

Rational ClearQuest®

Administrator's Guide

VERSION: 2002.05.00

PART NUMBER: 800-025121-000

WINDOWS/UNIX

IMPORTANT NOTICE

COPYRIGHT

Copyright ©1977-2001, Rational Software Corporation. All rights reserved.

Part Number: 800-025121-000

Version Number: 2002.05.00

PERMITTED USAGE

THIS DOCUMENT CONTAINS PROPRIETARY INFORMATION WHICH IS THE PROPERTY OF RATIONAL SOFTWARE CORPORATION ("RATIONAL") AND IS FURNISHED FOR THE SOLE PURPOSE OF THE OPERATION AND THE MAINTENANCE OF PRODUCTS OF RATIONAL. NO PART OF THIS PUBLICATION IS TO BE USED FOR ANY OTHER PURPOSE, AND IS NOT TO BE REPRODUCED, COPIED, ADAPTED, DISCLOSED, DISTRIBUTED, TRANSMITTED, STORED IN A RETRIEVAL SYSTEM OR TRANSLATED INTO ANY HUMAN OR COMPUTER LANGUAGE, IN ANY FORM, BY ANY MEANS, IN WHOLE OR IN PART, WITHOUT THE PRIOR EXPRESS WRITTEN CONSENT OF RATIONAL.

TRADEMARKS

Rational, Rational Software Corporation, Rational the e-development company, ClearCase, ClearCase Attache, ClearCase MultiSite, ClearDDTS, ClearQuest, ClearQuest MultiSite, DDTS, Object Testing, PureCoverage, PureDDTS, PureLink, Purify, Purify'd, Quantify, Rational Rose, Rational Suite, Rational Visual Test, Requisite, RequisitePro, RUP, AnalystStudio, ClearGuide, ClearTrack, Connexis, Rational Suite AnalystStudio, Rational Suite ContentStudio, Rational Suite Enterprise, Rational Suite ManagerStudio, Rational Unified Process, TestStudio, among others, are either trademarks or registered trademarks of Rational Software Corporation in the United States and/or in other countries. All other names are used for identification purposes only, and are trademarks or registered trademarks of their respective companies.

Microsoft, the Microsoft logo, Active Accessibility, Developer Studio, Direct3D, FrontPage, J/Direct, JScript, the Microsoft eEmbedded Visual Tools logo, the Microsoft Internet Explorer logo, the Microsoft Office Compatible logo, Microsoft Press, the Microsoft Press logo, MS-DOS, MSDN, the Office logo, Outlook, PhotoDraw, PowerPoint, SourceSafe, Visual C++, Visual J++, Visual SourceSafe, Visual Studio, the Visual Studio logo, Win32, Win32s, Win64, Windows, the Windows CE logo, the Windows logo, Windows NT, the Windows Start logo, and XENIX are trademarks or registered trademarks of Microsoft Corporation in the United States and other countries.

The Sun J2EE Patterns are used with permission from the book "Core J2EE Patterns" by Deepak Alur, John Crupi, and Danny Malks, published by Sun Microsystems Press/Prentice Hall. Copyright 2001 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, CA 94303. All rights reserved. SUN PROVIDES EACH J2EE PATTERN "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

FLEXIm and GLOBEtrotter are trademarks or registered trademarks of GLOBEtrotter Software, Inc. Licensee shall not incorporate any GLOBEtrotter software (FLEXIm libraries and utilities) into any product or application the primary purpose of which is software license management.

Portions Copyright ©1992-2001, Summit Software Company. All rights reserved.

PATENT

U.S. Patent Nos. 5,193,180 and 5,335,344 and 5,535,329 and 5,835,701. Additional patents pending.

Purify is licensed under Sun Microsystems, Inc., U.S. Patent No. 5,404,499.

GOVERNMENT RIGHTS LEGEND

Use, duplication, or disclosure by the U.S. Government is subject to restrictions set forth in the applicable Rational Software Corporation license agreement and as provided in DFARS 277.7202-1(a) and 277.7202-3(a) (1995), DFARS 252.227-7013(c)(1)(ii) (Oct. 1988), FAR 12.212(a) (1995), FAR 52.227-19, or FAR 227-14, as applicable.

