one behavior for each field.

ClearQuest supports the following field behaviors:

Behavior	Description
Mandatory	The user is required to enter a value in this field before applying the changes to a record. Failure to do so will result in a runtime validation error. Mandatory fields show up in red on the record form.
Optional	The user can enter data into this field but is not required to do so. <b>Note:</b> Optional is the default setting for new fields.
Read only	The user can view the contents of the field but cannot modify them. <b>Note:</b> Hooks can modify Read only fields.
Use_hook	Use the field's permission hook to determine the level of user access.

To define the behavior for a field, use the Behaviors grid. In the Workspace, expand Record Types > Defect > States and Actions and double-click Behaviors.

Double-click to open the Behaviors grid

Right-click the state cell for a field and select a behavior

Substate	Assigned	Opened	Resolved	Closed	Duplicate	Proposed	Default Behavior
Mandatory	Mandatory	Mandatory	Mandatory	Mandatory	Mandatory	Mandatory	Mandatory
Optional	Optional	Optional	Optional	Optional	Optional	Optional	Optional
Read Only	Read Only	Read Only	Read Only	Read Only	Read Only	Read Only	Read Only
Use Hook	Use Hook	Use Hook	Use Hook	Use Hook	Use Hook	Use Hook	Use Hook

You can use the Default Behavior column to set a default behavior for a field. The default behavior applies to the field in every state and is automatically applied to the field when you add a new state.

**Note:** You can also set the behavior of a field by using a hook. Hooks operate using Super User privileges and therefore can modify any field, even if the field behavior is Read Only.

**More information?** Look up *fields: behaviors* in the ClearQuest Designer Help index.

## Changing the name of a field

You can change the name of a field. However, if you explicitly refer to the field by its name in a script, be sure to update your script to use the new name.

To change the name of a field, open the Fields grid, right-click the field, and select Field Properties. Type a new name for the field.

## Deleting a field

To delete a field, use the Fields grid. Select the row that contains the field that you want to delete and select Edit > Delete Field.

The following restrictions apply to deleting fields:

- To delete a field, you must also delete the field from the record form. See “Editing forms” on page 94.
- You cannot delete, rename, or modify a system field. System fields appear dimmed in the Fields grid.
- When you delete a field, ClearQuest retains the DB Column Name it generated for the field (by default, the same as the field name). If you later reuse the same name to create a new field, ClearQuest will generate a unique DB Column Name.
- If you explicitly refer to the name of a field in script code, you need to remove any references to the field from your script.
- In the ClearQuest client, any queries that use this field will become invalid.

## Using fields to link records

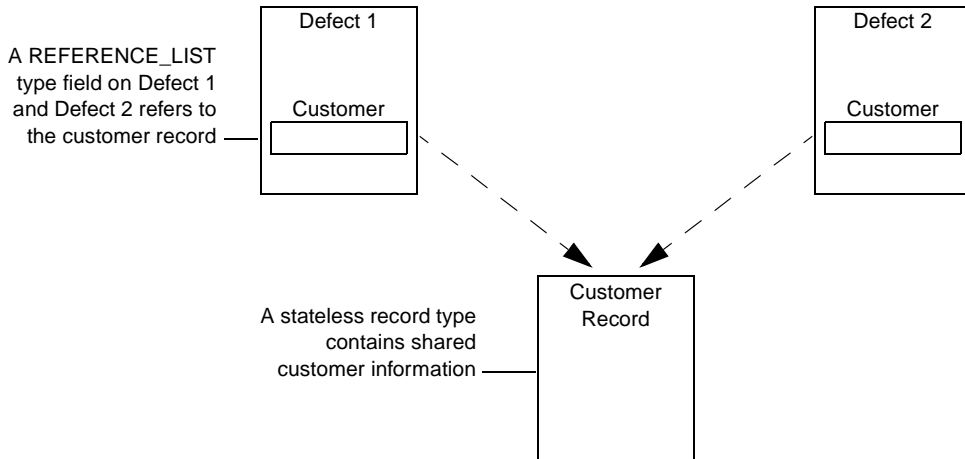
You can use fields to link records of the same type or records of different types. You can link records to:

- Share common data
- Create a parent-child hierarchy

### *Linking records to share common data*

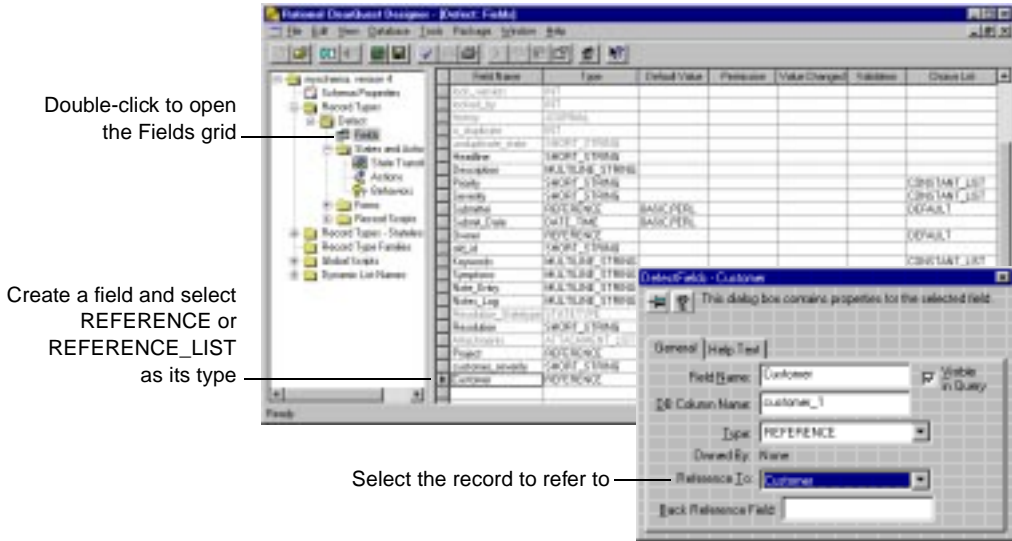
You can use REFERENCE or REFERENCE\_LIST type fields to link records in order to share common data. Use a REFERENCE field to link one record; use a REFERENCE\_LIST field to link multiple records.

For example, you might have the same customer data that must be entered for multiple records.



To link records to share common data:

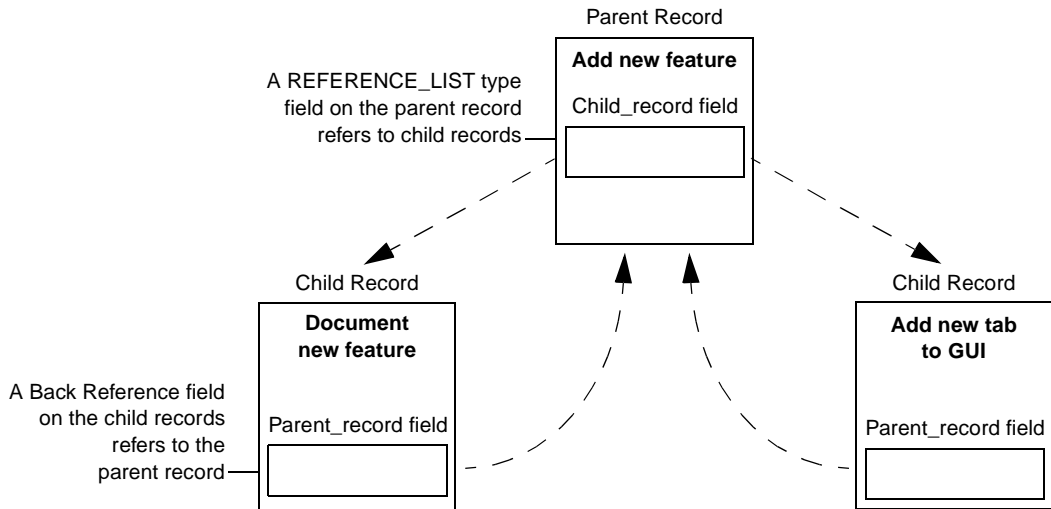
- 1 Open the Fields grid for the record type and create a new field, selecting REFERENCE or REFERENCE\_LIST as its type.
- 2 Select Edit > Field Properties and select the record type to refer to.



- 3 To make the field available to users, you must add it to the record form. Use a parent/child control with a REFERENCE\_LIST field type. See “Adding a field to a form” on page 93.

### ***Linking records to create a parent/child hierarchy***

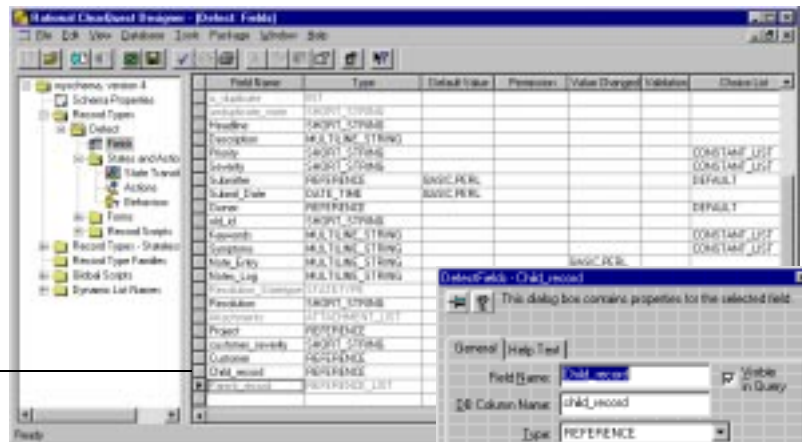
You can use REFERENCE or REFERENCE\_LIST type fields to link records of the same type to create a parent/child hierarchy. For example, you can relate a parent record that requests the addition of a new feature to one or more child records that describe related tasks, such as documenting the new feature and adding a new tab to the interface.



To link records to create a parent/child hierarchy:

- 1 Add a REFERENCE\_LIST or REFERENCE type field to the parent record.
- 2 Select the field and select Edit > Field Properties.
- 3 Enter a name for a Back Reference field. A Back Reference field is a read-only field that makes traversing parent-child relationships easier by allowing you to view the link from the child record point of view. This field is automatically added to the field list of the child record.

Add a field and select REFERENCE or REFERENCE\_LIST as its type



Select the record type to refer to

Type a name for the Back Reference Field. A Back Reference Field is read-only so it appears grayed in the Fields grid.

- 4 Add the new fields to the record forms. Use a parent/child control with a REFERENCE\_LIST field type. See “Adding a field to a form” on page 93.

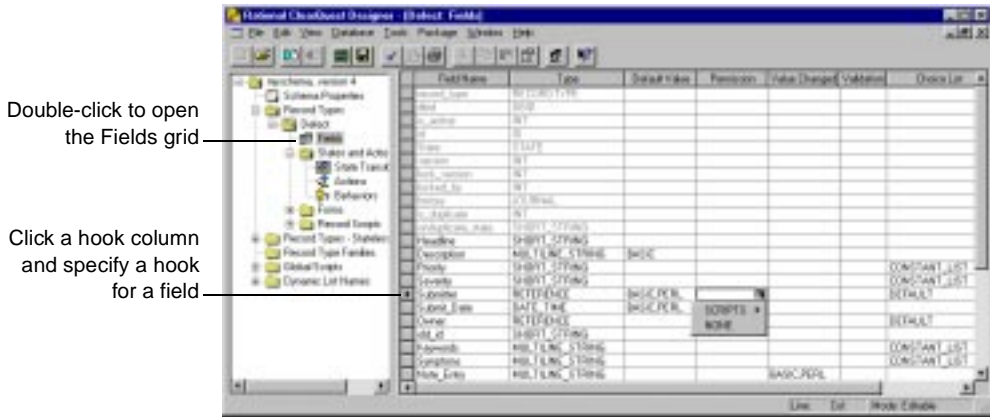
**More information?** Look up *records, linking related* in ClearQuest Designer Help index. See also “Action hook for setting the value of a parent record” on page 138 of this guide.

## Customizing fields by adding hooks

Hooks allow you to customize how fields work. For example, you can customize the schema so that field default values are assigned whenever someone submits a new record.

ClearQuest provides several field hooks: Default Value, Permission, Value Changed, Validation, and Choice List.

To define a field hook, use the Fields grid.



To create dependent fields, use a Value Changed hook (see “Creating a dependency between fields” on page 115). To find out how to create a choice list for a field, see “Working with choice lists” on page 117.

You can customize ClearQuest hooks by incorporating scripts that use the ClearQuest API. When you finish editing a scripted hook, select **Hooks > Compile** to check the syntax of your code.

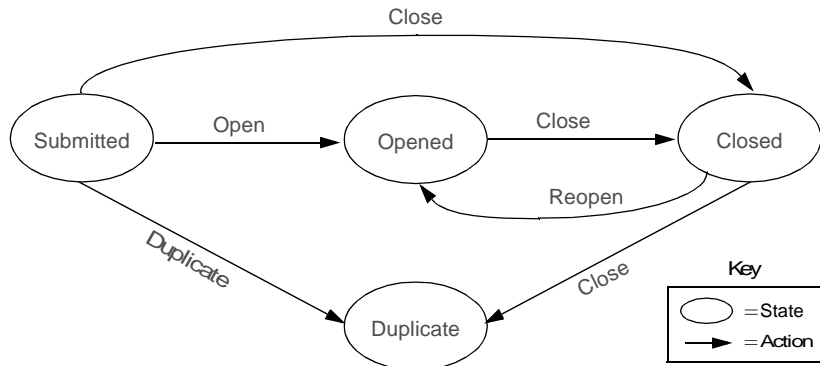
**More information?** For a complete description of field hooks and information about how they work with action hooks, see Chapter 7, “Using hooks to customize your workflow.” Also see “Selecting a scripting language” on page 51. Look up *field hooks* in the ClearQuest Designer Help index.

## Defining your state model

ClearQuest uses *states* such as Submitted, Opened, and Closed to define a record's status as it moves through your system. To work with a record, users perform *actions* such as Submit, Modify, and Close on the record.

A state model is a diagram that shows how ClearQuest users can work with a particular record type. It shows all of the states that a record of that type can be in and the actions that can be performed on the record in each state.

For example, the state model below shows how the EnhancementRequest record type (included in several predefined schemas) moves from one state to another as the result of the actions performed on it.



Begin designing a state model by listing the states you want and describing them. For example, the following table describes the states for the EnhancementRequest record type.

State	Description
Submitted	First state of a new record
Opened	Record is being worked on
Closed	Record fix has been verified
Duplicate	Record duplicates another record



## Using the State Transition Matrix

The movement of a record from one state to another is called a *state transition*. A state transition consists of a source state, a destination state, and the action that moves the record from the source state to the destination state. A record's current state is the source state; the state it changes to is the destination state.

Each state-based record type has a State Transition Matrix that lists all of the states available to the record type and the actions that move the record from one state to another. You can use the State Transition Matrix to create, modify, and delete states.

Double-click to open the State Transition Matrix

For example, the Postpone action moves the record from the Submitted state to the Postponed state

To/From	Submitted	Assigned	Opened	Resolved	Closed
Submitted					
Assigned	Assign				
Opened		Open		Reescalate	Reopen
Resolved			Resolve		
Closed	Close	Close		Validate	
Duplicate	Duplicate		Duplicate	Duplicate	Duplicate
Postponed	Postpone	Postpone	Postpone		

In the State Transition Matrix, source states are listed in columns (From), and destination states are listed in rows (To). The action necessary to move from a particular source state to a destination state is listed at the intersecting cell. If there is no action listed in the cell, a state transition is not allowed.

**Note:** You cannot have two different actions that go between the same source state and destination state.

## Adding a new state

To add a new state, open the State Transition Matrix. Select **Edit > Add State** and type a name for the new state. ClearQuest automatically adds the new state to the row and column headers in the State Transition Matrix.

Once you create a new state, you must also create a state transition that defines that state's place in the state model. See "Creating a state transition" on page 87.

**Note:** If you define a state in your schema, you must use that state. It is a validation error to define a state that cannot be reached by any actions.

If your schema contains packages that use state types, when you add a new state you must map the new state to a state type in your schema. See "Mapping state types" below.

**More information?** Look up *states, adding to record type* in the ClearQuest Designer Help index.

## Mapping state types

Some schema packages, such as the UnifiedChangeManagement (UCM) package and the Resolution package, modify schemas by adding hooks. These packages use state types to identify the states in which to execute hooks.

To ensure that a package with state types works properly with your schema, you need to map each state in each record type to an appropriate package state type. You can map more than one state to a state type, but each state type requires at least one state.

If you add a new state to a schema that uses state types, you must map that state to the appropriate package state type.

To map state types:

- 1 Open the State Transition Matrix for the desired record type.

- 2 Right-click a state and select Properties to display the properties dialog for the state.
- 3 In the State Types tab, select a package that requires state type mapping, then select a state type.

For example, to map the Closed state in an existing schema to the Complete state type in the UnifiedChangeManagement package:

In the Properties dialog for the Closed state . . .

. . . select the **Unified ChangeManagement** package to display the states types that must be mapped . . .



. . . then select the **Complete** state type to map it to the Closed state

- 4 Close the dialog to save the state mapping.
- 5 Repeat this process for each new state in the record type.

**Note:** If you are using the UCM schema or package, you must also assign default actions for your states; see “Setting default actions for the UCM package” on page 4.

### Changing the name of a state

You can modify the name of a state at any time. When you change the name of a state, ClearQuest automatically updates state-name information in any actions that refer to that state.

To rename a state, open the State Transition Matrix. Select the row of the state that you want to rename, and select **Edit > Rename State** to open the Rename State dialog.

**Note:** If a hook refers to the name of a state explicitly, you must manually change the name of the state in the hook code.

## Working with actions and action types

After defining your states, you are ready to define the actions you need to submit new records to the database, to modify or delete records, and to move records from state to state.

ClearQuest supports the following types of actions:

Action type	Description/Comments
Base	<p>A Base action is a secondary action that runs as a side-effect of every other action. Base actions allow you to write an action hook only once, but use it with multiple actions. Each time an action fires, the Base action checks to see if the hook criteria is met; if it is, the base action completes its process.</p> <p>For example, you can add a Notification action hook to a Base action to have the Base action automatically send e-mail notification when a Close action (a Change_state action type that moves the record to the Closed state) occurs.</p> <p>Base actions do not appear in the list of actions in the ClearQuest client.</p>
Change_state	<p>Change_state actions are available only for state-based record types. A Change_state action moves a record from a source state to a destination state. A Change_state action can reference many source states, but only one destination state.</p> <p>Change_state actions appear in the list of actions in the ClearQuest client only if the current record is one of the source states.</p>
Delete	<p>Allows users to delete a record from the database. Delete actions appear in the list of actions in the ClearQuest client.</p>
Duplicate	<p>Duplicate actions are available only for state-based record types. A Duplicate action links the record to another record that contains similar information.</p> <p>Duplicate actions appear in the list of actions in the ClearQuest client only if the current record is one of the source states.</p>

Action type	Description/Comments
Import	<p>Allows ClearQuest to import records from another source. During Import, ClearQuest validates the contents of imported records. However, ClearQuest does not perform field-level validation during Import. In addition, when a set of state-based records is imported, ClearQuest assigns them to a state specified in the data files without verifying whether they could have legally transitioned to that state.</p> <p>Import actions do not appear in the list of actions in the ClearQuest client.</p>
Modify	<p>Allows users to modify field values in a record without moving the record between states. Modify actions appear in the list of actions in the ClearQuest client.</p>
Record_script_alias	<p>Allows you to associate an action with a record script. Record_script_alias actions appear in the list of actions in the ClearQuest client.</p>
Submit	<p>Enters a new record into the ClearQuest user database. For state-based records, Submit assigns a destination state, but does not require a source. Each record type can have only one action whose type is Submit.</p>
Unduplicate	<p>Removes the link between duplicate records. Unduplicate actions are only available for state-based record types.</p>

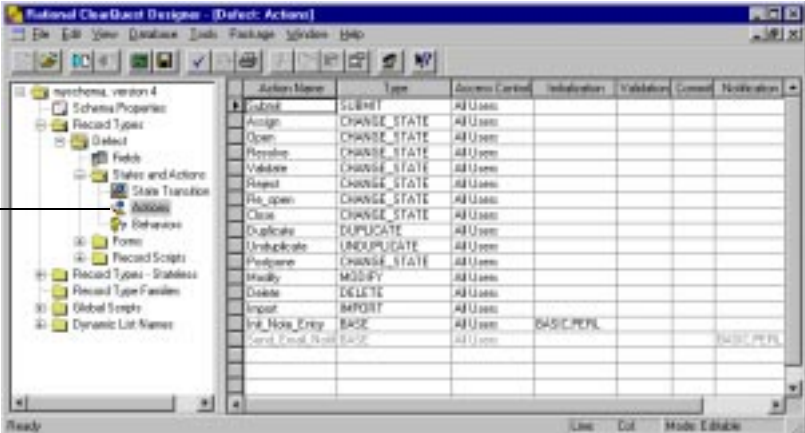
## Adding and modifying actions

Each record type has an Actions grid that defines the actions available for records of that type. You can use the Actions grid to add, modify, and delete actions, and to create state transitions.