WARRANTY DISCLAIMER

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

Contents

Before you begin	xiii
Audience	xiii
ClearQuest documentation	xiii
ClearQuest documentation roadmap	xiv
Contacting Rational Technical Publications	xv
Contacting Rational Technical Support	xv
1 Understanding ClearQuest administration	1
Starting ClearQuest Designer	1
Getting around in ClearQuest Designer	2
Using toolbars, palettes, and menus	2
Using keyboard shortcut and mouse actions	3
Overview of ClearQuest architecture	5
How ClearQuest components work together	5
How you use ClearQuest components	6
Understanding ClearQuest schemas and databases	7
Designing your ClearQuest workflow	8
Understanding record types, states, and actions	8
Customizing your workflow	9
Before you begin customizing	9
Overview of administrator tasks	10
Getting ClearQuest users started	11
2 Managing databases	13
Working with ClearQuest databases	13
Backing up your data	13
Overview of database maintenance	14
Creating new databases	15
Creating a new schema repository	15
Creating a new user database	16
Connecting to ClearQuest	17
Modifying connections	17
Using connection profiles	19
Moving a database to a new location or to a new vendor	20
Preparing to move a ClearQuest database	20

Moving a schema repository	21
Updating schema repository properties	22
Moving a user database	23
Updating the properties of a user database	26
Deleting databases	27
Deleting a user database	27
Undeleting a user database	28
Viewing database properties	29
3 Working with ClearQuest schemas	31
Overview of schema procedures	32
Working with a test database	33
Creating a test database	33
Setting the test database for the schema	34
Testing your work	34
Working with schemas	35
Checking out a schema	35
Creating a new schema	37
Selecting a scripting language	38
Validating schema changes	39
Checking in a schema	40
Undoing a schema checkout	41
Applying schema changes to a user database	41
Saving work in progress	42
Deleting a schema or schema version	43
Selecting a ClearQuest schema	44
Changing a database to a different schema	45
Working with packages	46
Adding packages to a schema	47
Enabling record types	48
Setting up state types	49
4 Customizing a schema	51
Overview of customizing a schema	52
Working with record types	54
State-based record types	54
Stateless record types	54
Adding a new record type	55
Selecting a unique key for a stateless record type	56

Selecting a default record type	57
Working with record type families	57
Renaming a record type or family	60
Deleting a record type or family	60
Working with fields	61
Adding a new field	62
Defining field behavior	66
Modifying a field	68
Deleting a field	69
Using fields to link records	70
Customizing fields by adding hooks	74
Defining your state model	75
Using the State Transition Matrix	76
Adding a new state	77
Mapping state types	78
Changing the name of a state	79
Deleting a state	80
Working with actions and action types	81
Understanding actions and action types	81
Supported action types	83
Adding a new action	84
Creating a state transition	84
Modifying actions	85
Customizing actions by adding hooks	86
Using default actions	87
Deleting an action	88
5 Working with forms	89
Using the Forms window	90
Working with forms	91
Creating a new form	91
Changing the name of a form	92
Changing the size of a form	92
Changing the font on forms	92
Deleting a form	93
Adding tabs to a form	93
Changing the name of a tab	93
Restricting access to a tab	94
Changing the order of tabs	94

Deleting tabs	95
Copying tabs	95
Working with form controls	96
Adding controls to a form	98
Editing control properties	103
Deleting a control from a form	104
Changing the size and location of controls	104
Moving controls	105
Aligning controls	105
Resizing controls	106
Changing the tab order of controls	107
Creating forms for multiple platforms	108
Creating forms to view and modify imported data	108
Reusing forms	109
Exporting a form	109
Importing a form	110
Form controls reference	111
ActiveX control	111
Attachment control	112
Checkbox control	113
Combo box control	114
Drop-down list box control	116
Drop-down combo box control	118
Duplicate base control	120
Duplicate dependents control	121
Group box control	122
History control	122
List box control	123
List view control	125
Option button control	127
Parent/Child control	128
Picture control	129
Push button control	130
Static text box control	131
Text box control	132
6 Administering users	135
Overview of user administration	136
Viewing database subscriptions	137

ClearQuest user privileges	138
Working with users	139
Adding a new user	139
Assigning user access privileges	141
Subscribing users and groups to databases	142
Unsubscribing users and groups from databases	143
Applying schema changes to the user database	144
Editing users	145
Changing user privileges	145
Editing a user profile from the ClearQuest client	146
Making users and groups inactive	147
Working with user groups	148
Creating a new user group	148
Creating subgroups	149
Adding users to a group	149
Removing users from a group	150
Subscribing user groups to databases	150
Restricting user access to actions	151
Exporting and importing users and user groups	152
Administering users in a MultiSite environment	153
How mastership affects ClearQuest client users	153
How mastership affects user administration	154
Finding out where users and groups are currently mastered	154
Changing the mastership of a user	155
7 Using security in ClearQuest	157
Hiding records in ClearQuest	158
Designing a security system	161
Security example	162
Checking out a schema	162
Creating a security context field	163
Adding the security context field to the form	165
Applying the schema changes	165
Creating the user groups	166
Submitting the security context records	166
Associating groups with each security context record	167
Editing records to grant privileges to groups	168
Using ClearQuest's other security features	169
Restricting access to fields	169

Restricting access to actions	169
Restrict access to dialog tabs	169
Adding password protection	169
8 Using hooks to customize your workflow	171
Understanding hooks	172
Important hook considerations	172
Writing scripts	173
Operating context for using scripts	174
Working with field hooks	175
Adding a field hook	176
Editing a field hook	177
Deleting a field hook	177
Creating a dependency between fields	178
Creating a choice list for a field	180
Creating a dynamic choice list	181
Defining choice list behavior	183
Specifying a default value for a field	184
Validating user input in a field	185
Working with action hooks	187
Adding an action hook	189
Editing an action hook	190
Deleting an action hook	190
Execution order of field and action hooks	191
When an action begins	191
When a field value is set	192
When the record is validated	192
When the record is committed	192
Working with record scripts	194
Understanding record scripts	194
Using record scripts on ClearQuest Web	195
Form control events	196
Adding a record script to a record type	197
Editing a record script	198
Deleting a record script	198
Working with global scripts	199
Understanding global scripts	199
Creating a global script	200
Editing a global script	200

Deleting a global script.	201
Writing external applications.	201
Using the ClearQuest API	202
Working with sessions	202
Working with queries	202
Working with records	203
Common API calls	204
Finding text in hook scripts	206
Examples of common hooks	207
Hook for creating a dependent list	207
Field choice list hook to display user information.	208
Action initialization hook for a field value	211
Action hook for setting the value of a parent record.	212
9 Administering ClearQuest Web.	217
ClearQuest Web considerations	217
Customizing ClearQuest Web	218
Limiting access to ClearQuest Web	218
Using hooks in ClearQuest Web.	219
Displaying messages on ClearQuest Web.	220
Using hooks to detect a web session.	221
10 Administering ClearQuest E-mail	223
Enabling e-mail notification.	224
Setting up e-mail rules	224
Configuring ClearQuest clients to send e-mail.	226
Enabling e-mail submission	227
Configuring Rational E-mail Reader.	227
Formatting e-mail for submission	228
Using "round-trip" e-mail.	230
Creating e-mail actions.	231
E-mail notification hook example	232
11 Importing and exporting with ClearQuest.	237
Preparing for a successful data import.	238
Creating an import schema	238
Creating a database for imported data.	240
Testing the import process	240
Creating a ClearQuest import file	241

Formatting the record import files	241
Formatting the history import file	244
Formatting the attachments import file	245
Performing the data import	246
Determining what records to import	246
Using the ClearQuest Import Tool	246
Importing duplicate records	250
Importing records from the discarded data log	251
Updating existing records	251
Using the ClearQuest Export Tool	252
A ClearQuest schemas and packages	257
ClearQuest predefined schemas	258
ClearQuest packages	259
State model for the Defect record type	266
State model for the EnhancementRequest record type	267
State-type models for packages	268
Resolution package state type model	268
UnifiedChangeManagement package state type model	269
B Adding ClearQuest integrations	271
Overview of ClearQuest integrations	272
Independent integrations	272
Dependent integrations	273
Enabling record types for integrations	274
Viewing the packages in your schema	275
Testing integrations	276
Adding independent integrations	276
Adding dependent integrations	278
Adding a Rational Administrator integration	279
Adding a Rational ClearQuest Project Tracker integration	280
Adding a Rational TeamTest integration	282
Adding a Rational UCM integration	284
Adding a Microsoft Visual SourceSafe integration	290
Index	293

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 know how to use the ClearQuest client, have experience administering relational databases, and know how to write scripts in VBScript or Perl.