To display the Actions grid, in the Workspace expand Record Types > Defect > States and Actions and double-click Actions.

Double-click to open the Actions grid

Right-click any action and select **Properties** to view and modify the action's properties



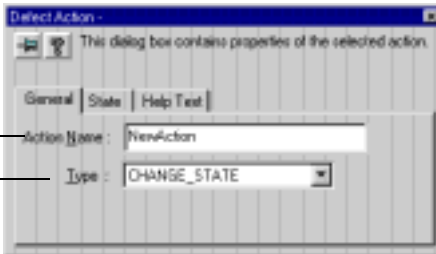
Action Name	Type	Access Control	Initialization	Validation	Control	Notification
Assign	CHANGE_STATE	All Users				
Open	CHANGE_STATE	All Users				
Resolve	CHANGE_STATE	All Users				
Validate	CHANGE_STATE	All Users				
Reassign	CHANGE_STATE	All Users				
Re-open	CHANGE_STATE	All Users				
Close	CHANGE_STATE	All Users				
Duplicate	DUPLICATE	All Users				
Unduplicate	UNDUPLICATE	All Users				
Prepare	CHANGE_STATE	All Users				
Ready	MODIFY	All Users				
Delete	DELETE	All Users				
Import	IMPORT	All Users				
Init_History_Entry	BASE	All Users	BASIC.PEPL			
Send_Email_Notif	EMAIL	All Users				BASIC.PEPL

## Adding a new action

To add a new action, open the Actions grid and select **Edit > Add Action**.

In the General tab, type a name

Select an action type



This dialog box contains properties of the selected action.

General | State | Help Text

Action Name: ViewAction

Type: CHANGE\_STATE

You must select a type for the action. If you plan to use this action to initiate a state transition, select **CHANGE\_STATE** as the type.

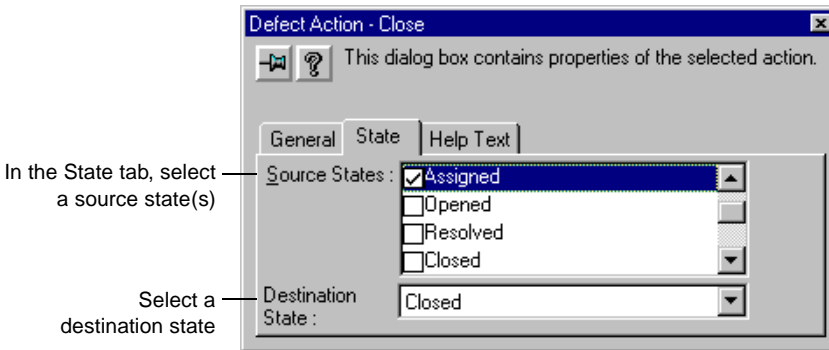
## Creating a state transition

To create a state transition, you define an action of the type `CHANGE_STATE` and then select the source state(s) and a destination state for that action:

- 1 Open the Actions grid and create a new action, selecting `CHANGE_STATE` as its type.

Or, right-click an existing action of the type `CHANGE_STATE` and select **Action Properties**.

- 2 In the State tab of the Action Properties dialog, select one or more source states and a destination state for the action.



Each `CHANGE_STATE` action must have at least one source state and a destination state. If none are defined, ClearQuest reports an error during validation.

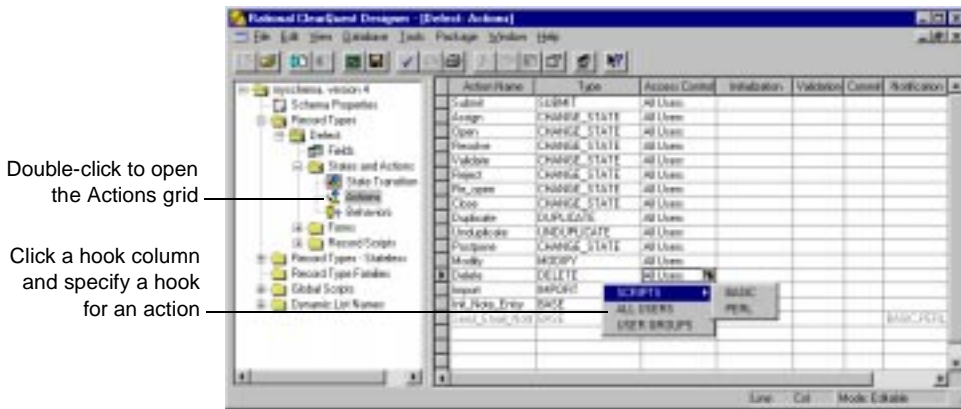
- 3 After creating the state transition, open the State Transition Matrix to verify that ClearQuest applied the transitions to the states. See “Using the State Transition Matrix” on page 81.

## Customizing actions by adding hooks

You can add action hooks that implement tasks at key points in a record's life. For example, by default ClearQuest grants all users access to every action. You can limit the access to an action by using an access-control hook.

ClearQuest provides several action hooks: Access Control, Initialization, Validation, Commit, and Notification.

To define an action hook, use the Actions grid.



You can customize ClearQuest action hooks by including scripts that use the ClearQuest API. When you finish editing a scripted hook, select **Hooks > Compile** to check the syntax of your code.

**More information?** For a description of action hooks and information about how they work with field hooks, see Chapter 7, “Using hooks to customize your workflow.” To learn how to create an access control action hook, see “Controlling user access to actions” on page 109. Also see “Selecting a scripting language” on page 51.



## Using default actions

You can define default actions for states. A default action for a state appears in bold in the ClearQuest client Actions menu.

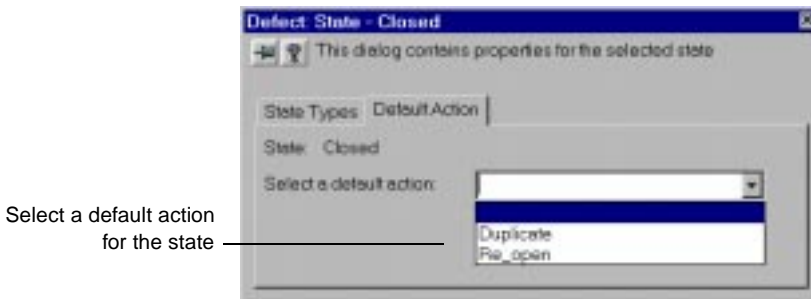
Associating a state with a default action:

- helps guide users through your state model
- is required for certain schemas and packages, such as the UCM schema and package

**Note:** If you use the UCM schema or package, the default actions of your states must provide a valid path through the state type model. For more information, see “Setting default actions for the UCM package” on page 4.

To select a default action for a state:

- 1 Open the State Transition Matrix for the record type.
- 2 Right-click the state you want to edit, and select **Properties** to open the Properties dialog for that state.
- 3 In the Default Action tab of the Properties dialog, select a default action for the state.



**More information?** Look up *actions, default* in the ClearQuest Designer Help index.

## Working with record forms

Each record type can have two forms associated with it: a record form and a submit form. A record type must have a record form, but a submit form is not required.

A submit form allows users to use a different form to submit records of that type. When a user first submits a new record, ClearQuest displays the submit form. Later, when the user views and works with the record, ClearQuest uses the record form. If the record type does not have a submit form, ClearQuest uses the record form both for submitting records and for working with them.

**More information?** Select [Working with record forms > Form Control Reference](#) in the ClearQuest Designer Help.

### Creating a new form

To create a new form:

- 1 In the Workspace, expand the record type you want to modify.
- 2 Right-click the Forms folder and select **Add** from the shortcut menu. ClearQuest displays a new empty form and highlights the name of the form in the Workspace so that you can change it.
- 3 In the Workspace, type a name for the new form.
- 4 Right-click the form name and select **Submit Form** or **Record Form**.

After creating a form, you use controls to add fields to the form. If the record that owns the form contains too many fields to display on one dialog tab, you can add additional dialog tabs as necessary. Every form must have at least one dialog tab.

ClearQuest automatically adds three controls to every form: an Apply button, a Revert button, and an Actions button. You cannot delete these buttons. ClearQuest uses them to initiate and manage actions.

## Working with form controls

You use controls to display fields on the form. ClearQuest provides controls for text boxes, list boxes, check boxes, option buttons, and so on. For example, you can associate a field containing a string with a text-box control. Not all controls work with all field types. For example, a list-view control or a parent/child control must be used with a reference-list field.

In addition to displaying the contents of fields, you can use some controls to perform special tasks. Controls such as push buttons and list boxes can be associated with record scripts. For example, in the TestStudio schema, a push button is associated with the Build\_Properties record script, which allows users to view the properties of the build they've selected.

ClearQuest supports the following form controls:

Form control	Description
ActiveX	Incorporates any registered ActiveX control into a form.
Attachment	Displays a list of attached files and includes a set of controls that allow users to add, remove, or view attached files.
Check Box	A two-value control you can use for Boolean values or any field that has only two values. To specify the two values, right-click the control on the form and select <b>Properties</b> .
Combo Box	Combines an editable text field with a list box.
Drop-down Combo Box	Combines an editable text field with a drop-down list box.
Drop-down List Box	Displays a list of possible values for a particular field.
Duplicate Box	Displays the ID of the record of which this record is a duplicate.
Duplicate Dependent	Displays the IDs of any records that are duplicates of this record.
Group Box	A control used to visually group one or more other controls.
History	Displays the actions that have been applied to a record.
List Box	Displays a list of possible values for a particular field.

<b>Form control</b>	<b>Description</b>
List View	Allows you to display the records associated with a field of the REFERENCE_LIST type.
Option Button	Option button controls are used in groups to represent a set of mutually exclusive choices.
Parent/child	Allows you to set up a form to link associated records. Used with REFERENCE_LIST field type.
Picture	Lets you display a static image on your form.
Push Button	Initiates specific tasks related to the record. You can associate push buttons with record hooks or with list views.
Static Text	Displays an uneditable text string.
Text Box	Displays a field's value as an editable text string.

**More information?** For detailed descriptions and instructions on how to use each control, select **Working with record forms > Form Control Reference** in the ClearQuest Designer Help.

## Adding a field to a form

Before you can add a field to a form, you must first create the field in the Fields grid. For instructions on how to create a new field, see “Adding a new field” on page 71.

You can add field controls using the Control Palette, Fields List, or the Form Controls menu. Each technique has its benefits. The Control Palette gives you a visual cue as to the type of control you are adding. The Fields List automatically creates a control that is appropriate for the type of field you have chosen. The Form Controls menu allows you to use keyboard mnemonics to select controls quickly.

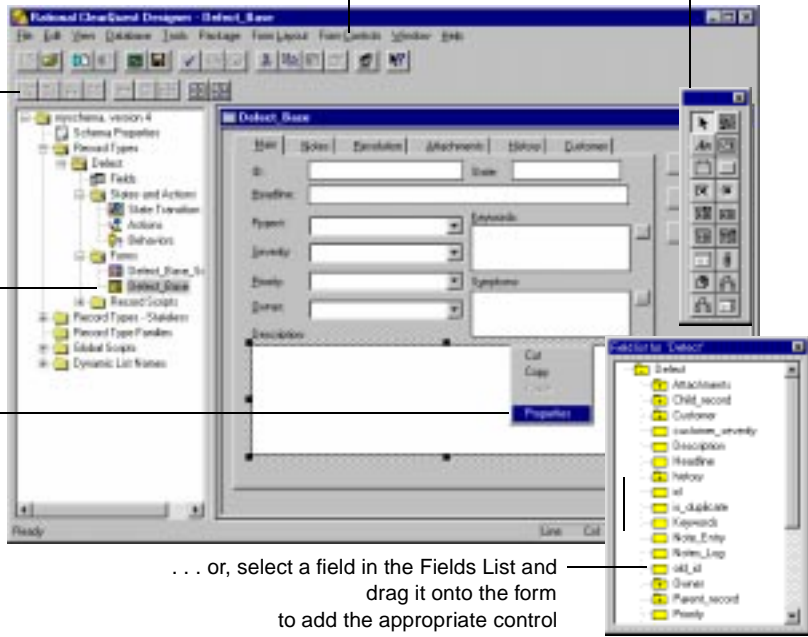
To add a field to a form, expand **Record Types > Record\_Type\_Name > Forms** in the Workspace and double-click the form to open it for editing.

To add controls to a form, use the Form Controls menu or the Controls Palette . . .

Use the Form toolbar to arrange the control on the form

Double-click a form to open it for editing

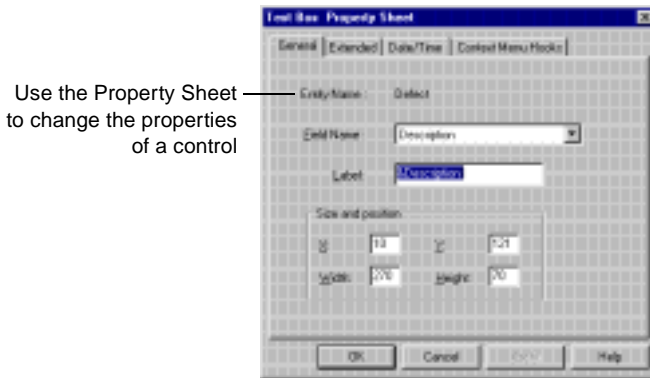
After adding the control, right-click the control and select **Properties** to edit the control's properties



## Editing field control properties

After adding the field control to the form, right-click the control and select **Properties** from the shortcut menu.

**Note:** If you did not create the control by dragging the field from the Fields List, you must edit the control's properties to associate the field with the control.



You can select or modify such items as the field's name and label, data formats, context menu hooks, and whether or not the field has horizontal or vertical scrolling. The options you are allowed to select depend on the type of field.

## Editing forms

For step-by-step instructions on how to edit a form, select **Working with record forms > Editing forms** in the ClearQuest Designer Help.

You can find out how to:

- Change the size of a form
- Add and delete dialog tabs
- Add tab-level access
- Add and delete controls

## Reusing forms

After creating a form, you can save time by reusing a form in other schemas. You can use it as is or modify it.

Right-click a form in the Forms folder and select **Export Form** from the shortcut menu. You can then import the form into other schemas.

You must import the form into a record type with a name identical to the record type that exported the form. Right-click a form in the Forms folder and select **Import Form**. You cannot export a form and then import it into a different record type within the same schema.

When importing a form, ClearQuest maps the fields associated with the form to the fields of the selected record type. ClearQuest maps fields based on their name and does not check to see if the fields have matching types. If a field in the form does not correspond to a field in the selected record type, you must reassign the corresponding control to a valid field or delete the control.

## Deleting forms

If you no longer need a form, you can delete it. However, remember that every record type must have at least one form associated with it. If a record type has only one form left, that form must be a record form; ClearQuest will not let you delete the last form of a record type.

To delete a form:

- 1 In the Workspace, expand **Record Types** or **Record Types - Stateless** and expand the record type you want to modify.
- 2 Expand the **Forms** folder, right-click the form, and select **Delete** from the shortcut menu.

## **Creating forms for multiple platforms**

The forms you create in ClearQuest Designer appear differently when viewed by the Windows and ClearQuest Web clients. For example, in Windows you click a tab to display it, while in ClearQuest Web you can either click a link or scroll to display various tabs. However, the same information is available no matter which client you use.

Also, some hooks work differently in ClearQuest Web client. See “Using hooks in ClearQuest Web” on page 146 for information on using hooks on the Web client.

**More information?** See Chapter 8, “Administering ClearQuest Web.”



# 6

## Administering users

Windows user profiles, including a user ID and password, cannot be used within ClearQuest. You must use ClearQuest Designer to set up users. ClearQuest stores information about users and user groups in the user database and in the schema repository.

This chapter describes how to set up and administer ClearQuest users and user groups. The topics covered include:

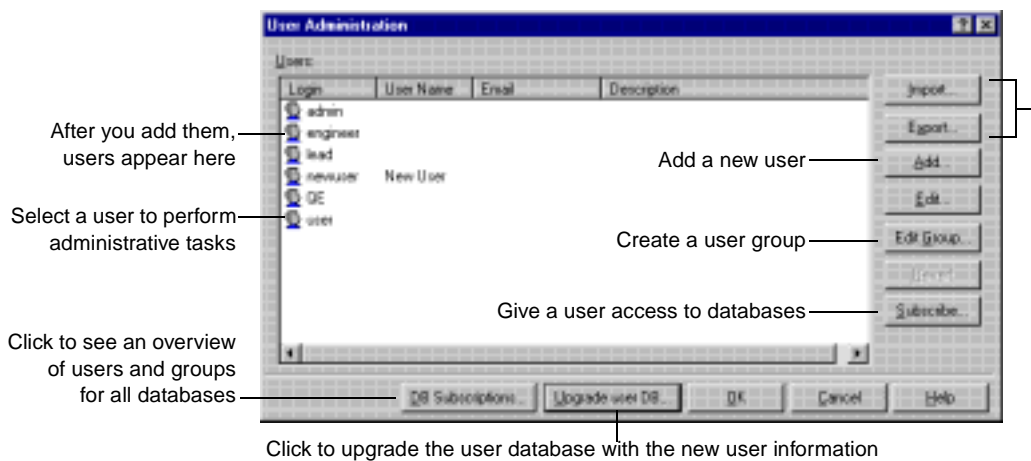
- Overview of user administration
- ClearQuest user privileges
- Working with users
- Working with user groups
- Controlling user access to actions
- Exporting and importing users and groups

**Note:** To administer users and user groups, you must have User Administrator or Super User privileges. ClearQuest Designer includes a default User Name (*admin*) that you can use to get started. You do not need to type a password. The *admin* user account is set up with the Super User privileges you need to perform all ClearQuest administrator functions. For more information, see “ClearQuest user privileges” on page 99.

## Overview of user administration

To administer ClearQuest users, you use the User Administration dialog. In ClearQuest Designer, select **Tools > User Administration**.

Export user data and import it into another schema repository



Use the User Administration dialog to:

- Create new users and user groups: First create users, then create a group and add the users to the group.
- Assign privileges to users and user groups that define the tasks they can perform within ClearQuest and ClearQuest Designer. See “ClearQuest user privileges” on page 99.
- Control the data that users and groups can access by giving them access to specific databases. See “Subscribing users to databases” on page 102.
- Export user data and import it into another schema repository. See “Exporting and importing users and groups” on page 110.

**Note:** Whenever you add or modify a user or user group, you must upgrade your user database(s) with the new information. See “Upgrading the user database” on page 101.

You can also restrict user and group access to specific ClearQuest actions by adding an access-control hook to the action. See “Controlling user access to actions” on page 109.

## ClearQuest user privileges

ClearQuest supports the following user and user group privileges:

Privilege	Allows user /group to
Active User	(Default) Log into ClearQuest and submit new records; modify existing records; and create, modify, and save personal queries, charts, and reports.  Log into ClearQuest Designer to view schemas and to view user administration information. Cannot edit schemas or change user information.
Schema Designer	Perform all Active User tasks and:  Create, modify, and save public queries, charts, and reports in ClearQuest.  Use ClearQuest Designer to create and modify schemas.
User Administrator	Perform all Active User tasks and:  Use ClearQuest Designer to create and administer users and user groups.
Super User	Perform all Active User, Schema Designer, User Administrator tasks and:  Use ClearQuest Designer to create and delete databases. Edit ClearQuest Web settings.

**More information?** Look up *access, privileges* in the ClearQuest Designer Help index.

## Working with users

Use the User Administration dialog to set up and modify users.

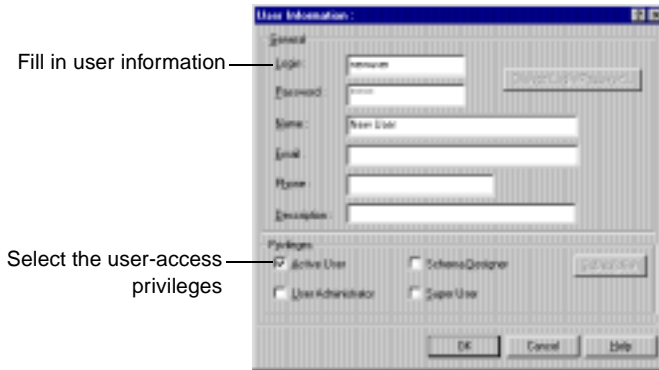
### Adding new users

To add a new user:

- 1 In ClearQuest Designer, select **Tools > User Administration** to open the User Administration dialog.