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.

ClearQuest documentation

ClearQuest includes the following printed documentation:

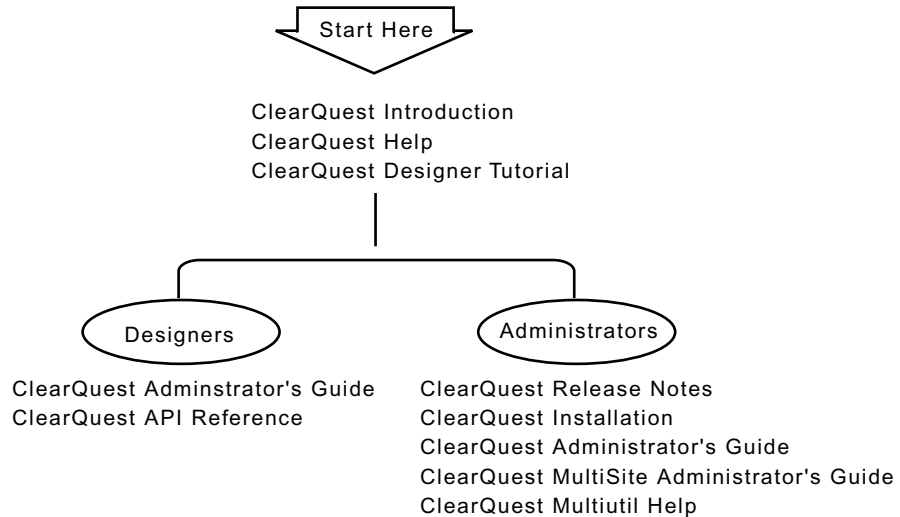
- *Introducing Rational ClearQuest*: Provides an overview of how to use ClearQuest, and a brief example of how you can customize ClearQuest to fit your organization's workflow.
- *Installation Guide, Rational ClearQuest*: Explains how to install ClearQuest, ClearQuest Designer, vendor databases, and related tools.
- *Administrator's Guide, Rational ClearQuest*: Explains how to administer ClearQuest databases, customize how you use ClearQuest client, and administer users and user groups.

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.
- *Rational ClearQuest Microsoft Project Tracker Guide*: Online user guide providing instructions on installing and using the Rational ClearQuest Microsoft Project Tracker.

ClearQuest documentation roadmap



Contacting Rational Technical Publications

To send feedback about documentation for Rational products, please send e-mail to our technical publications department at techpubs@rational.com.

Contacting Rational Technical Support

For technical information about ClearQuest Designer, 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>. You can also contact Rational Technical Support as follows:

Location	Telephone	Facsimile	E-mail
North America	(800) 433-5444 (toll free) (408) 863-4000 Cupertino, CA	(781) 676-2460 Lexington, MA	support@rational.com
Europe, Middle East, Africa	+31 (0) 20-4546-200 Netherlands	+31 (0) 20-4545-202 Netherlands	support@europe.rational.com
Asia Pacific	+61-2-9419-0111 Australia	+61-2-9419-0123 Australia	support@apac.rational.com

Note: When you contact Rational Technical Support, please be prepared to supply the following information:

- Your name, telephone number, and company name
- Your computer's make and model
- Your operating system and version number
- Product release number and serial number
- Your case ID number (if you are following up on a previously-reported problem)

Understanding ClearQuest administration

1

This chapter provides essential concepts to help you administer ClearQuest, and to use this guide effectively. The topics covered include:

- Starting ClearQuest Designer
- Getting around in ClearQuest Designer
- Overview of ClearQuest architecture
- Understanding ClearQuest schemas and databases
- Designing your ClearQuest workflow
- Getting ClearQuest users started
- Understanding ClearQuest schemas and databases

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 138.

Starting ClearQuest Designer

To start 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 “Editing users” on page 145.

Getting around in ClearQuest Designer

Using toolbars, palettes, and menus

Using toolbars

ClearQuest Designer has two toolbars:

- The Designer toolbar includes icons for some of the more commonly used menu options, such as opening a schema, checking schemas in and out, and upgrading a database.
- The Form Layout toolbar includes icons for aligning controls on a form.