- 2 In the User Information dialog:



When you're done, click **OK**.

- 3 In the User Administration dialog, click **Upgrade user DB** to add this new information to your user database(s). See the next section "Upgrading the user database."

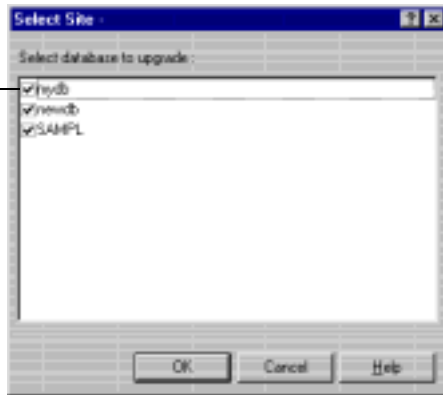
**More information?** Look up *users, editing an account profile* in the ClearQuest Designer Help index.

## Upgrading the user database

Whenever you add or modify users or user groups, you must upgrade your user database(s) with the new information.

In the User Administration dialog, select **Upgrade user DB**.

Select the databases to upgrade with the new user information

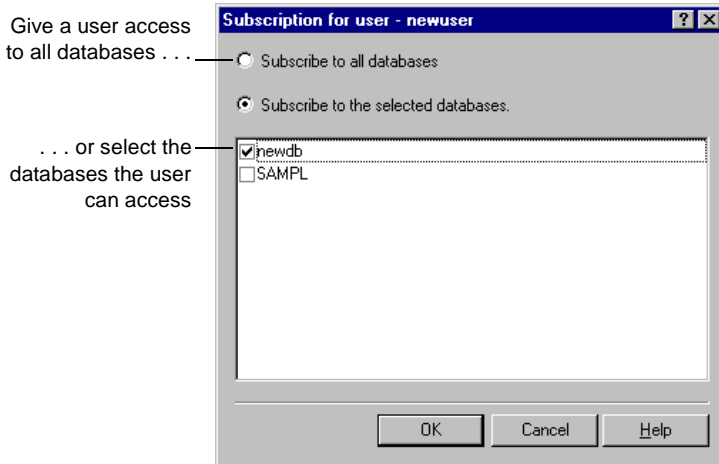


## Subscribing users to databases

Subscribing a user to a database means giving that user access to the database. By default, ClearQuest subscribes users to all databases. You can restrict the access of a user or a group to the database(s) you specify.

To subscribe a user to a database:

- 1 In ClearQuest Designer, select **Tools > User Administration** to open the User Administration dialog.
- 2 In the User Administration dialog, select the user and click **Subscribe** to open the Subscription for User dialog.



When you're done, click **OK**.

- 3 In the User Administration dialog, click **Upgrade user DB** to add this information to the user database. See "Upgrading the user database" on page 101.

**More information?** Look up *users, subscribing to databases* in the ClearQuest Designer Help index.

## Modifying user profiles

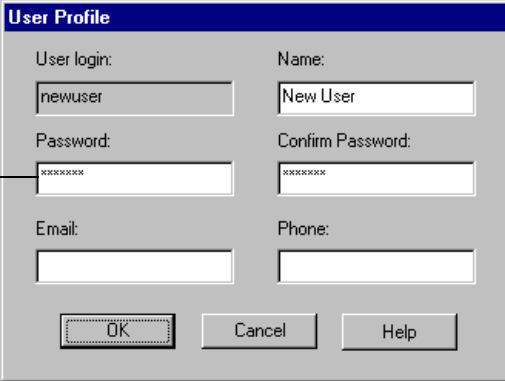
As an administrator with User Administrator or Super User privileges, you can edit a user profile, including the privilege(s) assigned to the user.

To edit a user profile, select **Tools > User Administration** to open the User Administration dialog, then click **Edit**.

Active users can also change some of their own user information, including their name, e-mail address, password, and telephone number, from either ClearQuest Designer or the ClearQuest client. However, active users cannot change their own user login or privileges.

To modify a user profile from the ClearQuest client, select **View > Change user profile**.

Change the user information



The image shows a dialog box titled "User Profile" with a blue header bar. It contains several text input fields arranged in two columns. The left column has fields for "User login:" (containing "newuser"), "Password:" (containing "xxxxxxx"), and "Email:". The right column has fields for "Name:" (containing "New User"), "Confirm Password:" (containing "xxxxxxx"), and "Phone:". At the bottom of the dialog are three buttons: "OK", "Cancel", and "Help". A line from the text "Change the user information" points to the Password field.

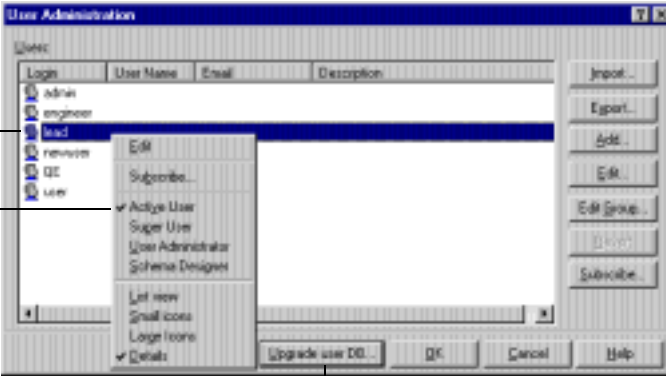
User login:	Name:	
newuser	New User	
Password:	Confirm Password:	
xxxxxxx	xxxxxxx	
Email:	Phone:	
OK	Cancel	Help

Changes made to the user profile apply to all the databases the user is subscribed to.

## Making users inactive

You can make users inactive, which means they cannot log into ClearQuest or have defect records assigned to them. Inactive users do not appear in any choice lists of users in ClearQuest.

To make a user inactive, select **Tools > User Administration** to open the User Administration dialog:



The screenshot shows the 'User Administration' dialog box with a table of users. The user 'mad' is selected, and a context menu is open over it. The 'Active User' option is checked. The 'Upgrade user DB' button is highlighted at the bottom.

Login	User Name	Email	Description
admin			
engineer			
<del>mad</del>			
reviewer			
QC			
user			

1. Right-click a user
2. Unselect all user privileges, including **Active User**
3. Click **Upgrade user DB** to add the changes to the database

An inactive user appears with an X in the list of users.



The screenshot shows the 'User Administration' dialog box. The user 'mad' is now marked with a small 'X' in the 'Login' column. The context menu is still open over 'mad', and the 'Active User' option is selected.

Login	User Name	Email	Description
admin			
engineer			
<del>X mad</del>			
reviewer			
QC			
user			

An inactive user appears crossed out

To make the user active again, right click and select **Active User**

For historical information and data integrity, an inactive user's name is retained in the database to which that user was subscribed.



## Working with user groups

Combining individual users into a group can make your administration tasks easier. For example, you can:

- Create groups such as “Software Engineers” or “Managers,” assign different privileges to each group, and subscribe them to different databases.
- Base an e-mail rule on a group (for example, notify the “Quality Assurance” group whenever a user submits a new defect).
- Write hook scripts or external applications with conditional logic for groups (for example, only members of the “Managers” group can reopen a defect marked “Resolved”).

## Creating user groups

To create a user group:

Select Tools > User Administration to open the User Administration dialog, then click Edit Group to open the User Group dialog.

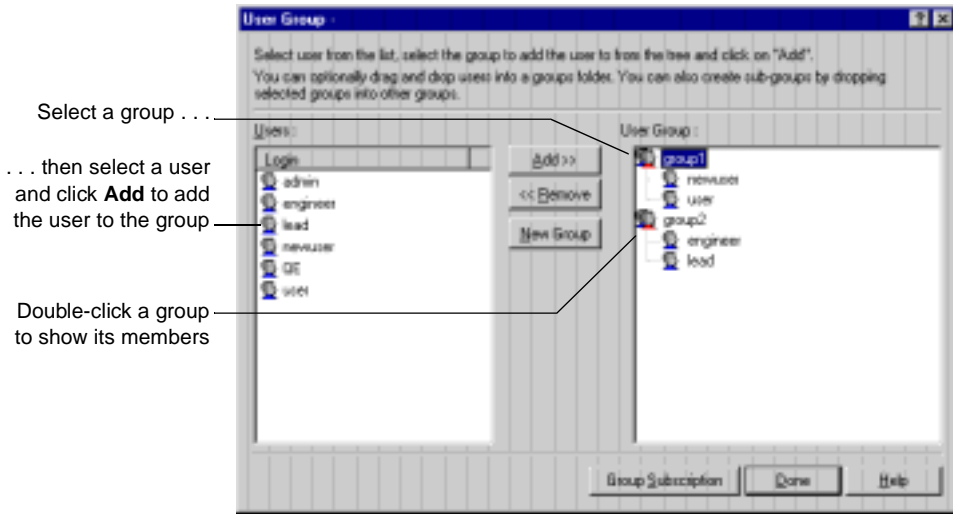


You can use the User Group dialog to create user groups, add users to groups and remove them, and subscribe groups to databases.

## Adding users to a group

To add users to a group:

Select **Tools > User Administration** to open the User Administration dialog, then click **Edit Group** to open the User Group dialog:



To remove users, double-click the group to display its members, then select a user and click **Remove**.

**More information?** Look up *user groups, creating* in the ClearQuest Designer Help index.

## Subscribing user groups to databases

By default, ClearQuest subscribes new groups to all databases. You can take advantage of user groups to subscribe multiple users to one or more databases.

To subscribe a group to databases:

- 1 Select Tools > User Administration to open the User Administration dialog, then click Edit Group to open the User Group dialog:

Select a group and click **Group Subscription** to select the databases the group can access



When you're finished, click Done.

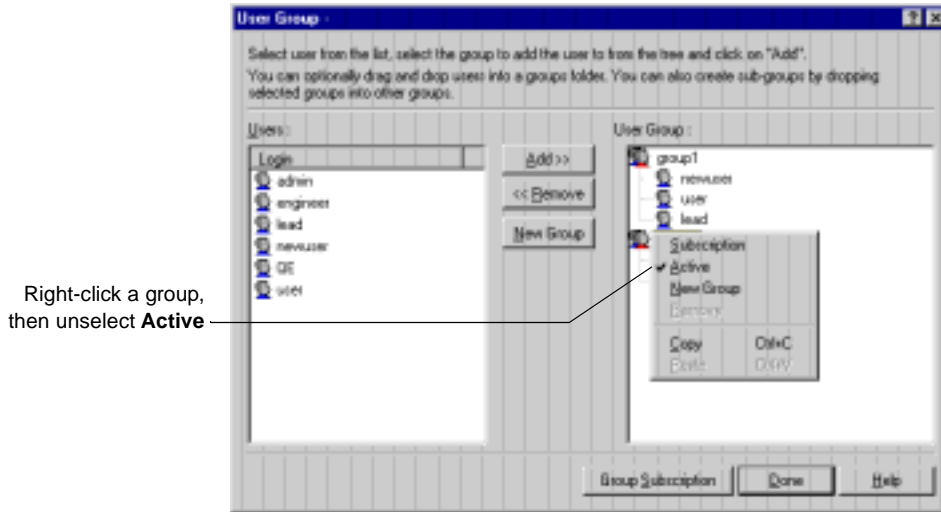
- 2 In the User Administration dialog select Upgrade user DB to add the changes to the user database. See "Upgrading the user database" on page 101.

**More information?** Look up *user groups, subscribing to a database* in the ClearQuest Designer Help index.

## Making user groups inactive

To make a user group inactive:

- 1 Select **Tools > User Administration** to open the User Administration dialog, then click **Edit Group** to open the User Group dialog:



When you're finished, click **Done**.

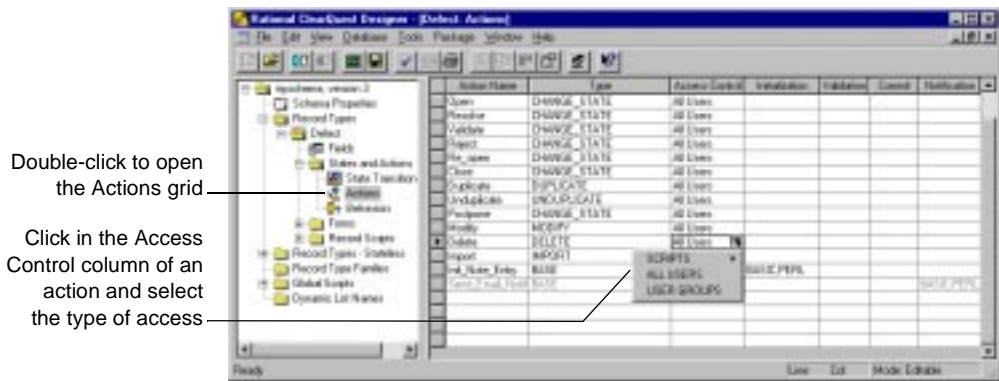
- 2 In the User Administration dialog, select **Upgrade user DB** to add the changes to the user database. See “Upgrading the user database” on page 101.

## Controlling user access to actions

You can focus the responsibility of a user or user group by limiting the actions they can perform in ClearQuest. For example, you might want to limit the Assign action to your Managers group and limit the Verify action to the Quality Assurance group.

To restrict the access to an action, add an Access Control hook to the action:

In the Workspace, open the Actions grid.



- Select **Scripts** to write your own access-control hook that restricts access to this action based on the criteria you define. For example, you can write a hook that allows access to an action to users who have Super User privileges. For more information, see Chapter 7, “Using hooks to customize your workflow.”
- Select **All Users** (the default) to allow any ClearQuest user to access the action.
- Select **User Groups** and select the groups you want to have access to this action.

**More information?** Look up *actions, restricting user access to* in the ClearQuest Designer Help index.

## **Exporting and importing users and groups**

You can use the **Import** and **Export** buttons in the **User Administration** dialog to transfer a list of users and user groups from one schema repository to another. When you export the list of users and groups, ClearQuest creates a text file you can import into another schema repository. The text file contains the profiles of each user and group.

# 7

## Using hooks to customize your workflow

ClearQuest includes many predefined hooks that you can use to implement your workflow. ClearQuest also includes an Application Programming Interface (API) that you can use to customize predefined hooks, to write your own hooks, and to write external applications for performing tasks on a ClearQuest database.

This chapter describes how to work with hooks and the ClearQuest API. The topics covered include:

- Overview of hooks
- Using field hooks
- Using action hooks
- Execution order of field and action hooks
- Using record scripts
- Using global scripts
- Writing external applications
- Using the ClearQuest API
- Common API calls
- Finding specific hook script text
- Examples of common hooks

**More information?** For more information on the ClearQuest API, select **Rational ClearQuest API Reference** from the Start menu. The *Rational ClearQuest API Reference* is an online reference guide for all ClearQuest API calls. To learn about using hooks in the web environment, see Chapter 8, “Administering ClearQuest Web.”

## Overview of hooks

Hooks are entry points, like triggers, for pieces of code that ClearQuest executes at specified times to customize how users work with ClearQuest. ClearQuest supports four types of hooks:

- **Field hooks**

Field hooks provide a way of checking a field's value at runtime and of adjusting other fields as necessary. For example, you can validate the contents of a field or assign it a default value.

- **Action hooks**

Action hooks provide a way of implementing tasks at key points in the life cycle of a record.

In general, action hooks are associated with events that affect the whole record as opposed to field hooks, which are associated with events that affect a particular field. For example, you can validate the entire record and send notifications when the action is complete.

- **Record scripts**

Record scripts provide a way to perform specific tasks at runtime. They are specific to a record type and are usually associated with form controls.

- **Global scripts**

Global scripts provide a way to define libraries of routines that can be shared by all of the record types in your schema. For example, you can write a subroutine, such as an e-mail notification, that can be called from any hook in any record type.

**Note:** You can write hooks in VBScript (for Windows) and Perl (for Windows and UNIX). By default, ClearQuest runs hooks in VBScript on Windows. To learn how to have ClearQuest run hooks in Perl for Windows, see “Selecting a scripting language” on page 51. When you finish editing a hook, select **Hooks > Compile** to check the syntax of your code.



## Planning hooks for your workflow

A carefully planned set of hooks can enhance and support your organization's workflow. Consider the following when you plan your hooks:

- Be aware that hooks run with Super User privileges and are not subject to the usual access control or field behavior restrictions. This means you can use hooks to set and reset values in fields that are normally read-only. (You cannot reset ClearQuest system fields that are read-only, such as the History field.) Required fields remain required.
- You can take advantage of outside code to extend hook capabilities. VBScript has access to COM objects, and Perl can gain access to COM objects through a third-party package. In addition, Perl can take advantage of many third-party packages. For example, a hook could interact with the user through dialog boxes, or read from and write to external files. You are responsible for ensuring that the proper third-party objects are installed on the client machine.
- Although it is possible to include SQL code in your scripts, for best results you should use the ClearQuest API to run queries and to retrieve data within script code. ClearQuest allows you to rename record types and fields, and this will be problematic in any SQL code that you include.
- If you plan to run your hooks on ClearQuest Web, write them in VBScript and be sure to test them in the web environment. See “Using hooks in ClearQuest Web” on page 146.
- Test and debug your hook code so that you do not write incorrect values into your database or cause the program to hang while processing an infinite loop or waiting for nonexistent input. To view debugging information (the output of the OutputDebugString method), you can use dbwin32.exe, which ships with ClearQuest.

## Using field hooks

You use a field hook for an event that affects a particular field within the record. A field hook can set an initial value, respond to events when a field value changes, enforce access permissions so only the user groups you specify can change field values, and validate the values your users provide.

The scope of a field hook is the current field within the current record. ClearQuest provides the following predefined field hooks:

Field hook	Use
Choice List	Returns a set of legal values. Use this hook with fields that are displayed using a list-type control, such as a list box or combo box. You can also provide values without scripting by using a constant or a dynamic list. See “Working with choice lists” on page 117.
Default Value	Sets the initial value of the field. This hook is called at the beginning of a Submit action. You can also assign a constant value as the default value.
Permission	Returns one of the BehaviorType constants indicating the user's access to the field. Use this hook to force workflow and/or security. (See the online <i>ClearQuest API Reference</i> for enumerated constants.)  If you add a Permission hook to a field, you must modify the Behaviors grid so that at least one of the field's behaviors is set to USE_HOOK. Failure to do this will result in a validation error.
Validation	Validates the contents of the field. This hook is called when the value changes, to provide the user with immediate feedback regarding the validity of the field's contents before committing the record to the database.
Value Changed	Responds to changes in the value of a field. Use this hook to trigger updates for other fields (for example, dependent lists). After executing this hook, ClearQuest validates any field the script has modified by calling the field's Validation hook (if any).

**More information?** For instructions on how to specify field hooks, see “Customizing fields by adding hooks” on page 79. To learn about the sequence in which hooks run, see “Execution order of field and action hooks” on page 122.



of setting a parent value based on child values, see “Action hook for setting the value of a parent record” on page 138.)

- 4 When you finish editing your script, select **Hooks > Compile** to check the syntax of your script code.

### **Using dependent fields on ClearQuest Web**

If you want dependent fields to be enabled in ClearQuest Web, you must use one of the following controls when you add the fields to the record form:

- Drop-down list box
- Combo box
- Drop-down combo box

When you add the control to the record form, you must also edit the control’s properties to specify the field on which the dependency is based.

**More information?** See “Enabling dependent fields for ClearQuest Web” on page 146.



## Creating a dynamic list hook

To create a Dynamic List hook:

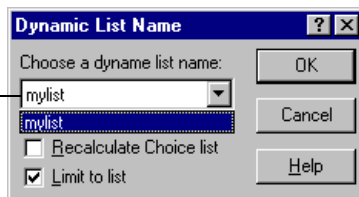
- 1 In the Workspace, create a dynamic-list name.

Right-click **Dynamic List Names** and type a name for the dynamic list



- 2 Open the Fields grid. Click the Choice List cell for the desired field and select Dynamic List from the drop-down list.
- 3 When prompted, select the dynamic list name you just created.