Using palettes

ClearQuest Designer provides two palettes for creating controls and adding them to forms:

- Control palette: Use to add specific controls to the form.
- Field List palette: Use to add specific fields (along with a default control) to the form.

Using menus

Most ClearQuest Designer features and commands are available from the program's pull-down menus. Menu options are dynamic and change as you use different ClearQuest Designer features. For example, when viewing a form, the **Edit** and **View** menus change and new menus appear that apply to the form.

Using the shortcut menu

Right-click a cell in a grid, a form control, an error message in the Validation Output window, or an item in the Workspace to display a shortcut menu for the clicked item. Some menu options, as well as time-saving features, are included only on the shortcut menu. For example, setting the default record type can only be done from the record type's shortcut menu.

To get help about a particular item, right-click the item and select **What's This?** from the shortcut menu.

Using keyboard shortcut and mouse actions

ClearQuest Designer includes the following keyboard shortcuts and mouse actions.

File menu	
Ctrl + O	Displays the Open Schema wizard
Ctrl + K	Displays the Check Out dialog box
Ctrl + I	Validates the schema and displays the Check In dialog box
Ctrl + U	Undoes the schema checkout and removes it from ClearQuest Designer
Ctrl + D	Validates schema changes
Ctrl + P	Displays the Print dialog box
Edit menu	
Ctrl + X	Deletes the selected item
Ctrl + C	Copies the selected item from the active window and places it onto the Clipboard
Ctrl + V	Pastes the last item placed on the clipboard into the active window
Ctrl + R	Displays the Properties dialog box for the selected item
Del	Deletes the selected item
Ctrl + Z	Undoes previous action
Form Layout menu	
F7	Sets the size of a control based on its content
Ctrl + Left arrow	Aligns the left side of selected controls
Ctrl + Right arrow	Aligns the right side of selected controls
Ctrl + Up arrow	Aligns the tops of selected controls
Ctrl + Down arrow	Aligns the bottoms of selected controls
Ctrl + Shift + Right	Distributes horizontal distance evenly between selected controls
Ctrl + Shift + Down	Distributes vertical distance evenly between selected controls
F9	Aligns the vertical centers of selected controls
Shift +F9	Aligns the horizontal centers of selected controls
Ctrl + E	Sets the height and width of selected controls to the same size

Keyboard accelerators

F1	Displays Help for the selected item or active window
Right arrow	Expands the selected item
Left arrow	Collapses the selected item
Keypad *	Expands all branches for the selected item
Keypad +	Expands the selected item
Keypad -	Collapses the selected item
Down arrow	Moves to the next row in a grid or the next item in the Workspace
Up arrow	Moves to the previous row in a grid or the previous item in the Workspace
Left arrow	Moves one cell to the left in a grid
Right arrow	Moves one cell to the right in a grid
Ctrl + click	Select non-adjacent items, such as cells in a grid

Mouse actions

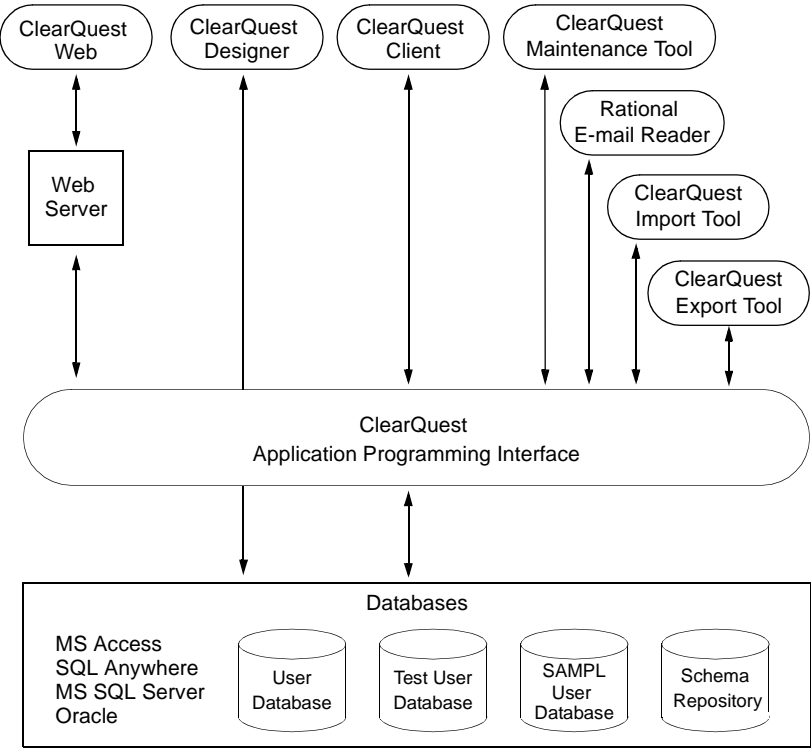
- Double-click a cell in a grid or a form control to display the Properties dialog box, if available.
 - Drag and drop a field from the Field list dialog box to a form control.
 - Right-click a cell in a grid, a form control, an error message in the Validation Output window, or an item in the Workspace to display a shortcut menu for the clicked item.
-

Overview of ClearQuest architecture

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

How ClearQuest components work together

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



How you use ClearQuest components

Here's how you use ClearQuest components:

Component	Used by	Use to
Client tools		
ClearQuest for Windows	Everyone	Submit, modify, and track defect and change requests, and analyze project progress by running queries, charts, and reports.
ClearQuest for UNIX	Everyone	Submit, modify, and track defect and change requests, and analyze project progress by running queries.
ClearQuest Web	Everyone	Access ClearQuest across multiple platforms through Netscape Navigator® or Microsoft Internet Explorer. Submit change requests and run queries. On Windows you can also run charts and reports.
ClearQuest Microsoft Project Tracker	Everyone	Exchange and synchronize data between Microsoft Project 2000 and ClearQuest.
Administrator tools		
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 10, "Administering ClearQuest E-mail."

Understanding ClearQuest schemas and databases

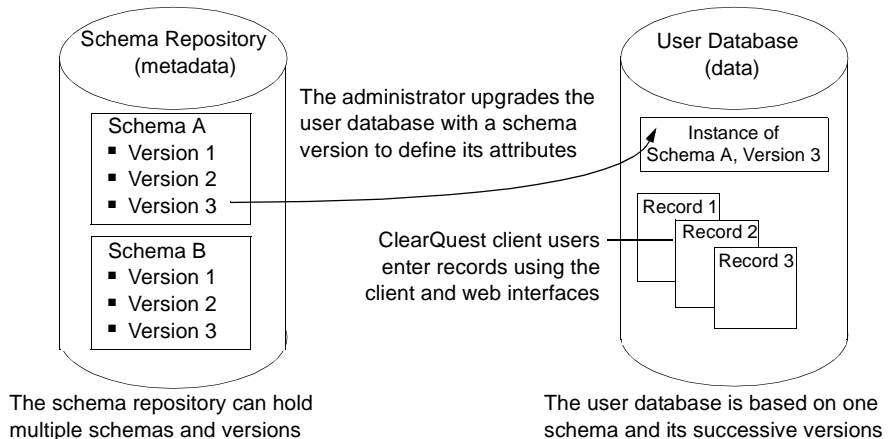
A ClearQuest schema contains the metadata that defines your workflow, that is, how your users work with records within ClearQuest. It includes record definitions, form and field definitions, record states and the actions that can be performed on records, and optional hook code that customizes your workflow.

ClearQuest 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 use ClearQuest Designer to customize schemas and to apply schemas to user databases.



ClearQuest supports multiple schemas. You can apply each schema to a different user database, or apply a single schema to multiple databases.

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.”

Designing your ClearQuest workflow

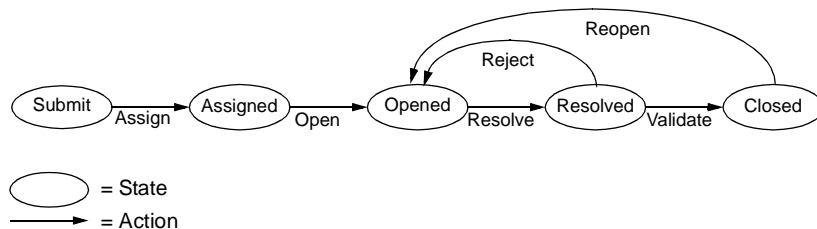
ClearQuest allows you to track multiple types of records, such as defects and enhancement requests. As the ClearQuest administrator, you will use ClearQuest Designer to define how your users will work with records within the ClearQuest client. This section provides a brief overview to help you understand the basic elements involved in defining how users work with records in the ClearQuest client and some of the opportunities you have to customize this workflow.