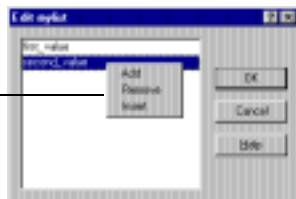
Select the dynamic list name to use



- 4 Next, you must define the values for the dynamic list. You must have Schema Designer or Super User privileges to edit the list. (See “ClearQuest user privileges” on page 99.)

In ClearQuest client, select **Edit > Named Lists** and select the list you want to edit.

Right-click and use the shortcut menu to add, remove, and insert field-list values



## Defining choice list behavior

When you define a Constant Value, Constant List, or Dynamic List hook, you can select the following behaviors:

- Select **Limit to List** to prevent users from entering values that are not in a choice list. Any value not within the list fails validation. If you do not choose the Limit to List option, users can enter a value not in the choice list.
- Select **Recalculate Choice List** to have ClearQuest reinitialize the list during runtime interaction if it depends on changes to another value. For a scripted choice list, this refresh operation might incur some performance overhead. If you do not select Recalculate Choice List, ClearQuest only refreshes the values at the beginning of each action.

## Using action hooks

Action hooks can control who has permission to change record values and validate user entries before ClearQuest commits them to the database. Action hooks can also validate the entire record and send e-mail notification when the action is complete.

The scope of an action hook is the current record. ClearQuest provides the following predefined action hooks. They are listed in the order in which they execute.

Action hook	Use	Executed
Access Control	Returns a Boolean indicating whether the specified user can initiate the specified action on a record. This hook is called before the user performs the action.  You can write an access-control hook as a VBScript or Perl subroutine.  See "Controlling user access to actions" on page 109.	When the action is about to start.
Initialization	Sets initial field values (or any task you specify).  Allows complex initialization of a record. You can use this hook to set up field values before ClearQuest begins an action. This hook is called after the action has been initialized but before the contents of the record are displayed in a form.  You must write an initialization hook as a script subroutine.	When the action starts.



Action hook	Use	Executed
Validation	<p>Validates the field values you specify. If the user types invalid data, ClearQuest prompts the user for valid data.</p> <p>You can use this hook to check conditions that are difficult to verify inside the individual field validation hooks. For example, you can use this hook to verify information across a group of fields. ClearQuest executes this hook prior to committing any changes to the database.</p> <p>Validation hooks must use a script.</p>	When the user commits the action.
Commit	<p>Links an action on multiple records into a single transaction (for example, resolving all the duplicates of a change request when the original is resolved).</p> <p>Updates a set of external data sources so that they stay parallel with the database contents. This hook is called after changes are added to the database but before those changes are committed.</p> <p>You can write a commit hook as a VBScript or Perl subroutine.</p>	Just before ClearQuest commits the transaction to the database.
Notification	<p>Starts a post-commit action that notifies users when an action is performed. See Chapter 9, "Administering ClearQuest E-mail" .</p> <p>Notification hooks must use a script.</p>	After ClearQuest commits the transaction.

**More information?** For instructions on how to specify action hooks, see "Customizing actions by adding hooks" on page 88. To learn about the sequence in which hooks run, see "Execution order of field and action hooks" on page 122.

## Execution order of field and action hooks

ClearQuest executes field and action hooks at predetermined times and in a predetermined order. There are four hook execution points while a record is open:

- When an action begins
- When a field's value is set
- When the record is validated
- When the record is committed

### When an action begins

When a user initiates an action, ClearQuest executes hooks in the following order:

- 1 The **Action Access Control** hook stops the processing of the action if the user is not allowed to initiate the action. In this case, the remaining hooks do not execute.
- 2 The **Field Permission** hook determines whether each field is mandatory, optional, or read-only.
- 3 The **Action Initialization** hook performs initialization tasks such as setting up field values before ClearQuest begins the action.
- 4 The **Field Default Value** hook sets the value for each field (for Submit actions only).

### When a field's value is set

When a user edits a record, ClearQuest executes hooks in the following order:

- 1 The **Field Value Changed** hook runs for each field that changes.  
  
If the Limit to List option is set and the user enters an illegal value, ClearQuest marks the field as invalid. Only when the user enters a legal value does the next hook run.

**Note:** If a Field Value Changed hook calls SetFieldValue to change the value of another field, ClearQuest executes the other field's Field Value Changed hook immediately.

- 2** The **Field Validation** hook runs for each field.
- 3** The **Field Choice List** hook runs for each field that uses the Recalculate Choice List option.

If you use dependent fields with the Recalculate Choice List option, ClearQuest first runs the Field Validation hook and then the Field Choice List hook for each field, until all changed fields have been set and validated.

**Note:** In ClearQuest Web, field hooks run only when the user invokes the Submit action. See "Using hooks in ClearQuest Web" on page 146.

### **When the record is validated**

Before committing changes to the database, ClearQuest validates the record by executing hooks in the following order:

- 1** The **Field Validation** hook runs for each field in the record.
- 2** The **Action Validation** hook runs for the action that was performed.

## When the record is committed

The Commit hook executes after the database has been updated with changes to the current record, but before the update transaction has been *committed* to the database. This means that you cannot use a Commit hook to modify the current record; such modifications are not applied to the record.

Use a Commit hook only for actions against other records that you want to be part of the same database transaction as the main action. (For example, resolving a duplicate defect when the parent defect is resolved.)

After ClearQuest validates a record, it commits the record to the database by executing hooks in the following order:

- 1 The **Action Commit** hook runs.
- 2 ClearQuest commits the transaction to the database.
- 3 The **Action Notification** hook runs. For example, ClearQuest might send an e-mail message to various users based on the e-mail rules you set up.

## Using record scripts

You can write a script that performs custom behavior in the context of a record. A record-script subroutine is specific to one record type. For example, a record script could send data about the current record to another system.

There are three ways to invoke a record script:

- Associate a record script with a form control, either a button or a (right-click) shortcut menu.

For example, in the TeamTest schema, the Build\_Properties record script allows users to view the properties of the build they have selected. The script is associated with a button that, when clicked, runs the script. The Build\_Properties record script then retrieves the build information from the Rational Repository and displays the information in a dialog box.

- Use an action to invoke a record script. Create a Record\_script\_alias action and associate it with a record script. When the user selects the action, the record script is invoked immediately. To update the current record from within such a script, begin an action in the script using the ClearQuest API `EditEntity` method in the `session` object.
- Call a record script from within another script or hook by using the ClearQuest API `FireNamedHook` method in the `entity` object.

**More information?** Look up *record scripts* in the ClearQuest Designer Help index and refer to the online *ClearQuest API Reference*.

## Using global scripts

You can use global scripts to define common functions that you can call from any hook in your schema. Global scripts are like a library of subroutines; ClearQuest does not invoke them directly.

Global scripts are useful when you have multiple record types that call the same hook code. They enable you to centralize hook-code maintenance and avoid copying the hook code into multiple places. For example, the global script included in ClearQuest's Email package allows multiple actions to send notification by calling one global script.

**More information?** Look up *global scripts* in the ClearQuest Designer Help index.

## Writing external applications

You can write an external application in VBScript or Perl to perform tasks against a ClearQuest database. You can use an external application to create a query, execute a saved query, create new records, and perform actions on existing records. For example, an external application can:

- At midnight each day, execute a predefined set of queries and mail the results to your managers' group.
- Each Sunday, find all defects that have been open for more than a month and escalate their status.

An external application must begin by creating a session object and logging into a ClearQuest database. See "Working with sessions" on page 127.

**Note:** ClearQuest ships its own version of Perl (CQPerl.exe, CQPerl.dll). When creating external applications using Perl, make sure you use the CQPerlExt package. See "Using the ClearQuest API" in the online *ClearQuest API Reference*.

## Using the ClearQuest API

You can use the ClearQuest API to customize ClearQuest's predefined hooks, to write your own hooks, and to write external applications that perform tasks against ClearQuest databases.

**More information?** See the online *ClearQuest API Reference* for details on the objects, methods, properties, and constants you can use when you write hooks and/or external applications.

### Working with sessions

The Session object represents the current database-access session and is the starting point of all operations. If you are writing hooks, ClearQuest provides access to the current Session object through the GetSession method of the Entity object. Because hooks operate in the context of modifying a record (entity), you always have a corresponding Entity object from which to call GetSession.

If you are writing an external application to access ClearQuest databases, you must create a Session object and log into the database. To work with an entity, you must then call the API that returns the entity object.

**More information?** See “Working with sessions” in the online *ClearQuest API Reference*.

### Working with queries

You can run queries to retrieve data from a ClearQuest database based on a set of search criteria that you provide. To build a query:

- Build a query using the QueryDef object to specify the data you want.
- Create a ResultSet object to hold the data.
- Execute the query to retrieve the data in the result set.
- Access the data.

**More information?** To learn how to build a query using objects such as QueryDef and ResultSet, see “Working with Queries” in the online *ClearQuest API Reference*.

## **Working with records**

When one of your users enters a change request, ClearQuest stores the data in a logical record called an entity. You can create, edit, and view a record's data, as well as view data about the record's entity type. The BuildEntity method enables you to create a new record; the EditEntity method enables you to edit an existing record. The ClearQuest API provides methods, as well, for you to validate your changes and commit the updated record to the database.

**More information?** See “Working with Records” in the online *ClearQuest API Reference*.



## Common API calls

This section lists the basic building blocks from which you can create hooks. Each API call is shown first in VBScript and then in Perl. The syntax uses an `<object.><method>` format.

**Note:** In Perl, the current `Entity` object and `Session` object are predefined as `entity` and `session` (lowercase). For VBScript, the current `Entity` object is assumed and you do not need to explicitly identify it when calling its methods.

**More information?** See the online *ClearQuest API Reference*.

API Call (VBScript/Perl)	Function
<code>[entity.]GetSession</code> <code>\$entity-&gt;GetSession</code>	Gets the session, which is necessary to invoke many other APIs.
<code>session.OutputDebugString</code> <code>\$session-&gt;OutputDebugString</code>	Captures information that you can use for debugging your hook code or external application.
<code>session.GetEntity</code> <code>\$session-&gt;GetEntity</code>	Retrieves a record from the database.
<code>session.EditEntity</code> <code>\$session-&gt;EditEntity</code>	Edits a record that ClearQuest has retrieved from the database.
<code>[entity.]SetFieldValue</code> <code>\$entity-&gt;SetFieldValue</code>	Assigns a value to a field.
<code>[entity.]Validate</code> <code>\$entity-&gt;Validate</code>	Ensures that the data in a record are acceptable before ClearQuest attempts to save the record to the database.
<code>[entity.]Commit</code> <code>\$entity-&gt;Commit</code>	Commits the record, including any edits, to the database.
<code>[entity.]Revert</code> <code>\$entity-&gt;Revert</code>	Cancels the changes. A good method to use if validation fails and no commit occurs.
<code>[entity.]GetFieldValue</code> <code>\$entity-&gt;GetFieldValue</code>	Retrieves the field info object for the specified field.
<code>FieldInfo.GetValue</code> <code>\$FieldInfo-&gt;GetValue</code>	Retrieves the value(s) of a field.
<code>session.BuildQuery</code> <code>\$session-&gt;BuildQuery</code>	Builds a query.

API Call (VBScript/Perl)	Function
<pre>QueryDef.BuildField \$QueryDef-&gt;BuildField</pre>	Includes a field in a query's search results.
<pre>QueryDef.BuildFilterOperator QueryFilterNode.BuildFilterOpera tor \$QueryDef-&gt;BuildFilterOperator \$QueryFilterNode-&gt;BuildFilter Operator</pre>	Builds a filter operator for a query such as "equal to" or "greater than."
<pre>QueryFilterNode.BuildFilter \$QueryFilterNode-&gt;BuildFilter</pre>	Creates support for a complex query.
<pre>session.BuildResultSet \$session-&gt;BuildResultSet</pre>	Creates the ResultSet object necessary to run a query.
<pre>ResultSet.Execute \$ResultSet-&gt;Execute</pre>	Runs the query with the current ResultSet object.
<pre>ResultSet.MoveNext \$ResultSet-&gt;MoveNext</pre>	Moves the cursor to the next record in the data set.
<pre>ResultSet.GetColumnValue \$ResultSet-&gt;GetColumnValue</pre>	Retrieves the value in the column you specify of the current row.
<pre>session.GetUserLoginName \$session-&gt;GetUserLoginName</pre>	Gets the user's login ID.
<pre>entity.Revert \$entity-&gt;Revert</pre>	Discards any changes made to the Entity object.

Do not use the Revert API to abort the current action from within a hook. This API is only for reverting an action that was explicitly started within a hook or script. If you need to abort the current action, use the exception mechanisms of the scripting language to throw an exception or cause the action-validation hook to return "false."

## Finding hook script text

Use this menu option to find or edit specific script text in hooks used by a given schema. You don't need to specify a given hook; this option searches *all* hooks for the specified script text.

- 1 Open the schema for which you want to search hook script text.
- 2 Select **Edit > Find in Hooks** and enter the text string that you want to search using the following fields:



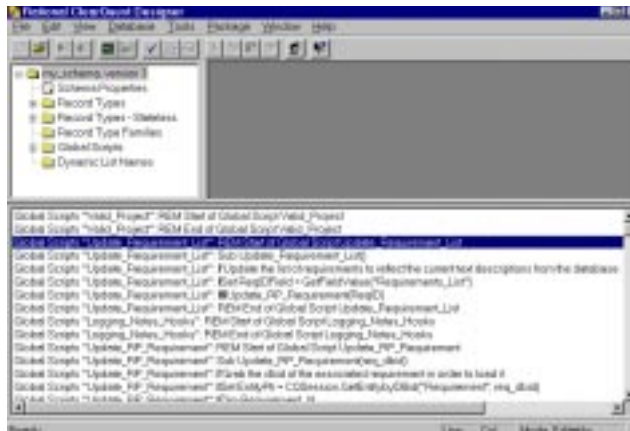
- **Find What**

Type the text for which you want to search.

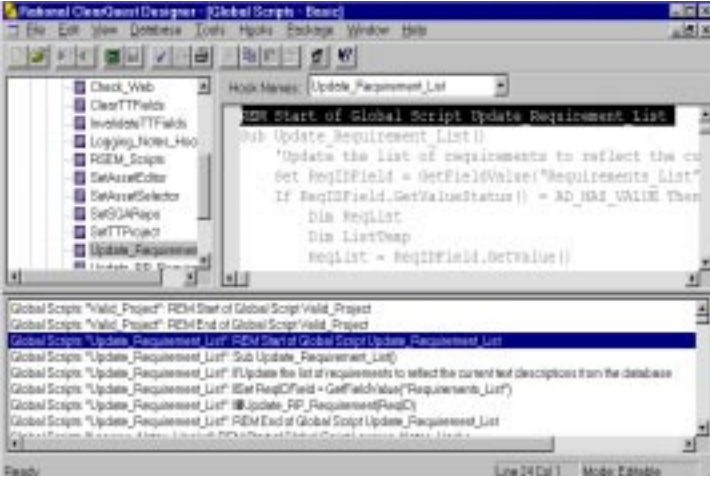
- **Match Case**

Click the check box if you want ClearQuest Designer to search for the text string exactly as you typed it (case-sensitive).

- 3 The Validation pane displays the search results:



Double-click the specific entry you want to view or edit. This brings up an editor window with the specific script and text for editing purposes.



## Examples of common hooks

This section provides several examples of hooks that are commonly used by ClearQuest administrators:

- Hook for creating a dependent list
- Field choice list hook to display user information
- Action initialization hook for a field value
- Action hook for setting the value of a parent record

### Hook for creating a dependent list

The example below assumes that the values you want for the client operating system depend on the values your user selects for the server operating system.

- 1 On the `server_os` field, create a Choice List hook with the enumerated list of values set to Windows NT and UNIX:

VBScript:

```
choices.AddItem("NT")
choices.AddItem("Unix")
```

Perl:

```
push(@choices, "NT", "Unix");
return @choices; #ClearQuest Designer provides this line of code
```

- 2 To prevent your users from adding new members to the list, check the `Limit to List` property.
- 3 To clear the old value in `client_os` when a new value is selected in `server_os`, add the following to the `server_os` Value Changed hook:

VBScript:

```
SetFieldValue "client_os", ""
```

Perl:

```
$entity->SetFieldValue("client_os", "");
```

#### 4 On the client\_os field, create a Choice List hook:

##### VBScript:

```
dim server_os_choice
set server_os_choice = GetFieldValue("server_os")
select case server_os_choice.GetValue()
case "NT"
    choices.AddItem ("Win95")
    choices.AddItem ("NT")
    choices.AddItem ("Web")
case "Unix"
    choices.AddItem ("Web")
end select
```

##### Perl:

```
$server_os_choice = $entity->GetFieldValue("server_os");
$svalue = $server_os_choice->GetValue();
if ($svalue eq "NT") {
    push(@choices, "Win95", "NT", "Web");
} elsif ($svalue eq "Unix") {
    push(@choices, "CQWeb");
}
return @choices; #ClearQuest Designer provides this line of code
```

- 5 In the properties for the client\_os hook, check Recalculate choice list, so that whenever the server\_os field changes, the values recalculate.
- 6 Add the client\_os and server\_os fields to your form using list box controls.

## Field choice list hook to display user information

This hook automatically extracts user login names with access to this database and loads them into your choice list for this field.

### VBScript:

```
Sub AssignedTo_ChoiceList(fieldname, choices)
    ' fieldname As String
    ' choices As Object
    'entityDef = Defect

    Dim sessionObj
    Dim queryObj
    Dim filterObj
    Dim resultSetObj

    Set sessionObj = GetSession()

    ' start building a query of the users
    Set queryObj = sessionObj.BuildQuery("users")

    ' have the query return the desired field of the user object(s)
    queryObj.BuildField ("login_name")

    ' filter for members of group "MyGroup" (whatever group you want)
    Set filterObj = queryObj.BuildFilterOperator(AD_BOOL_OP_AND)
    filterObj.BuildFilter "groups", AD_COMP_OP_EQ, "MyGroup"
    Set resultSetObj = sessionObj.BuildResultSet(queryObj)

    ' run it
    resultSetObj.Execute

    ' add each value in the returned column to the choicelist
    Do While resultSetObj.MoveNext = AD_SUCCESS
        choices.AddItem resultSetObj.GetColumnValue(1)
    Loop
End Sub
```

### Perl:

```
sub AssignedTo_ChoiceList {
    my($fieldname) = @_;
    my @choices;
    # $fieldname as string scalar
    # @choices as string array
    # record type name is Defect
    # field name is myuser
    # use array operation to add items. Example:
    # push(@choices, "red", "green", "blue");
}
```

```

my $session = $entity->GetSession();

# start building a query of the users
my $queryDefObj = $session->BuildQuery("users");

# have the query return the desired field of the user object(s)
$queryDefObj->BuildField("login_name");

# filter for members of group "MyGroup" (whatever group you want)
my $filterOp = $queryDefObj->BuildFilterOperator(
    $CQPerlExt::CQ_BOOL_OP_AND);
my @array = ("MyGroup");
$filterOp->BuildFilter("groups", $CQPerlExt::CQ_COMP_OP_EQ, \@array);
my $resultSetObj = $session->BuildResultSet($queryDefObj);

# run it
$resultSetObj->Execute();

# add each value in the returned column to the choicelist
while ($resultSetObj->MoveNext() == $CQPerlExt::CQ_SUCCESS) {
    push(@choices, $resultSetObj->GetColumnValue(1));
}
return @choices;
}

```



## Action initialization hook for a field value

In this example, when the user invokes the Submit action, the action hook initializes the `problem_description` field with the login name of the person who is submitting the record. That way, users know who created the original change request.

### VBScript:

```
Dim session
Dim loginName

` Get the session
set session = GetSession

` Get the logon name
loginName = session.GetUserLoginName
SetFieldValue "problem_description", loginName
```

### Perl:

```
# session is preset for Perl. No GetSession is required.
# Get the logon name
$loginName = $session -> GetUserLoginName();
$entity -> SetFieldValue("problem_description", $loginName);
```

## Action hook for setting the value of a parent record

Use the following action hook in conjunction with parent/child linking to create a state-based relationship between a parent record and its children. This hook allows you to automatically move the parent record to the next state if all the children are in that state (see “Using fields to link records” on page 75).

**Note:** This code assumes certain field names and record types. If you cut and paste, you will have to make some changes.

### VBScript:

```
Dim SessionObj
Dim ParentID `Dimension variables that hold objects
Dim ParentObj
Dim ChildRefList
Dim ChildArray
Dim ChildID
Dim DefectChildEntityObj
Dim StateStatus
Dim SameState
Dim CurrentState
Dim RetValue
Dim ActionJustPerformed

set SessionObj = GetSession
ThisID = GetDisplayName
ActionJustPerformed = GetActionName
SessionObj.OutputDebugString "action is: " & _
ActionJustPerformed & vbCrLf
StateStatus = ""
SameState = 0

SessionObj.OutputDebugString "current db id is: " & ThisID _
& vbCrLf

ParentID = GetFieldValue("parent").GetValue()
SessionObj.OutputDebugString "parent id is: " & ParentID _
& vbCrLf

if ParentID <> "" then
    set ParentObj=SessionObj.GetEntity("defect", parent_id)
    ChildRefList=ParentObj.GetFieldValue("children").GetValue
    ChildArray=split (ChildRefList, vbLf)

For Each ChildID In ChildArray
    set DefectChildEntityObj=SessionObj.GetEntity("Defect", _
    ChildID)
    CurrentState=DefectChildEntityObj.GetFieldValue _
    ("State").GetValue

    SessionObj.OutputDebugString "StateStatus is: " & _
    StateStatus & vbCrLf
```

```

SessionObj.OutputDebugString "CurrentState is: " & _
CurrentState & vbCrLf

SessionObj.OutputDebugString "SameState is: " & _
SameState & vbCrLf

if StateStatus = "" then
    StateStatus = CurrentState
    SameState = 1

SessionObj.OutputDebugString "coming to statestatus _
is null" & vbCrLf

elseif StateStatus = CurrentState then
SessionObj.OutputDebugString "coming to same state" & _
vbCrLf
SameState = 1

else
SessionObj.OutputDebugString "states are different" & vbCrLf
SameState = 0

end if

Next

if SameState = 1 then
SessionObj.OutputDebugString "samestate = 1, setting _
parent state" & vbCrLf
SessionObj.EditEntity ParentObj, ActionJustPerformed
status = ParentObj.Validate

if (status <> "") then
    SessionObj.OutputDebugString "error when updating parent _
state: " & status & vbCrLf
    ParentObj.Revert
    exit sub
end if

    ParentObj.Commit
end if
end if

```

## Perl:

```

$SessionObj=$entity->GetSession();
$thisID=$entity->GetDisplayName();
$actionJustPerformed->GetActionName();
$ParentID=$entity->GetFieldValue("parent1")->GetValue();
$StateStatus="";
$SameState=0;

# ClearQuest has a message monitor to display
# ClearQuest messages and your messages
$SessionObj->OutputDebugMessage ("perl current db id is:  $thisID\n");
$SessionObj->OutputDebugMessage ("perl parent id is:  $parent_id\n");

if ($ParentID ne "") {
    $ParentObj = $SessionObj->GetEntity("defect", $ParentID);
    $ChildRefList=$ParentObj->GetFieldValue
    ("children")->GetValue();
    $SessionObj->OutputDebugMessage ("children are:
    $ChildRefList\n");
    @ChildArray = split (/\\n/, $ChildRefList);

foreach $ChildID (@ChildArray) {

    $DefectChildEntityObj = $SessionObj->GetEntity("defect",
    $ChildID);
    $CurrentState = $DefectChildEntityObj->
    GetFieldValue("State")->GetValue();

    $SessionObj->OutputDebugMessage ("perl StateStatus is:
    $StateStatus\n");

    $SessionObj->OutputDebugMessage ("perl Current Status is:
    $CurrentStatus\n");

    $SessionObj->OutputDebugMessage ("perl SameState is:
    $SameState\n");

    if ($StateStatus eq "") {

        $StateStatus = $CurrentState;
        $SameState = 1;

        $SessionObj->OutputDebugMessage ("coming to
        statestatus is null\n");

    } elseif ($StateStatus eq $CurrentState) {

        $SessionObj->OutputDebugMessage ("coming to same
        state\n");
        $SameState = 1;

    } else {
        $SessionObj->OutputDebugMessage ("states are
        different\n");
        $SameState = 0;
    } #nested if statements
} #End foreach Loop

```

```

if ($SameState == 1) {
    $SessionObj->OutputDebugMessage("samestate = 1, setting
    parent state\n");
    $SessionObj->EditEntity($ParentObj, $ActionJustPerformed);
    $status = $ParentObj->Validate();

    if ($status ne "") {
        $SessionObj->OutputDebugMessage ("error when updating
        parent state: $status\n");
        $ParentObj->Revert
        return -1; # Exit
    }

    $ParentObj->Commit();
} #end if for ($SameState == 1)
} #end if for ($ParentID ne "")

```



# 8

## Administering ClearQuest Web

ClearQuest Web allows users to access ClearQuest data from a web browser. This chapter describes how to administer ClearQuest Web. It includes the following topics:

- ClearQuest Web considerations
- Limiting access to ClearQuest Web
- Using hooks in ClearQuest Web

**Note:** To set up web support for your users, see the *Installing Rational ClearQuest* guide.

## ClearQuest Web considerations

You should advise your users of the following when they use ClearQuest Web:

- Users cannot drill down (query) on charts.
- Users can use only public charts and reports (they cannot create their own).
- Users can generate reports in HTML format only.
- Users should use the ClearQuest Web buttons to move through the application. They should not use the web browser's Back and Forward buttons to navigate in ClearQuest Web.

## Customizing ClearQuest Web

You can customize the ClearQuest Web interface. For example, you can change the fonts used and the color scheme of the interface. You can also edit performance-related settings.

To edit ClearQuest Web settings, you must have Super User privileges. Log into ClearQuest Web and select **Operations > Edit Web Settings**.



## Limiting access to ClearQuest Web

You can limit access to ClearQuest Web by restricting users or user groups to the following activities:

- Submit records
- Run a single query that you provide

This “web entry” version of ClearQuest Web provides another layer of user privileges.

For example, you can enable beta customers to provide feedback through a password-protected “extranet.” First, you would create a ClearQuest user group (see “Working with user groups” on page 105) that consists of customers you want feedback from, and then distribute the ClearQuest URL to that group. That group will only be able to submit records and use the query that you provide. The query results are read-only.

To configure ClearQuest Web to be a “web entry” client for specific users or user groups, select **Operations > Edit Web Settings** in ClearQuest Web. You must have Super User privileges to configure web entries.

**More information?** For more information, select **Editing Settings** in the ClearQuest Web Help.

## Using hooks in ClearQuest Web

Hooks that you create in your schema will run on the web server with ClearQuest Web. Keep in mind the following when using hooks on ClearQuest Web:

- Hooks for the web must be written in VBScript.
- You must enable dependent fields for ClearQuest Web.
- You cannot use message boxes.
- Context-menu hooks are not supported.
- You can use hooks to detect a web session.

### Enabling dependent fields for ClearQuest Web

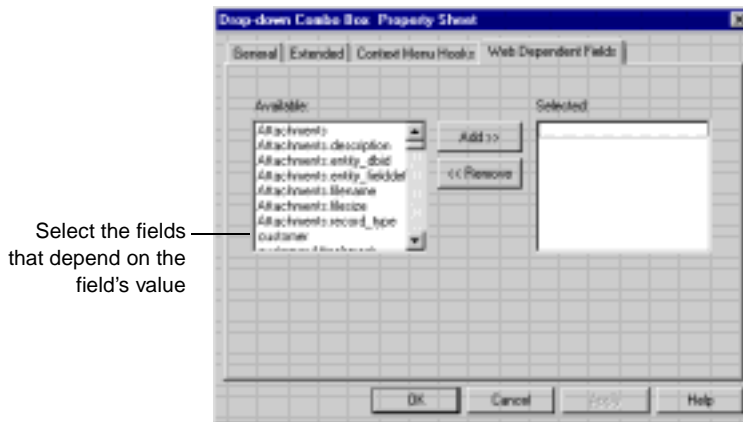
ClearQuest Web is not aware of dependencies that exist between fields. To enable dependent fields for ClearQuest Web:

- 1 Create the dependent field relationship using a value-changed field hook.

When running on ClearQuest Web, field hooks can set values of dependent fields, but they cannot reference field values other than their own.

- 2 Use the appropriate control to add the field to the record form. Both the field that creates the dependency and the dependent fields must use one of the following controls. You can mix and match them:
  - Drop-down list box
  - Combo box
  - Drop-down combo box
- 3 Right-click the control on the form and select **Properties** from the shortcut menu.

In the control's property sheet, use the Web Dependent Fields tab to specify the fields that depend on the respective field's value. Only the parent field of the dependency needs to be web-enabled.



**Note:** To update the choice list associated with the dependent field, select Recalculate Choice List when you create the choice list in the field. ClearQuest recalculates the contents of the list before displaying it to the user. This can decrease web performance.

## Displaying messages on ClearQuest Web

Functions that call other Windows applications, such as a message box, cause the web client to freeze. For example, if a message box function runs on a web server, the message box pops up on the server's screen. Since the user cannot click OK on the server, the client is left waiting. This requires you to reboot the web server.

However, if record script hooks return a string value, that string is displayed to the user.

## Using hooks to detect a web session

When writing hooks, you can use the ClearQuest API to detect whether a user is on a web browser rather than in the native client. This allows you to take appropriate action if you have not adjusted your schema to match the functionality available on the

web. For example, when you detect a web session in a function that creates a message box or a new window, you can call code modified for the web environment or exit the function.

### ***Web session detection in VBScript***

```
dim currDBSession      ' Current Db session

set currDBSession = GetSession
' Test for existence of the web session variable
if currDBSession.HasValue ("_CQ_WEB_SESSION") then
' Either exit or do something else
end if
```

### ***Web session detection in Perl***

```
my $currDBSession; # Current Db session

$currDBSession = $entity->GetSession();
# Test for existence of the web session variable
if ( $currDBSession->HasValue ("_CQ_WEB_SESSION") {
# Either exit or do something else
}
```

# 9

## Administering ClearQuest E-mail

ClearQuest takes advantage of e-mail in two ways:

- ClearQuest can send e-mail notification to a user or a user group when a record meets the criteria you define. For example, you can send e-mail to your testing team as soon as a defect is fixed. To do this, you set up e-mail rules in ClearQuest client.
- ClearQuest users can submit and modify records by e-mail. For example, users can respond to an e-mail notification with notes that can be appended to a record in the ClearQuest database. To set up e-mail submission capability, you use the Rational E-mail Reader.

**Note:** You can combine e-mail notification with e-mail submission to update records using “round-trip” e-mail.

This chapter describes how to enable ClearQuest e-mail capabilities. The topics covered include:

- Enabling automatic e-mail
- Submitting records by e-mail
- Using “round-trip” e-mail

## Enabling automatic e-mail

To enable ClearQuest to send automatic e-mail:

- You must set up e-mail rules in ClearQuest client.
- Each ClearQuest client user must enable e-mail notification on his or her own machine.

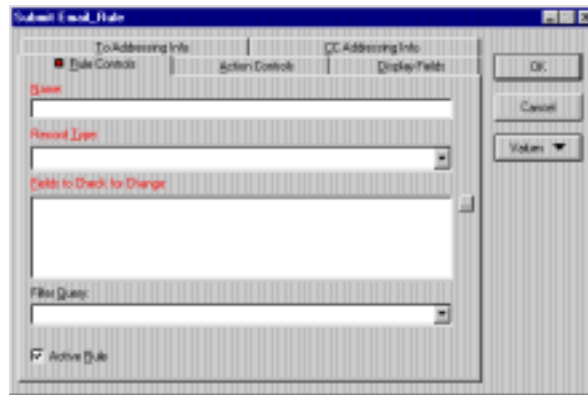
### Setting up e-mail rules

E-mail rules are used to set up e-mail notification for ClearQuest users. For example, you can create an e-mail rule that sends e-mail to the quality-assurance team when a defect is resolved. You can set up the e-mail to include any of the fields of the resolved defect.

You create e-mail rules by submitting Email\_Rule records from the ClearQuest client. To add or modify an e-mail rule, you must have Super User or Schema Designer privilege (see “ClearQuest user privileges” on page 99).

To set up an e-mail rule, select **Actions > New** in ClearQuest client and select the **Email\_Rule** record type.

Use the **Email\_Rule** record type to set up the parameters used to determine when e-mail is sent, which users or user groups it is sent to, and the content of the e-mail message.



When you create an e-mail rule, you establish the criteria that triggers e-mail notification. You can send e-mail based on the following events:

- A field's value changes in a record.
- A specific action occurs in a record.
- A record matches the criteria of a specific query.

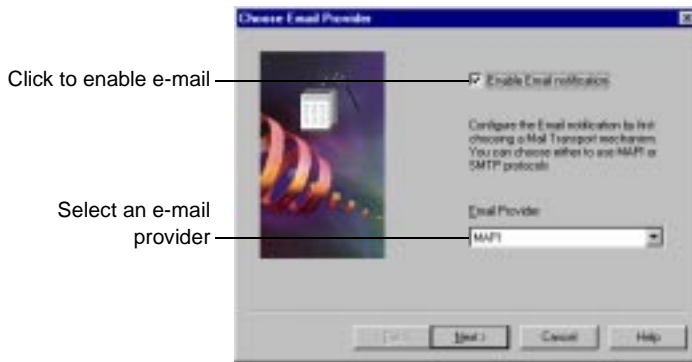
**Note:** Most ClearQuest predefined schemas include the Email\_Rule record type. If the schema you are using does not, you can add this functionality by adding the Email package (see Appendix A, "ClearQuest schemas and packages").

**More information?** Look up *e-mail* in the ClearQuest Help index.

## Configuring ClearQuest clients to send e-mail

To send notification e-mail messages, ClearQuest users must configure their e-mail settings and enable e-mail notification. This allows individual client machines to use the ClearQuest e-mail rules you set up.

Users set up their e-mail by choosing **View > E-mail Options** in ClearQuest.



If you click on enable e-mail notification, click **Next** to define the e-mail provider parameters.

**Note:** To configure ClearQuest Web to send e-mail, see “Installing ClearQuest Web” in the *Installing Rational ClearQuest* guide.

**More information?** Look up *e-mail* in the ClearQuest Help index.



## Submitting records by e-mail

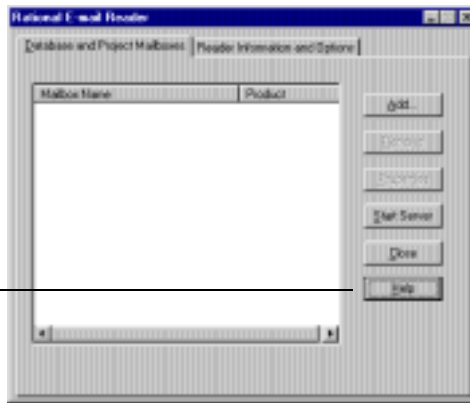
To enable your users to submit and modify records by e-mail, you must configure the Rational E-mail Reader. The E-mail Reader is an e-mail processing tool that parses e-mail submissions and commits new records or changes to the database. You only need to run one instance of the E-mail Reader on a machine in the same domain as the ClearQuest database server.

### Configuring Rational E-mail Reader

The E-mail Reader (`mailreader.exe`) is automatically installed in the `\Rational\common` directory. You can drag a copy of the `mailreader.exe` to the desktop to create a shortcut to it.

To start the E-mail Reader, double-click `mailreader.exe`.

Click **Help** and follow the instructions to set up E-mail Reader



Keep in mind the following when configuring the E-mail Reader:

- The E-mail Reader requires a dedicated e-mail account that has been assigned for parsing records.
- You must designate one machine to run the E-mail Reader. To ensure that e-mail services are always available, run the E-mail Reader on the same machine as your database server.
- Any e-mail sent must be in the required format. See “Formatting e-mail for submission” on page 154.

- You should set up defaults to be used. This allows users to quickly respond to e-mail notification without taking into account the required format.
- Field values submitted by e-mail overwrite the existing data. If you want to append existing data, you must specify a field that allows users to append data, such as the Notes Log (available in most predefined schemas).

### Formatting e-mail for submission

Any e-mail sent must conform to the format shown below. If it does not conform to the following format, the e-mail submission will fail and the e-mail will be deleted.

Subject: <record type> <action> <record ID>

Body: <fieldname: legal value>  
 <fieldname: legal value>  
 {<multiline\_fieldname: This is an example of a field valuefor a field whose data type is a multi-line text field. It requires curly braces>  
 } *Note that the right curly brace (}) must be on a separate line from the multi-line text it encloses.*

The Subject line follows these guidelines:

- The record type is always required and is listed first.
- The action name is listed after the record type. If you have set up defaults, you can omit the action (see the example below). Users can override the defaults by specifying a different action. If you do not provide an action, and defaults have not been set up, the submission will be rejected.
- The record ID is listed after the action, but is not necessary when submitting new records.

**Note:** You can configure the E-mail Reader to track errors and to e-mail you when a record fails to be committed to the database.

### ***E-mail format example***

In this example, you have set up the default action as **Modify** and the default field to be **Notes Log**. The **Subject** line requires the record type and record ID, but because of the defaults, the action name is optional.

**Subject:** RE: defect SAMPL00000017

**Body:** {Today we posted a patch for this problem on the company intranet. Please download the patch to solve your customer's problem.  
}

**More information?** Look up *e-mail* in the ClearQuest Designer Help index.

### ***Guidelines for using the e-mail format***

Keep in mind that your schema rules, such as required fields and legal values, apply when users submit or modify records with e-mail. For example, if the user specifies the **Submit** action, but fails to include all the required fields for the **Submit** state, the e-mail is sent back to the sender with an error message.

Use the following guidelines when sending e-mail:

- Use curly braces for a text field with multiple lines. *Note that the right curly brace (}) must be on a separate line from the multi-line text it encloses.*

```
{description: my description here... multiple lines follow...  
}
```

- If legal values are not used, the Rational E-mail Reader sends an error message to the submitter.
- E-mail submission does not support adding attachments.
- Fields can be included in any order.
- The e-mail must include values for any required fields.

- Include only fields that you want to overwrite. To avoid overwriting updates made by other users, use a field that appends text and preserves earlier entries.
- Do not supply a value for a read-only field (for example, defect ID), because ClearQuest does not change such fields.
- ClearQuest's own e-mail notification mail uses the required format automatically. Users can use this mail as a template for sending additional records or modifications via e-mail.

**Note:** You can configure the E-mail Reader to keep a log or send e-mail to you whenever a submission fails.

## Using “round-trip” e-mail

Your team can use e-mail notification along with e-mail submission to update records. This combination is called “round-trip” e-mail. Round-trip e-mail requires that you use the same account specified in the mailreader configuration for the From Address in your e-mail rule.

Here's an example of a typical sequence of events:

- 1 A user submits a record by e-mail, using the required format. (See “Formatting e-mail for submission” on page 154.) ClearQuest assigns the new defect ID number PROD0000137.
- 2 An e-mail rule you established notifies the lead engineer of this defect. (See “Setting up e-mail rules” on page 150.)
- 3 The lead engineer reads the e-mail entitled “PROD0000137,” then responds by e-mail, including remarks in the description field, using the required format.

**Note:** The reply should include only fields that you want to modify. Any field included in the message will overwrite existing data.

- 4 The Rational E-mail Reader processes the e-mail message and submits the modified record to the ClearQuest database.

# 10

## Importing data into ClearQuest

This chapter describes how to import data from an existing defect tracking system into ClearQuest.

Topics covered include:

- Preparing for a successful data import
- Creating a ClearQuest import file
- Performing the data import

## Preparing for a successful data import

Before you begin importing data, plan your conversion carefully:

- Analyze your current data and workflow so that you know how to implement it in ClearQuest. For an overview of how to work with ClearQuest, see Chapter 2, “Understanding ClearQuest administration.”
- Create an import schema to support the data you want to import. See “Creating an import schema” on page 158.
- Upgrade an existing user database or create a new database and associate it with the import schema. See “Creating a database for imported data” on page 160.
- Export your existing data to text files that use the ClearQuest import-file format. See “Creating an import schema,” below.
- Use the ClearQuest Import Tool to import the data from the import files into your user database. See “Performing the data import” on page 166.

**Note:** Before importing all of your data, first test the entire import process on a subset of your data. See “Testing the import process” on page 160.

### Creating an import schema

Before importing data into ClearQuest, you must create an import schema, or modify an existing schema, to support the data you want to import. Keep in mind the following when creating an import schema:

- The import schema must contain the record types and fields with the appropriate behaviors and data types to support your data. The schema must also contain the actions and state transitions that you want to use with the imported records.
- The schema must contain a form that users can use to view and modify the imported data.
- For records that contain reference lists, the schema must contain a record type that is the destination for the reference.

For example, if you are importing defect records that contain a field for a project, you must create the project record type before importing the defect records and then populate the project record type with project names.

- If you are importing history, attachments, or duplicate records, or if you are upgrading existing records, the schema must contain a field to store the old ID values.
- If you are importing choice lists, make sure that the schema contains values for the choice list that match the values specified by the imported records.

**Note:** ClearQuest does not validate data types during import. If the records you are importing contain data types that ClearQuest does not support, you can map those data types to other ClearQuest types. By default, ClearQuest maps unsupported data types as `STRING` type. See “Data types supported in ClearQuest” on page 163.

It is important to map all fields and states in your current system to a field and state in ClearQuest. If you do not provide a mapping between an exported field and a ClearQuest field, the data for that field is not imported. If the state field is not mapped, ClearQuest defaults all records to the Submitted state, thereby making your state model ineffective.

**More information?** See Chapter 4, “Working with ClearQuest schemas,” and Chapter 5, “Customizing a schema.”

### ***Original record ID***

ClearQuest assigns a new record ID to each imported record, so be sure that your import schema has a field to contain the original record ID. ClearQuest uses the original record ID when importing duplicates, history, and attachments. It is also useful for finding records based on the original record ID.

**Note:** If you use the ClearQuest Export Tool to export data from another ClearQuest database, map the ID field in the import file to the original (old) record ID field.

## Creating a database for imported data

Before importing data into ClearQuest, you must create a user database to hold the imported data and associate this database with the import schema. You can use an existing user database that's already associated with the schema you've customized to be the import schema. Just upgrade the user database with the newer "import" version of the schema. Remember, however, that you can only upgrade a user database with a newer version of the same schema, not with a different schema.

**More information?** See "Upgrading a user database" on page 54. Look up *databases, creating* in the ClearQuest Designer Help index.

## Testing the import process

Before you import all of your data into ClearQuest, you should test the entire import process on a subset of your data:

- Export a subset of your existing data to a ClearQuest import file.
- Run ClearQuest Import Tool to import the data.
- Work with the data in the ClearQuest client to see if your import schema functions as expected.
- Modify the import schema as necessary based on your test, then test the process again.



## Creating a ClearQuest import file

Before you can import data into ClearQuest, you must use the tool of your choice to export the data from your current system into a delimited text file that uses the ClearQuest import-file format. You must create separate import files for each record type and for history and attachments. This section describes the file-format requirements for the record, history, and attachments import files.

**Note:** Make sure you have a reliable way to export data from your old system. Test the export several times to be sure the results are consistent.

### Formatting the record import files

The import file for a record type is a list of delimited, double-quoted strings ending with a newline character. The most common file format is comma-delimited, but you can also use colons, semi-colons, pipes, or tabs.

The first row of the import file contains a list of the field names being exported. Subsequent rows contain the field values for the individual records, ordered according to the order of the field names in the first row. During import, ClearQuest converts the value for each field to the corresponding field type defined in the ClearQuest schema. For example:

```
"id","state","submitdate","severity","priority","summary","description"
"1","Submitted","4/11/00 7:00.00","3-Workaround","3","The shortcut for
"Printing" is grayed out,"See summary --John"
"2","Opened","4/14/00 11:32.00","1-Crash","1","Can't login to the
system","I typed my login and the hourglass sign appears, but after
15 minutes, I still can't type my password. There's an infinite
loop somewhere."
```

The value of the State field determines the current state of the record when it is imported. ClearQuest validates the fields of imported records, but does not validate the actions required to reach a particular state.

### ***Important import-file format considerations***

Keep in mind the following when formatting an import file:

- In a comma-delimited data file, field values can contain embedded commas as long as they are enclosed within the quotes surrounding the field. For example:

"I typed my login and the hourglass sign appears, but after 15 minutes, I still can't type my password. There's an infinite loop somewhere."

- Embedded double quotes in field values must be enclosed within more double quotes. For example:

"The shortcut for ""Printing all"" does not work."

- If a field is empty, use " ", "<<None>>", or "<<Unassigned>>".
- Do not include spaces before or after the delimiter character.
- Before you can import reference fields, you must first import values for the referenced record types. The values that the import file has for those fields must already exist in ClearQuest. This includes users.
- Items in a reference list must be comma-separated in the import file (for example, "Ref1, Ref2, Ref3").

### ***Data types supported in ClearQuest***

The values of the fields in the import file are interpreted according to the data type defined for the corresponding field in the ClearQuest schema. ClearQuest supports the following data types:

<b>Data Type</b>	<b>Description</b>	<b>Behavior</b>
ATTACHMENT_LIST	A list of fields with type Attachment	Interpreted as a list of pathnames to attachments
DATE_TIME	SQL date and time	Converted to date/time
INT	SQL integer	Converted to an integer
MULTILINE_STRING	A variable-length character string of unlimited size	Inserted as is to the maximum length in the schema for this type
REFERENCE	A reference to a display name in a state-based or stateless record type	Interpreted as a key to a stateless record type
REFERENCE_LIST	Multiple references to display names in a state-based or stateless record type	Interpreted as a list of references. Note that reporting is not supported for REFERENCE_LIST fields.
SHORT_STRING	A variable-length character string with a maximum length of 254 characters	Inserted as is
STATE	Reserved for system fields	CQ validates value of the field, but does not trigger actions required to reach the state

**Note:** ClearQuest maps unsupported data types as `STRING`. You map the data types and fields during the actual import process using the ClearQuest Import Tool (described later in this chapter).

## Formatting the history import file

The format for the history import file requires that each history be its own row, and that each entry have an original ID number identifying the record to which the history belongs. The remaining fields contain information on the action that was performed, and should include the following fields:

- original record ID
- timestamp
- user name
- action performed
- old state
- new state

**Note:** If you use the ClearQuest Export Tool to export data from another ClearQuest database, the Export Tool saves the original (old) record ID in a field called “display\_name.” Map the display\_name field to the original (old) record ID.

Below is an example of a history import file. It uses the original IDs of the records for which this is the history:

```
"id","timestamp","user_name","action_name","old_state","new_state"  
"1","Apr 6 2000 8:31AM","srahman","close","open","closed"  
"2","Apr 6 2000 9:40AM","srahman","verify","closed","verified"
```

The date must specify the full year. You can use the following date formats:

- "6 April 2000"
- "April 6, 2000 8:30:00"
- "8:30:00 Apr 6 2000"
- "4/6/2000 8:30:00PM"

## Formatting the attachments import file

The import file format for attachments requires that each entry have an original ID number identifying the record to which the attachments belong. The remainder of each entry lists the attached files associated with each attachment field of the record. The pathnames for the attached files of a single attachment data type field are grouped together inside one set of quotation marks and separated by delimiter characters.

**Note:** If you use the ClearQuest Export Tool to export data from another ClearQuest database, the Export Tool saves the original (old) record ID in a field called “display\_name.” Map the display\_name field to the original (old) record ID.

The following example associates three attached files with a record whose original ID is 101. The schema contains two attachment fields, attfield1 and attfield2:

```
"id","attfield1","attfield2"  
"101","c:\temp\101_1.txt,c:\temp\101_2.txt","c:\temp\101_3.txt"
```

In this example, attfield1 has two attached files associated with it. A comma separates the pathnames for the files. ClearQuest stores the actual contents of the attachments, not just a reference to them, and uses the pathname to locate and read the file.

## Performing the data import

After you set up a user database to receive your data and export the data into an import file, you are ready to perform the data import. You must perform the import at least three times, once for each record type and for history and attachments.

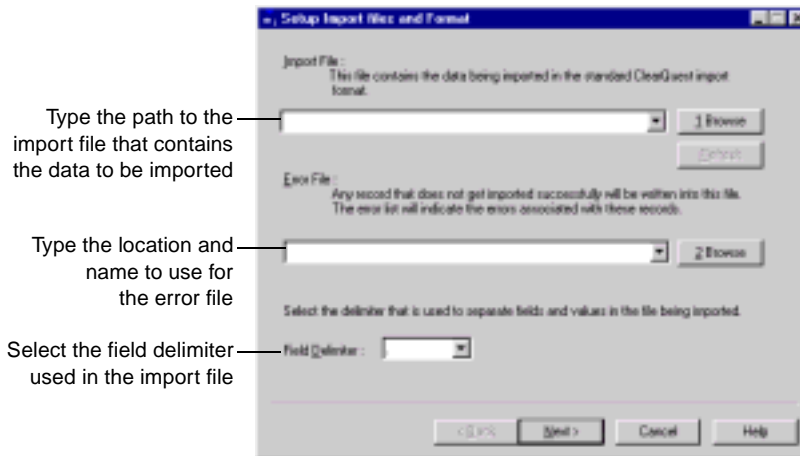
If you have multiple record types, you must import them in an order that allows referencing. For example, if you have a defect record type that refers to a project record type, first import the project record, then import the defect record.

Consider importing all of the records in your system, even those that have been resolved. By including all records, you can access historical information about your projects and immediately generate useful management reports.

**Note:** You can also import duplicate records separately. See “Importing duplicate records” on page 173.

To perform the import for records:

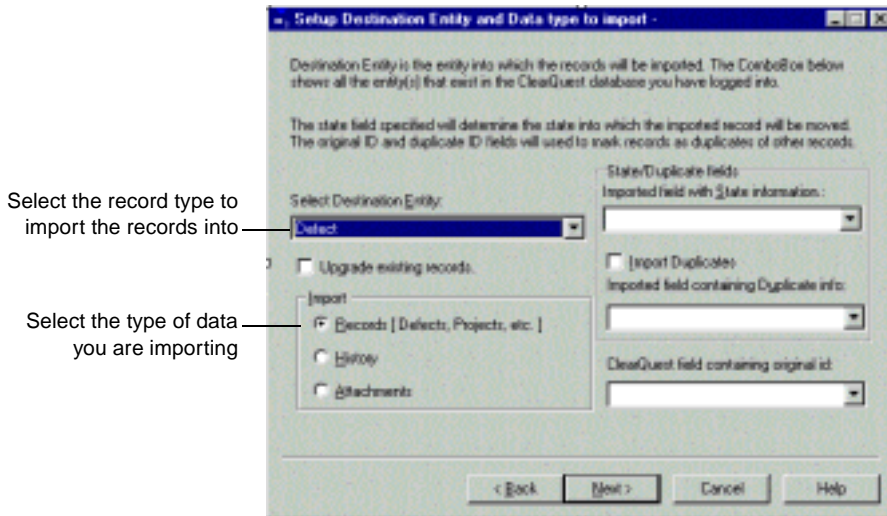
- 1 Select **Rational ClearQuest Import Tool** from the Start menu. Type your user name and password, then select the database to import into. Click OK to open the Setup Import files and Format dialog.
- 2 In the Setup Import files and Format dialog:



When you're done, click Next to open the Setup Destination Entity and Data Type to Import dialog.

**Note:** If a record does not convert successfully, it is saved to the error file you specify. You can use this error file to fix problems and to reimport the record. See “Importing records from the error file” on page 174.

**3** In the Setup Destination Entity and Data Type to Import dialog:



**Note:** In the ClearQuest Import and Export Tools, the term *Entity* refers to a record type.

If the destination entity has states, select the **Imported field with State information**. Specify the exported field that maps to the state field you defined in the import schema. The state field specified determines the state the record is imported into. If this field is empty, ClearQuest defaults to the Submit state.

**Note:** The state names in your import file must match those you defined in the import schema. If you want to use different state names, you must edit your import file to use the new names. For example, if your current application uses the state name

Submitted, but you called that state New in the import schema, you must edit your import file and replace Submitted with New.

If the imported records contain duplicates, click **Import Duplicates** and select the exported field name that contains the duplicate information. For more information on importing duplicates, see “Importing duplicate records” on page 173.

Select the ClearQuest field containing original ID. This is the field in the import schema that contains the original record ID of the exported data. This field is important for importing history, attachments, and duplicate records.

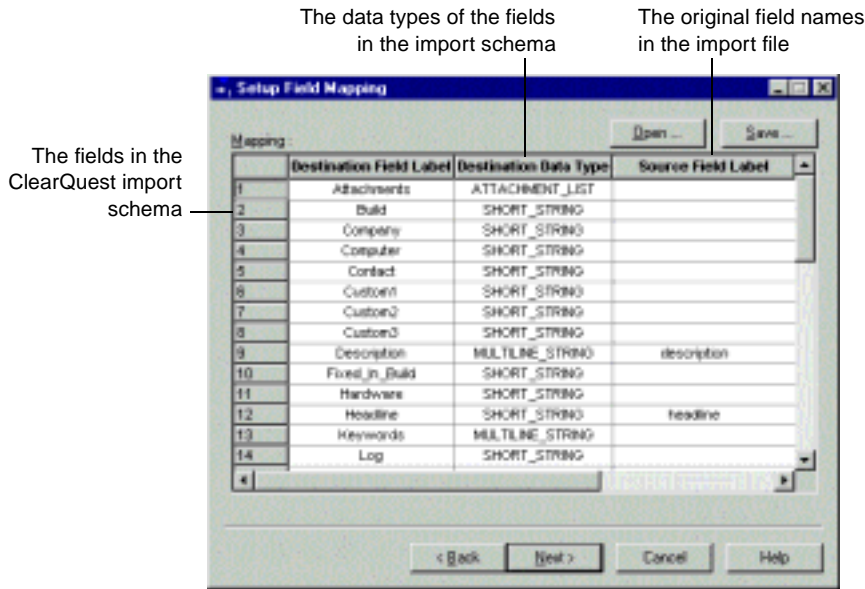
**Note:** ClearQuest allows you to update records that you previously imported. If you are upgrading previously imported records, history, or attachments, click **Upgrade existing records**.

If you used the ClearQuest Export Tool to export duplicate record information separately, click **Upgrade existing records** and **Import Duplicates**. This adds the duplicate information to the existing records. If you are importing duplicate records from another application, see “Importing duplicate records” on page 173.

When you're done, click **Next** to open the Setup Field Mapping dialog.



- In the Setup Field Mapping dialog, create a one-to-one mapping between the fields in the import file and the fields you created in the ClearQuest import schema. If the field names are the same, the mapping is done automatically.



- You cannot map a field in the import file to more than one field in the ClearQuest schema.
- If the time field is mapped to a date-time field in ClearQuest, today's date is appended to the time. Also, if the date field is mapped to a date-time field in ClearQuest, the current time is appended to the date.

Click Save to save the mappings to a text file. This allows you to reopen the mapping file if you need to repeat the import process.

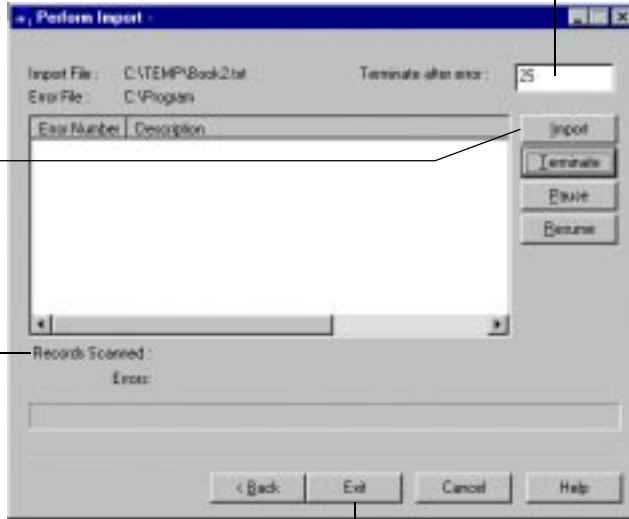
When you're done, click Next to open the Perform Import dialog.

## 5 In the Perform Import dialog:

Specify how many errors can occur before the import process is terminated

Click **Import** to begin the process

ClearQuest displays the number of errors and the number of records scanned



When the import is finished, click **Exit** to close the Import Tool

## Importing history

After importing your records, you are ready to import your history information. History information is important if you want historical and trend-analysis reports to work.

Before importing history information, be sure you have an import file containing the history fields. See “Formatting the history import file” on page 164.

To import history, follow the procedure for “Performing the data import” on page 166, with the following changes:

- In step 2, in the Setup Import files and Format dialog:
  - For Import File, type the path to the history import file.
  - For Error File, type the location and name to use for the error file. If you have an error file from importing records, use a different name for the history error file.
- In step 3, in the Setup Destination Entity and Data Type to Import dialog:
  - For the Destination Entity, select the owning record type.
  - For Import, select History.
  - Enter the name of the ClearQuest field of the owning record type containing the original record ID in the import file. ClearQuest must know the record to which the history information belongs.
- In step 4, in the Setup Field Mapping dialog, map your history fields to the corresponding fields in your ClearQuest schema, and save the map file.

All other steps remain the same.

## Importing attachments

After importing your records, you are ready to import your attachments. Before importing attachments, be sure you have an import file containing the attachment information. See “Formatting the attachments import file” on page 165.

To import attachments, follow the procedure for “Performing the data import” on page 166, with the following changes:

- In step 2, in the Setup Import files and Format dialog:
  - For Import File, type the path to the attachment import file.
  - For Error File, type the location and name to use for the error file. If you have an error file from importing records, use a different name for the attachment error file.
- In step 3, in the Setup Destination Entity and Data Type to Import dialog:
  - For the Destination Entity, select the owning record type.
  - For Import, select **Attachments**.
  - Enter the name of the ClearQuest field of the owning record type that contains the original record ID of the exported data. ClearQuest must know the record to which the attachment belongs.
- In step 4, in the Setup Field Mapping dialog, map your attachment fields to the corresponding fields in your ClearQuest schema, and save the map file.

All other steps remain the same.

## Importing duplicate records

You can import duplicate records when importing all of your records. To import duplicate records, considering the following requirements:

- Imported duplicates must have a pointer to their original ID.
- You must indicate which field in your ClearQuest import schema maps to the original record ID. ClearQuest uses the original record to locate the parent of the duplicate record.
- Within ClearQuest, duplicate records are handled using a Duplicate state and action. Be sure your schema includes both.

**Note:** If you use the ClearQuest Export Tool to export data from another ClearQuest database, a separate import file for duplicate records is created automatically.

To import duplicate records:

- 1 Follow the steps for “Performing the data import” on page 166.
- 2 In step 3, in the Setup Destination Entity and Data Type to Import dialog, click **Import Duplicates** and enter the name of the field that contains the original record ID (the ID of the record this is a duplicate of).
- 3 Run the import. Any duplicate that cannot reference its parent record (because it was imported before the parent) is saved to the Error file.
- 4 When the import is completed, use the Error file to import the duplicates that did not successfully import the first time.

### *Importing duplicate records separately*

Another method for importing duplicate records is to create a separate import file for duplicate records, just as you do for history and attachments. You can then import all of your other records (excluding the child records) and import the duplicates later.

## Importing records from the error file

If errors occur during the import, ClearQuest creates the following files:

- `error file`: ClearQuest saves the unimported records to the error file whose name and location you defined in step 2 above.
- `errlog.txt`: ClearQuest saves error messages to a text file in your system Temp directory.

To reimport these problem records:

- 1 Check the `errlog.txt` file and review the types of errors encountered.
- 2 Open the error file containing the unimported records.
- 3 Correct the errors in the records. Be sure the error file uses the import file format. See “Formatting the record import files” on page 161.
- 4 Perform the import process again, this time specifying the error file as the import file.

## Upgrading existing records

You can use the ClearQuest Import tool to update records that you previously imported. For example, if you modified records in your old system that have already been imported into ClearQuest, you can update the imported records with the new data.

**Warning:** Be sure you do not change the records in both locations. If you upgrade existing records that have already been modified in ClearQuest, you will lose your changes.

To upgrade existing records, follow the procedure for “Performing the data import” on page 166. In step 3, in the Setup Destination Entity and Data Type to Import dialog, select **Upgrade existing records**. All other steps remain the same.

# A

## ClearQuest schemas and packages

ClearQuest includes several predefined schemas: There's a schema for each Rational Windows suite, and a "Blank" schema, containing only required system fields, that you can use to build a schema from scratch.

Each ClearQuest predefined schema consists of a number of schema packages that provide specific functionality or specific support for integration with Rational suites. You can customize an existing schema by adding one or more of these packages to the schema.

This appendix describes ClearQuest schemas and packages. The topics covered include:

- ClearQuest predefined schemas
- ClearQuest packages
- State model for the Defect record type
- State model for the EnhancementRequest record type
- State-type models for packages

**Note:** When you upgrade to a new version of ClearQuest, you may need to apply the upgrade packages to your existing schema to take advantage of new ClearQuest functionality. See the release notes for a list of upgrade packages that you may have to apply.

**More information?** For instructions on how to add packages to a schema, see "Adding packages to a schema" on page 56. For information on how to customize a schema, see Chapter 5, "Customizing a schema."

## ClearQuest predefined schemas

ClearQuest includes the following predefined schemas:

Schema	Description	Packages included
Blank	Contains only system fields. Use Blank to create a schema from scratch.	None
<b>Common</b>	Contains fields and record types that are common to all predefined schemas. Each ClearQuest schema consists of this schema and one or more packages.	None
DefectTracking	Contains the fields necessary to start using ClearQuest to track defects in a software development environment.	Attachments, Customer, Email, History, Notes, Project, Resolution
AnalystStudio	For use with Rational Analyst Studio. Contains fields and rules that work with Rational Rose and RequisitePro.	Attachments, Email, EnhancementRequest, History, Notes, Repository, RequisitePro, Resolution, RequisiteProSupplement
DevelopmentStudio	For use with Rational DevelopmentStudio. Contains fields and rules that work with Rational Purify, Quantify, and Pure Coverage.	Attachments, Email, EnhancementRequest, History, Notes, PQC, Repository, RequisitePro, Resolution, RequisiteProSupplement
TestStudio	For use with Rational TestStudio. Contains fields and rules that work with Rational TeamTest, RequisitePro, Purify, Quantify, and Pure Coverage.	Attachments, Email, EnhancementRequest, History, Notes, PQC, Repository, RequisitePro, Resolution, TeamTest, RequisiteProSupplement
Enterprise	For use with Rational EnterpriseStudio. Contains fields and hooks that work with most Rational Suite products.	Attachments, BaseCMAActivity, Email, EnhancementRequest, History, Notes, PQC, Resolution, Repository, RequisitePro, TeamTest, UCMPolicyScripts, UnifiedChangeManagement, RequisiteProSupplement



---

<b>Schema</b>	<b>Description</b>	<b>Packages included</b>
UnifiedChange Management (UCM)	Supports the UCM process by providing integration with Rational ClearCase.	Attachments, BaseCMAActivity, Email, History, Notes, Resolution, UCMPolicyScripts, UnifiedChangeManagement

---

## ClearQuest packages

ClearQuest includes the following packages. Some packages are read-only; their functionality cannot be changed.

Package	Description	Record type(s) added/modified	Field(s)
Attachments (read-only)	Allows you to add and remove attachments related to a record.	Adds an Attachments tab to the enabled record type.	<b>Fields added to enabled record type:</b> Attachments
BaseCMAActivity	Provides support for the BaseCMAActivity record type. Included in the UCM and Enterprise schemas as a lightweight activity record type. You can use this alternative to the Defect record type as is, enable it for Unified Change Management (UCM), or develop it into a new record type.  For more information, see <i>Managing Software Projects with ClearCase</i> .	Adds the BaseCMAActivity record type.	<b>Fields included in BaseCMAActivity record type:</b> Owner Description Headline
ClearCase (read-only)	Provides support for integration with Rational ClearCase.  For UCM integration, see “UnifiedChange Management (UCM) (read-only)” on page 10.	Adds the cc_change_set and cc_vob_object record types.  Adds the ClearCase tab to the enabled record type.	<b>Fields included in cc_change_set record type:</b> objects  <b>Fields included in cc_vob_object record type:</b> name object_oid vob_family_uuid  <b>Fields added to enabled record type:</b> cc_change_set

<b>Package</b>	<b>Description</b>	<b>Record type(s) added/modified</b>	<b>Field(s)</b>
Customer	Supports the integration of customer data with your defect/change tracking system.	<p>Adds a customer stateless record type.</p> <p>Also adds reference fields for customer information to the record type you choose.</p>	<p><b>Fields included in the Customer record type:</b></p> <p>Attachment CallTrackingID Company Description Email Fax Name Phone</p> <p><b>Fields added to enabled record type:</b></p> <p>Customer Customer_Severity</p>
Enhancement Request	Supports an additional record type for product enhancement requests.	Adds the EnhancementRequest record type.	<p><b>Fields included in EnhancementRequest record type:</b></p> <p>Customer_Company Customer_Email Customer_Name Customer_Phone Customer_Priority Description Headline Keywords Owner Priority Product Product_Area Request_Type Submit_Date Submitter Target_Release</p>

Package	Description	Record type(s) added/modified	Field(s)
Email (read-only)	Supports automatic e-mail notification when records are modified.	Creates an Email_Rule stateless record type.  Adds a base action to the enabled record type called "Send_Email_Notif." This base action runs the e-mail rule whenever any action is invoked.	<b>Fields included in Email_rule record type:</b> Actions Action_Types CC_Actioner CC_Addr_fields CC_Additional CC_Groups CC_Users Change_Fields Display_Fields Entify_Def (record type) From_Addr Include_Defect Is_Active_Rule Filter_Query Name Show_Previous Source_States Subject_Fields Target_States To_Additional To_Addr_Fields To_Groups To_Users
History (read-only)	Allows you to keep a historical account of all actions taken on a record.	Adds a History tab to the enabled record type.	<b>No fields added.</b>
Notes	Allows you to keep a historical account of all notes that have been entered on a record, according to date and user.	Adds a Notes tab to the enabled record type Also adds a base action called "Init_Note_Entry" in the enabled record type.	<b>Fields added to enabled record type:</b> Note_Entry Notes_Log
PQC (read-only)	Provides support for integration with Rational Purify, Quantify, and Pure Coverage.	Adds a PQC tab to the form of the enabled record type.	<b>Fields added to enabled record type:</b> PQC_DiagnosticTool PQC_Executable PQC_TestCommand PQC_TestTool PQC_Stack PQC_StackID

Package	Description	Record type(s) added/modified	Field(s)
Project	<p>Allows you to track records according to project.</p> <p>(Note: No relation to the UCM package "Project" concept.)</p>	Creates a Project stateless record type.	<p><b>Fields included in Project record type:</b> Name Description</p> <p><b>Fields added to enabled record type:</b> Project</p>
Repository (read-only)	Provides support needed for both Rational RequisitePro and Rational TeamTest.	Creates a stateless record type called "repoproject" that allows you to use TeamTest or RequisitePro projects with ClearQuest. Also adds a RepoConnect base action to the enabled record type.	<p><b>Fields included in repoproject record type:</b> Name TT_Repo (Refers to the TeamTest Repository)</p> <p><b>Fields added to enabled record type:</b> RepoProject</p>

Package	Description	Record type(s) added/modified	Field(s)
RequisitePro (read-only)	Provides support for integration with Rational RequisitePro.	<p>Adds the Requirement and RequirementMap stateless record types.</p> <p>Adds the Requirements tab to the enabled record type.</p> <p>Also adds the ASCQIBase base action to the enabled record type.</p>	<p><b>Fields included in Requirement record type:</b>  Project_Name  Req_GUID  Req_ID  Requirement  Tag</p> <p><b>Fields included in RequirementMap record type:</b>  CQDatabase  CQDatabasePath  CQEntityDefName  CQReqListAttName  CQBackReqListAttName  CQRepoProjectAttName  CQModifyAction  CQDialogTitle  RPProjectName  RPProjectPath  RPRReqTypeKey  RPAAttrKey  HelpFileName  CQHelpContextID  RPHelpContextID</p> <p><b>Fields added to enabled record type:</b>  Requirements_List</p>
RequisitePro Supplement	<p>Provides additional support for integration with RequisitePro when you use the AnalystStudio, DevelopmentStudio, TestStudio, and Enterprise schemas.</p> <p><b>Note:</b> The latest version of the RequisitePro package must be installed before you install RequisiteProSupplement.</p>	Performs additional setup for the EnhancementRequest and Defect record types.	<p><b>Fields added to the Requirement record type:</b>  Defects_List  EnhancementsRequests_List</p>

<b>Package</b>	<b>Description</b>	<b>Record type(s) added/modified</b>	<b>Field(s)</b>
Resolution	<p>Adds support so users can track how a record was resolved.</p> <p>Requires you to map schema states to the following state types: Not_Resolved; Resolved.</p>	Adds a Resolution tab to the enabled record type.	<p><b>Fields added to enabled record type:</b></p> <p>Resolution Resolution_statetype (read-only)</p>
TeamTest (read-only)	Provides support for integration with Rational TeamTest.	Adds Test Data and Environment tabs to the record type you specify.	<p><b>Fields added to enabled record type:</b></p> <p>Build Company Computer Contact Custom1 (modifiable) Custom2 (modifiable) Custom3 (modifiable) Fixed_In_Build Hardware Log Log_Folder old_internal_id Operating_System Other_Environment Resolution_Description Requirement Requirement_ID Test_Script Verification_Point</p>
UCMPolicyScripts	Provides support for the UnifiedChangeManagement (UCM) package by adding three global scripts.	Does not add any record types.	

Package	Description	Record type(s) added/modified	Field(s)
UnifiedChange Management (UCM) (read-only)	<p>Provides supports for the UCM process by enabling integration with Rational ClearCase 4.0 and above; links a ClearCase Project VOB with a ClearQuest user database. Requires the UCMPocScripts package. Can be used with the BaseCMAActivity package.</p> <p>Requires you to map schema states to the following state types: Waiting; Active; Ready; Complete.</p>	<p>Adds UCMUtilityActivity</p> <p>Adds UCM_Project stateless record type</p> <p>Also adds queries to the client workspace</p>	<p><b>Fields included in UCMUtilityActivity record type:</b> Description Owner Headline</p> <p><b>Fields included in UCM_Project record type:</b> name ucm_vob_object ucm_chk_before_deliver ucm_chk_before_work_on ucm_cq_act_after_deliver</p> <p><b>Fields added to enabled record type:</b> ucm_statetype ucm_vob_object ucm_stream_object ucm_stream ucm_view ucm_project</p>
Visual SourceSafe (read-only)	<p>Provides support for integration with Microsoft's VisualSourceSafe.</p>	<p>Adds SSOBJect and SCSnapObject stateless record types.</p> <p>Adds the VisualSourceSafe tab to the form of the enabled record type.</p>	<p><b>Fields added to enabled record type:</b> VSSChangeSet</p> <p><b>Fields added in SSOBJect record type:</b> CQDefects VSSCheckOutState VSSFileName VSSSpec VSSUser VSSVersion</p> <p><b>Fields added in SCSnapObject record type:</b> CreatedBy CreatedOn Label SnapElements</p>

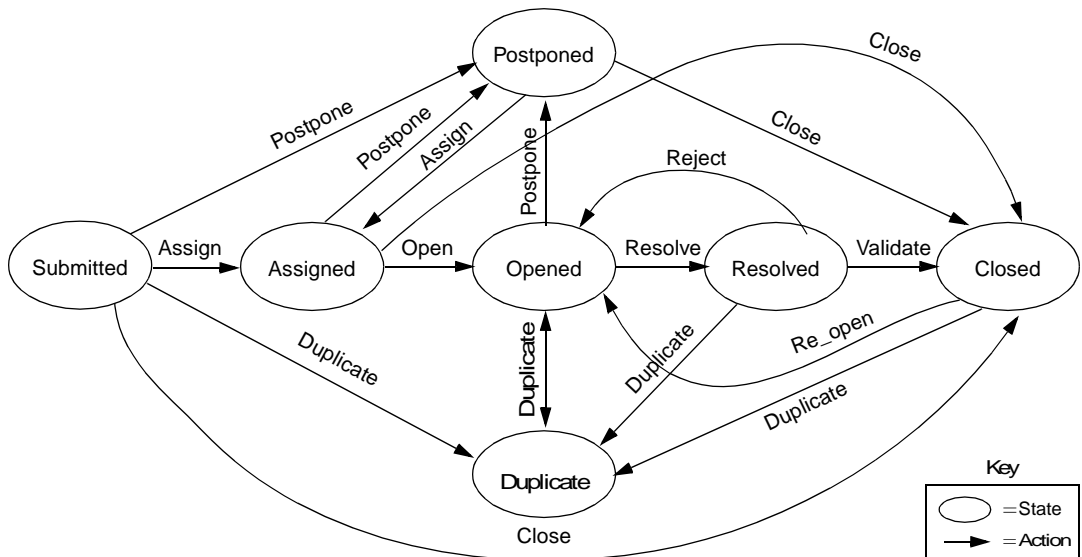


## State model for the Defect record type

The Defect record type contains the following states:

State	Description
Submitted	First state of a new defect
Assigned	Assigned to a team member
Opened	Being worked on
Resolved	Has been fixed
Closed	Has been verified
Duplicate	Duplicates another defect
Postponed	Postponed until another release or iteration

The state model for the Defect record type is the same for each predefined schema. You can also see this information in the Defect record type's State Transition Matrix.



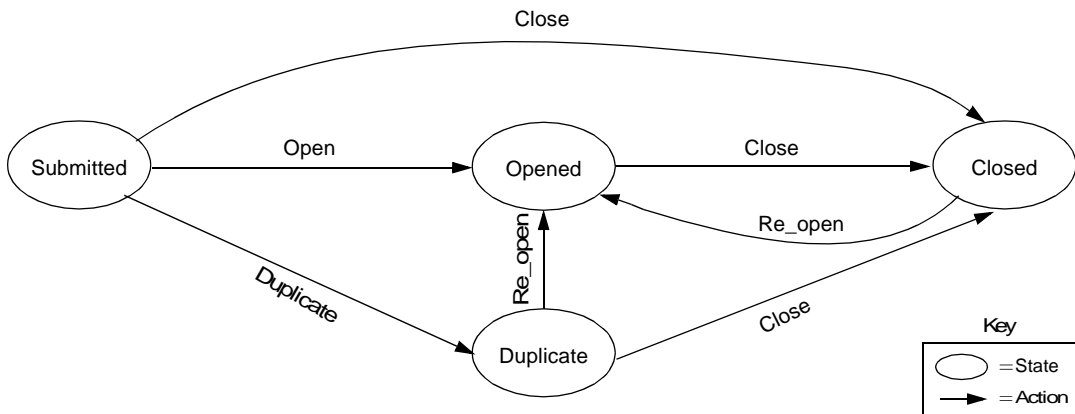
All Duplicate actions can be unduplicated by using the Unduplicate action.

## State model for the EnhancementRequest record type

The EnhancementRequest record type contains the following states:

State	Description
Submitted	First state of a new enhancement request
Opened	Being worked on
Closed	Has been verified
Duplicate	Duplicates another enhancement request

The following state model diagram shows how the EnhancementRequest record type moves from one state to another as the result of an action. You can also see this information in the EnhancementRequest record type's State Transition Matrix.



## State-type models for packages

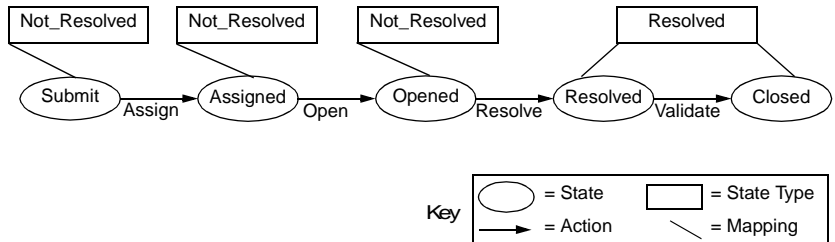
A state type is a label that defines a state's role in your state model. Some ClearQuest schemas have packages that require that each state in your state model be assigned or mapped to a specific state type. For example, the UnifiedChangeManagement schema and package use state types to invoke certain ClearCase triggers. ClearQuest's Resolution package uses state types to invoke certain hooks when a record moves to a state mapped to the Resolved state type.

**Note:** When you add a new state to a schema that uses state types, you must map the new state to a state type. See "Mapping state types" on page 82.

### Resolution package state type model

The following table and diagram show the state types and an example of a valid state mapping for the Resolution package.

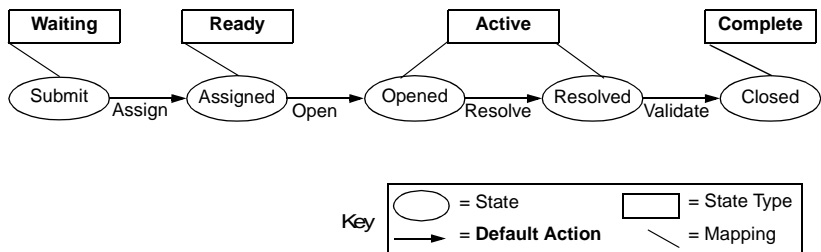
State type	Description
Not_Resolved	The record is not resolved.
Resolved	The record is resolved.



## UnifiedChangeManagement package state type model

The following table and diagram show the state types and a valid state mapping for the UCM package. The default actions of the states must provide a complete transition through the state type model.

State type	Description	Mapped state in UCM schema
Waiting	Record not yet assigned, triaged, or scheduled	Submitted Postponed
Ready	Record appears in the assigned user's To Do List query	Assigned
Active	User has started working on the record, which can now contain ClearCase element information	Opened
Complete	User has completed work on the record, which is now associated with a ClearCase project and its ClearCase elements	Resolved Closed Duplicate



**More information?** See Appendix B, “Enabling ClearQuest for Unified Change Management” .

# B

## Enabling ClearQuest for Unified Change Management

The ClearQuest/Unified Change Management (UCM) integration links ClearCase UCM project activities to ClearQuest records. The integration requires the following:

- A ClearQuest schema enabled for UCM
- ClearCase 4.x with a project enabled to work with ClearQuest

ClearQuest provides two predefined schemas that support UCM: the UnifiedChangeManagement schema and the Enterprise schema. The easiest way to implement UCM is to use one of these schemas.

You can also add UCM integration to an existing schema by installing the following packages into the schema:

- UCMPolicyScripts
- UnifiedChangeManagement

UCM packages must be added in the order shown above. Follow the instructions in “Adding UCM functionality to a schema” on page 2. For a complete description of these packages, see “ClearQuest packages” on page 4.

**More information?** For complete information on setting up and using the UCM integration, see the *Managing Software Projects with ClearCase* guide provided with ClearCase.

## Adding UCM functionality to a schema

This section describes how to add UCM functionality to an existing schema by installing the following UCM packages:

- UCMPolicyScripts package
- UnifiedChangeManagement package

These UCM packages must be installed in the order shown above. In addition, the UnifiedChangeManagement package requires that you map states and add default actions to your schema.

### Installing the UCM packages

To add the UCM packages to a schema:

- 1 In ClearQuest Designer, select **Package > Package Wizard**.
- 2 Select the UCMPolicyScripts package and click **Next**. (If the UCMPolicyScripts package is not listed, click **More Packages** and select the latest version to first install it into the schema repository.)
- 3 Select the schema to apply the UCMPolicyScripts package to and click **Finish**.

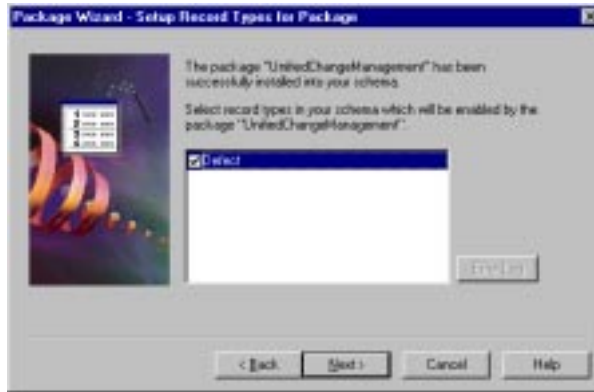
The Package Wizard checks out the schema and installs the UCMPolicyScripts package.

- 4 Select **File > Check In** to check in the schema.
- 5 Next, install the UnifiedChangeManagement package. Select **Package > Package Wizard**.
- 6 Select the UnifiedChangeManagement package and click **Next**. (If the UnifiedChangeManagement package is not listed, click **More Packages** and select the latest version to first install it into the schema repository.)
- 7 Select the schema to apply the UnifiedChangeManagement package to and click **Next**.

The Package Wizard checks out the schema again and installs the UnifiedChangeManagement package.

- 8 Select the record type(s) to enable for UCM and click Next to display the Setup State Types dialog.

**Note:** If you are enabling a record type with UCM packages, a separate submit form is required.



- 9 In the Setup State Types dialog, map the states in your schema to the UCM state types. Click in the State Types field for each state and select the appropriate UCM state type.

Select each record type you enabled for UCM and map its state types

Click the State Types field for each state to display the list of available state types



- 10 Repeat the state type mapping for each record type you enabled in step 8. When you're done, click Finish.

You must now assign a default action for each state in each of the record types you enable for UCM. See the next section.

**More information?** See “UnifiedChangeManagement package state type model” on page 14 of this guide and the *Managing Software Projects with ClearCase* guide provided with ClearCase.

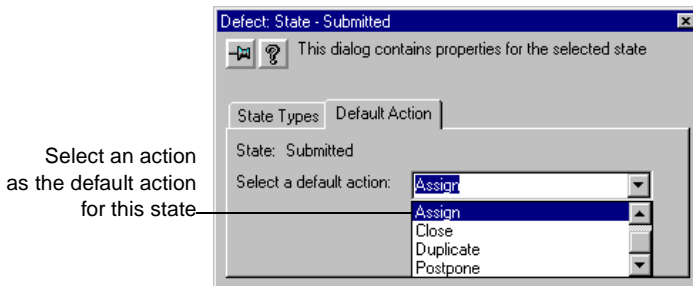
## Setting default actions for the UCM package

The State Transition Matrix in your schema must provide at least one path through the state type model for the UCM package, from the Waiting state type, to Ready, to Active, to Complete. (See “UnifiedChangeManagement package state type model” on page 14.)

For each state in your schema, except the state mapped to the UCM Complete state, you must assign a default action that moves the record from that state to the next state type in the UCM state type model.

To assign default actions:

- 1 Open the State Transition Matrix for the record type you enabled for UCM.
- 2 Right-click a state and select **Properties** to open the Properties dialog for that state.
- 3 In the Default Action tab of the Properties dialog, select a default action for the state. The Default Action tab lists the actions you have created for your state transitions in the State Transition Matrix.



For each state, select the action that moves the record to a state that is mapped to the next state type in the UCM model. For



example, the Submitted state (Waiting) moves to the Assigned state (Ready) through the Assign default action. If your schema has a Closed state and it is mapped to the Complete state type, it does not need a default action.

**More information?** See “Using default actions” on page 89 and “UnifiedChangeManagement package state type model” on page 14. Look up *actions*, *default* in the ClearQuest Designer Help index.



# Index

## A

access control  
   and user privileges 99  
   hook for 109  
 action hooks  
   Access Control 120, 122  
   adding 88  
   Commit 121, 124  
   execution order of 122  
   for setting value of parent  
     record 138  
   Initialization 120, 122  
   Initialization example 137  
   Notification 121, 124  
   reusing with Base action type 84  
   validating fields 120  
   Validation 121, 123  
   *see also* hook examples  
 action types  
   Base 84  
   Change\_state 84  
   Delete 84  
   Duplicate 84  
   Import 85  
   Modify 85  
   Record\_script\_alias 85  
   Submit 85  
   Unduplicate 85  
 actions  
   adding new 86  
   and state model 80  
   and state transition matrix 81  
   creating state transition 87  
   default 89  
   default for UCM B-4  
   grid 86  
   modifying 86  
   overview 18  
   properties 86  
   restricting access to 109  
   *see also* action hooks, action types  
 Active User privileges 99

ActiveX form control 91  
 admin user ID 13, 97  
 administrator, ClearQuest  
   defined 13  
   overview of tasks 16  
 AnalystStudio schema 55, A-2  
 API, ClearQuest  
   common calls 129  
   overview 14, 127  
   records (entities) 128  
   sessions 127  
   using to build queries 127  
   writing external applications 126  
 architecture, ClearQuest 14  
 assigning records to inactive  
   users 104  
 Attachment form control 91  
 ATTACHMENT\_LIST data type 72,  
   163  
 attachments  
   import file format 165  
   importing 172  
 Attachments package A-4

## B

back reference fields 78  
 backups, database 26  
 Base action type 84  
 BaseCMActivity package A-4  
 behavior, *see* field behavior  
 Behaviors grid 73  
 Blank schema 50, 55, A-2  
 BuildEntity method 128

## C

Change\_state action type 84, 86, 87  
 charts, using in ClearQuest  
   Web 144  
 Check Box form control 91  
 child/parent, *see* parent/child

- choice lists
    - and inactive users 104
    - field hook 114
    - hook example 135
    - limiting value options 117, 119
    - reinitializing 119, 147
  - ClearCase package A-4
  - ClearQuest
    - architecture 14
    - client 15
    - documentation 9
    - Export Tool, *see* Export Tool, ClearQuest
    - getting users started 20
    - Import Tool, *see* Import Tool, ClearQuest
    - Maintenance Tool, *see* Maintenance Tool, ClearQuest
    - Web, *see* Web, ClearQuest
  - ClearQuest Designer 15
    - starting 13
    - tutorial 9
  - code
    - reusing with Base action type 84
    - see also* hooks, scripts
  - COM objects 113
  - Combo Box form control 91
  - Commit action hook 121
  - common fields 67
  - Common schema 55, A-2
  - compiling hooks 112
  - constant list field hook 117
    - selecting behavior for 119
  - constant value field hook
    - selecting behavior for 119
  - controls, form 91
    - ActiveX 91
    - Attachment 91
    - Check Box 91
    - Combo Box 91
    - Drop-down Combo Box 91
    - Drop-down List Box 91
    - Duplicate Box 91
    - Duplicate Dependent 91
    - editing properties 94
    - Group Box 91
    - History 91
    - List Box 91
    - List View 92
    - Option Button 92
    - Parent/child 92
    - Picture 92
    - Push Button 92
    - Static Text 92
    - Text Box 92
  - CQLOAD 56
  - Customer package A-5
- ## D
- data
    - backing up 26
    - defining 62
    - exporting from ClearQuest
      - database 15, 38
    - exporting from other systems 161
    - how stored 24
    - linking records to share 75
  - data import 157, 164
    - attachments 172
    - attachments file format 165
    - delimiter 161, 166
    - duplicate records 168, 173
    - empty fields 162
    - error file 167
    - existing fields 168
    - existing records 174
    - exporting from other systems 161
    - file format 161
    - from the error file 174
    - history file format 164
    - history information 171
    - mapping fields 169
    - original (old) record ID 159
    - overview of process 158
    - performing the import 166
    - reference fields 162
    - supported data types 163
    - unsupported data types 163
  - data types
    - field 72
    - fields in import file 163
    - mapping unsupported 163
  - database properties, viewing 37
  - databases
    - accessing with API 127
    - backing up 26
    - changing vendors 29
    - choosing a vendor 25
    - creating new 26
    - location for 24
    - maintenance overview 27
    - master, schema repository 17
    - moving 29

- preserving integrity 24
- required 17, 24
- supported vendors 25
- test database 47
- see also* user databases, schema repository
- date format in import file 164
- DATE\_TIME data type 72, 163
- DB Column Name 71, 74
- DB Name 65
- DB Subscriptions 98
- default
  - actions for states 89
  - actions for UCM 82, B-4
  - field behavior 74
  - record type 66
  - user ID 13, 97
- Default Value field hook 114
- Defect record type state model A-11
- DefectTracking schema 55, A-2
- Delete action type 84
- delimiters for import files 161, 166
- dependent fields
  - creating 115
  - enabling for web client 116, 146
- dependent list hook example 133
- destination state 87
- DevelopmentStudio schema 55, A-2
- diagram, *see* state model and state type model
- dialog tabs, adding to forms 94
- display\_name field 35, 164, 165
- documentation, ClearQuest 9
- double quotes in imported records 162
- Drop-down Combo Box form
  - control 91
- Drop-down List Box form control 91
- Duplicate action type 84
- Duplicate Box form control 91
- Duplicate Dependent form
  - control 91
- duplicate records, importing 168, 173
- dynamic list field hook 117, 118
  - selecting behavior for 119

## E

- e-mail
  - configuring ClearQuest users 151

- Email\_Rule record type 150
  - fields to include 156
  - format example 155
  - format for submitting 154
  - format guidelines 155
  - multi-line fields 155
  - notification 150
  - round-trip 156
  - submitting records by 153
  - triggering notification 151
- Email package A-6
- E-mail Reader, Rational 14, 15
  - configuring 153
- e-mail rules
  - access privileges 150
  - basing on a user group 105
  - creating 150
  - triggering notification 151
- Email\_Rule record type 150
- empty fields, importing 162
- EnhancementRequest package A-5
- EnhancementRequest record type
  - state model A-12
- Enterprise schema 55, A-2
- entity 128
- Entity object 127, 129
- errlog.txt 174
- error file 167, 174
  - importing from 174
- execution order of hooks 122
- Export Tool, ClearQuest 14, 15
  - using 38
- exporting
  - data from ClearQuest database 15
  - data from other systems 161
  - users and groups 110
- external applications
  - and user groups 105
  - writing 126

## F

- family, *see* record type families
- field behavior
  - default 74
  - for choice list fields 119
  - reset by hooks 113
  - supported 73
- field hooks
  - adding 70, 79

- Choice List 114, 123
  - Default Value 114, 122
  - Dynamic-List 117
  - execution order 122
  - Permission 114, 122
  - Validation 114, 123
  - Value Changed 114, 122
    - see also* hook examples
  - fields
    - adding to a form 93
    - adding to fields grid 70
    - back reference 78
    - changing name of 74
    - common 67
    - data type, selecting 71
    - data types for import file 163
    - data types, changing 72
    - data types, supported 72
    - DB Column Name 71, 74
    - default behavior 74
    - defining behavior 63, 73
    - deleting 74
    - dependent 115
    - dependent for web client 116
    - editing properties 94
    - e-mail format for multi-line 155
    - Help text for 71
    - initializing values 137
    - linking records in 75, 77
    - modified by packages A-4
    - modifying 70
    - system fields 70, 74
    - Visible in Query 71
      - see also* field behavior, field hooks, and controls, form
  - file format
    - for importing 161
    - for importing attachments 165
    - for importing history 164
  - Finding hook script text 131
  - form controls, *see* controls, form
  - forms 90
    - adding fields 93
    - controls 91
    - creating 63, 90
    - deleting 95
    - editing 94
    - exporting 95
    - for multiple platforms 96
    - importing 95
    - reusing 95
    - submit 90
    - using two forms 90
- G**
- GetSession method 127
  - global scripts 112, 126
  - Group Box form control 91
  - groups, user
    - adding 100
    - controlling access to actions 109
    - importing and exporting 110
    - subscribing to databases 98, 107
- H**
- help text, adding to fields 71
  - history
    - import file format 164
    - importing 171
  - History form control 91
  - History package A-6
  - hook examples 133
    - choice list 135
    - dependent list 133
    - hook for initializing field value 137
    - hook to set value of parent record 138
  - hooks 112
    - and COM objects 113
    - and global scripts 126
    - and Super User privileges 113
    - and user groups 105
    - compiling 112
    - editing script text 131
    - execution order of 122
    - finding script text 131
    - for actions 88, 120
    - for detecting a web session 147
    - for fields 79, 114
    - in the web client 146
    - overview 112
    - planning 63, 113
    - resetting read-only field value 113
    - reusing action hooks 84
    - scripting language for 51
    - see also* hook examples, action hooks, field hooks

## I

- ID, user
  - admin default 13, 97
  - Windows user profiles 97
- Import action type 85
- import file
  - creating 161
  - field data types supported 163
  - format 161, 162
- Import Tool, ClearQuest 14, 15
- importing
  - attachments 172
  - attachments file format 165
  - creating an import file 161
  - data from another system 166
  - data into ClearQuest 157
  - date formats 164
  - delimiter 161, 166
  - duplicate records 168, 173
  - embedded double quotes 162
  - empty fields 162
  - error file 167
  - existing fields 168
  - file format 161
  - from the error file 174
  - history 171
  - history file format 164
  - mapping fields 169
  - original (old) record ID 159
  - overview of process 158
  - performing the import 166
  - reference fields 162
  - upgrading existing records 174
  - users and groups 110
- inactive
  - groups 108
  - users 104
- Initialization action hook 120, 137
- INT data type 72, 163

## K

- key, unique 65

## L

- languages, scripting 51
- Limit to List 119
- List Box form control 91
- List View form control 92

## lists

- and inactive users 104
  - choice 117, 119
  - dynamic 114
  - hook for dependent 133
- logging in
  - and inactive users 104
  - default admin ID 13

## M

- mailreader.exe 153
- Maintenance Tool, ClearQuest 14, 15
- maintenance, database 27
- mandatory field behavior 73
- mapping
  - fields for importing 169
  - state types 82
  - state types for UCM B-3
- master database, *see* schema repository
- message boxes, in ClearQuest
  - Web 147
- metadata 17, 45
- method
  - BuildEntity 128
  - GetSession 127
- Microsoft Access database 25
- Microsoft SQL Server database 25
- model
  - state type for packages A-13
- model, *see* state model, state type model
- Modify action type 85
- moving user databases 34
- MULTILINE\_STRING data
  - type 72, 163
- multi-platform forms 96

## N

- name field, changing 74
- Notes package A-6
- notification 150
  - action hook 121
  - hook for 112
  - setting up e-mail rules 150
  - triggering 151

## O

- object
  - Entity 127, 129
  - QueryDef 127
  - ResultSet 127
  - session 127, 129
- Option Button form control 92
- optional field behavior 73
- Oracle database 25
- original (old) record ID 159

## P

- Package Wizard 57
- packages
  - adding to schema 56
  - adding UCM B-2
  - Attachments A-4
  - BaseCMActivity A-4
  - ClearCase A-4
  - Customer A-5
  - Email A-6
  - Enhancement Request A-5
  - History A-6
  - list of predefined A-4
  - mapping state types for 82
  - Notes A-6
  - Package Wizard 57
  - PQC A-6
  - Project A-7
  - Repository A-7
  - RequisitePro A-8
  - RequisiteProSupplement A-8
  - Resolution A-9
  - TeamTest A-9
  - UCM A-10
  - VisualSourceSafe A-10
- parent/child
  - form control 92
  - hierarchy 77
  - hook to set value of parent record 138
  - record linking 77
- Perl scripting language 51
  - and COM objects 113
- Permission field hook 114
- permissions, *see* privileges
- Picture form control 92
- PQC package A-6
- privileges

- Active User 99
  - for schema design 45, 61
  - Schema Designer 99
  - Super User 99
  - system 103
  - types of 99
  - User Administrator 99
- Project package A-7
- properties
  - action 86
  - field 94
  - form control 94
  - setting for schemas 51
  - state types and packages 82
- Push Button form control 92

## Q

- query
  - field Visible in Query 71
  - multiple record types 67
  - using API to build 127
- QueryDef object 127

## R

- Rational E-mail Reader 14, 15
- Rational Software web site 11
- readme, ClearQuest 25
- read-only field behavior 73
  - values reset by hooks 113
- Recalculate Choice List 119, 147
- record forms, *see* forms
- record scripts 125
- record type families 64
  - common fields 67
  - creating 67
  - deleting 69
  - naming conventions 67
  - renaming 69
- record types 18, 64
  - adding with packages 65
  - creating 65, 67
  - DB Name 65
  - default 66
  - Defect A-11
  - deleting 69
  - Email\_rule 150
  - enabling for UCM B-3
  - EnhancementRequest A-12
  - fields 70



- forms 90
- querying across multiple 64
- renaming 69
- selecting default 66
- state-based 64
- stateless 64
- system 64
- see also* record type families
- Record\_script\_alias action type 85
- records
  - assigning to inactive users 104
  - e-mail format 154, 155
  - entity 128
  - importing duplicate 173
  - linking 75, 77
  - parent/child linking 77
  - submitting by e-mail 153
  - upgrading existing 174
- REFERENCE data type 72, 163
- reference fields, importing 162
- REFERENCE\_LIST data type 72, 163
- release notes, ClearQuest 25
- remote access, *see* Web, ClearQuest
- reports on ClearQuest Web 144
- Repository package A-7
- RequisitePro package A-8
- Resolution package
  - description A-9
  - state type model A-13
- ResultSet object 127
- round-trip e-mail 156

**S**

- SAMPL database, connecting to 27
- Schema Designer privileges 99
- schema properties 51
- schema repository
  - changing database vendor 30
  - connecting to 15, 27
  - creating new 26, 27
  - defined 17
  - importing schemas into with
    - CQLOAD 56
  - metadata 17
  - moving 29, 30
- schemas
  - adding packages to 56
  - adding UCM packages B-2
  - changing 34
  - checking in 53
  - checking out 49
  - considerations for moving 35
  - creating new 50
  - customizing 61
  - customizing overview 62
  - default record type for 66
  - deleting 54
  - for Rational suites A-1
  - how to work with 46
  - importing/exporting with
    - CQLOAD 56
  - overview 17
  - packages 55, 56
  - packages in A-4
  - predefined 55, A-2
  - privileges required to
    - customize 45, 61
  - saving 46
  - selecting 55
  - setting properties 51
  - testing 47
  - undo check out 53
  - upgrading user database with 54
  - validating 52, 53
  - version number 53
- scripts
  - access control 109
  - global 112, 126
  - record 125
  - selecting language for 51
- session object 127, 129
- Set Test Database 47
- SHORT\_STRING data type 72, 163
- source state 87
- SQL Anywhere database 25
- SQL Server database 25
- STATE data type 163
- state model
  - defining 18, 62, 80
  - for Defect record type A-11
  - for EnhancementRequest record type A-12
- state transition
  - creating 87
  - matrix 81
- state type model
  - for packages A-13
  - Resolution package A-13
  - UCM package A-14
- state types

- mapping 82
- mapping for UCM B-3
- Resolution package A-13
- see also* state type model
- state-based records 64
- stateless records 64
- unique key 65
- states 18, 80
  - action 81
  - adding 82
  - creating transitions 87
  - default actions for 89
  - destination 81, 87
  - renaming 83
  - source 81, 87
  - see also* state types, state transition
- Static Text form control 92
- Submit action type 85
- submit form 90
- Super User privileges 99
  - and hooks 113
- support, technical 11
- Sybase SQL Anywhere database 25
- system
  - fields 70, 74
  - record types 64

## T

- tabs
  - adding to forms 94
- TeamTest package A-9
- technical support 11
- test database 47
- TestStudio schema 55, A-2
- Text Box form control 92
- text, Help 71
- tutorial, ClearQuest Designer 9

## U

- Unduplicate action type 85
- Unified Change Management
  - adding UCM to a schema B-2
  - and BaseCMActivity A-4
  - default actions for B-4
  - enabling record types for B-3
  - mapping state types B-3
  - package A-10
  - package installation order B-2

- required packages B-1
- UCM-enabled schemas B-1
- UnifiedChangeManagement
  - schema 55, A-3
- unique key, stateless record type 65
- UNIX
  - ClearQuest support for 15
  - scripting language for 51
- updating user databases 36
- Upgrade user DB 98
- Use\_hook field behavior 73
- User Administration dialog 98, 100
- User Administrator privileges 99
- user databases 17
  - adding user information to 54, 98
  - associating with a schema 54
  - changing database vendor 29, 31
  - creating 26
- DB Subscriptions 98
  - defined 24
  - deleting 42
  - moving 29, 31, 34
  - subscribing users to 102
  - switching schemas 34
  - undeleting 43
  - updating 36
  - upgrading with new schema
    - version 54
  - upgrading with user
    - information 101
- Visible to the designer only 47
- working with test database 47
- user groups
  - adding users 106
  - and external applications 105
  - and hooks 105
  - creating 105
  - displaying members 106
  - importing and exporting 110
  - making inactive 108
  - privileges for administering 97
  - removing users 106
  - restricting access to actions 109
  - subscribing to databases 98, 107
  - upgrading database with user
    - group information 101
- user ID
  - admin default 13, 97
- user privileges 99
- users
  - adding new 100

- adding to groups 106
- administering 97
- exporting and importing 110
- making inactive 104
- modifying profile 103
- privileges 99
- privileges required for administering 97
- removing from user group 106
- restricting access to actions 109
- restricting access to ClearQuest Web 145
- subscribing to databases 98, 102
- upgrading database with user information 98, 101
- Windows accounts 97

- using hooks on 146
- using reports on 144

Windows

- ClearQuest for 15
- user profiles 97

workflow, defining in ClearQuest 18

## V

validation

- action hook 112, 121
- field hook 112, 114
- of schema 52, 53

Value Changed field hook 114

VBScript

- and COM objects 113
- for Web hooks 146
- selecting as scripting language 51

viewing database properties 37

Visible in Query 71

Visible to the designer only, test database 47

VisualSourceSafe package A-10

## W

web session

- detecting 147
- Perl example 148
- VBScript example 148

web site, Rational Software 11

Web, ClearQuest 15

- customizing 144
- dependent fields for 116, 146
- displaying messages on 147
- hook for detecting a web session 147
- limiting access (web entry) 145
- navigating back and forward 144
- server 14
- using charts on 144