ClearQuest includes several predefined schemas that define common workflows. You can select the predefined ClearQuest schema that most closely fits your workflow, using it as is or customizing it to fit your needs. You can also create your own schema by starting with the ClearQuest Blank schema. 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 “Working with packages” on page 46.

Understanding record types, states, and actions

ClearQuest client users work with records by moving them through various stages, or “states,” usually beginning when the record is submitted to the system and ending when the record is closed. Movement from state to state is initiated when a user performs an action such as Open or Resolve on the record.

Each record type in a schema has a state model that defines how that record can be used. The state model defines all the states the record can be in and the actions that can be performed on the record in each state. 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 quality assurance engineers 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.

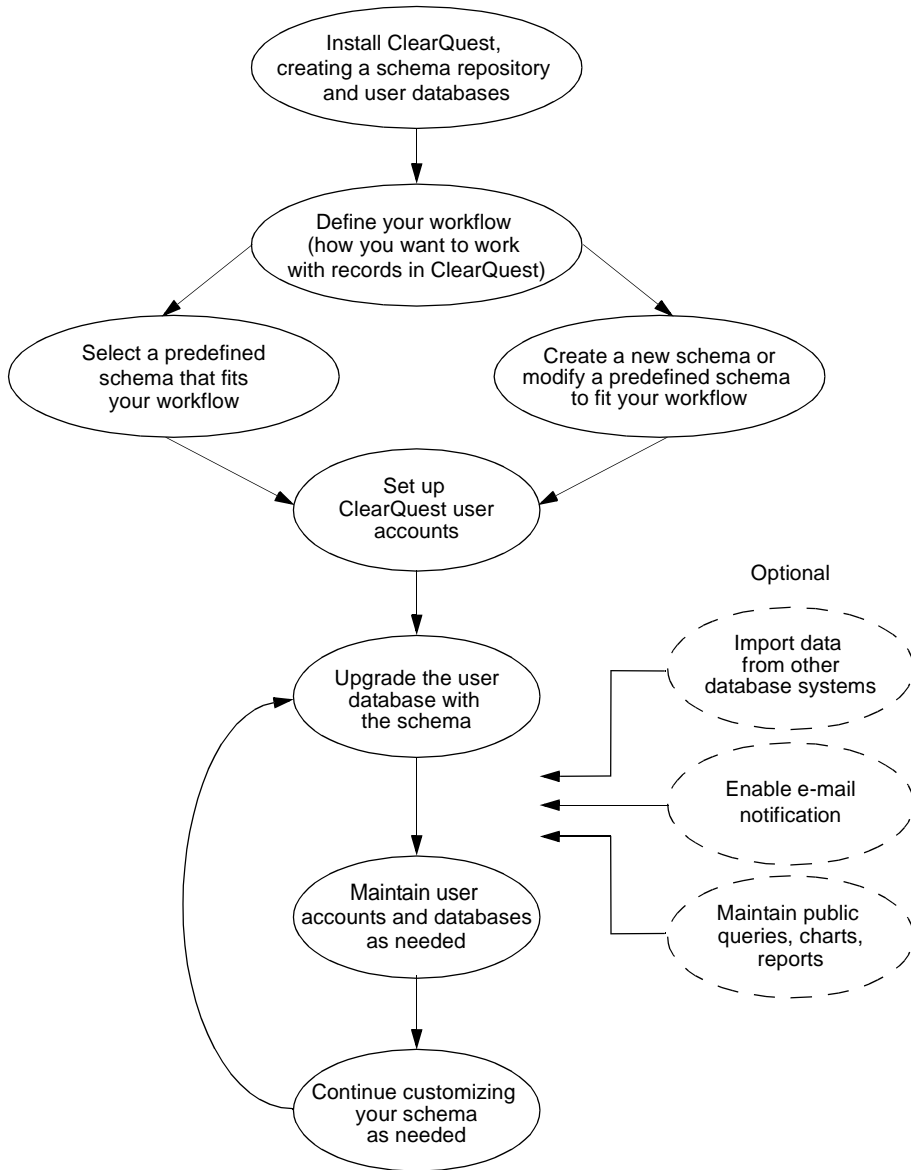
Before you begin customizing

Before you begin customizing ClearQuest, be sure to read:

- Chapter 3, “Working with ClearQuest schemas.” This chapter is essential reading whether you are using a predefined schema as is, customizing a schema, or starting with Blank schema. It describes the process for working with schemas, from checking them out of the schema repository, to applying them to the user database. It also describes how you can set up a test database to test your schema customizations in ClearQuest client without risking changes to your user database.
- Chapter 4, “Customizing a schema,” especially the section “Overview of customizing a schema” on page 52. This describes how to add record types, fields, states and actions to a schema.
- Chapter 5, “Working with forms” to learn how to create and modify record forms.
- Appendix A, “ClearQuest schemas and packages” for a complete list of ClearQuest’s predefined schemas and schema packages. To learn how to add packages to a schema, see “Working with packages” on page 46.

Overview of administrator tasks

Below is an overview of the ClearQuest administrator tasks.



Getting ClearQuest users started

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

Task	Do this	To find out how, see
Install ClearQuest and set up databases	Create a schema repository and one or more user databases	<i>Installation Guide for Rational ClearQuest</i>
Define your workflow	Plan states and actions; create a state transition model; define record types, forms and fields; plan hook code	Chapter 3, "Working with ClearQuest schemas" Chapter 4, "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 Customize the schema as needed Write hook code to further refine your workflow (optional)	"Working with packages" on page 46 Appendix A, "ClearQuest schemas and packages" Chapter 3, "Working with ClearQuest schemas" and Chapter 4, "Customizing a schema" Chapter 8, "Using hooks to customize your workflow"
Upgrade your user database with the selected/customized schema		Chapter 3, "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"
Import data from other database systems (optional)	Note: You must set up user accounts before importing data from other database systems	Chapter 11, "Importing and exporting with ClearQuest"
Enable e-mail notification (optional)	Have each ClearQuest user enable e-mail notification in ClearQuest Set up e-mail rules	Chapter 10, "Administering ClearQuest E-mail"

Task	Do this	To find out how, see
Enable ClearQuest Web (optional)		<i>Installing Rational ClearQuest</i> Chapter 9, "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 Alternatively, give permissions to other users, such as project managers, to do this	ClearQuest Help Chapter 6, "Administering users"

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

- Working with ClearQuest databases
- Creating new databases
- Connecting to ClearQuest
- Moving a database to a new location or to a new vendor
- Deleting databases
- Viewing database properties

Working with ClearQuest databases

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.

For a description of how ClearQuest databases and schemas work together, see “Understanding ClearQuest schemas and databases” on page 7.

Backing up your data

Develop a strategy for performing regular database backups. 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.

Whenever you do backups, you **MUST** back up your user databases and your schema repository at the same time. This preserves both your data and customizations.

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 <i>SAMPLE</i> user database, use the ClearQuest Maintenance Tool.
	and ClearQuest Designer	To create new user databases, use ClearQuest Designer. Note: 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, use only ClearQuest tools to directly modify ClearQuest data or tables. Do not use your database vendor tools to modify ClearQuest data.
Move a database or change database vendors	ClearQuest Designer or	To move a user database to a new location or to a different database vendor, use ClearQuest Designer.
	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 20.
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 databases" on page 27.
Upgrade to a new ClearQuest version	ClearQuest Maintenance Tool	To upgrade to a new version of ClearQuest, use the ClearQuest Maintenance Tool.

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 vendor databases, see the *Installation Guide for Rational 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 33. If you plan to use ClearQuest with a MS Access or SQL Anywhere database, you can create these databases when you use ClearQuest.

Once your databases are set up, you can begin using and customizing ClearQuest. See Chapter 3, “Working with ClearQuest schemas” in the *Administrator’s Guide for Rational ClearQuest*.

Creating a new schema repository

Before you create a new schema repository, you must create a vendor database to use for your schema repository. If you are using Microsoft Access or SQL Anywhere, you are prompted to make a share.

To create a new schema repository, use the ClearQuest Maintenance Tool.

- 1 Select **Schema Repository > Create**.
- 2 In the **Existing Connections** area, type a name for the schema repository connection, then press ENTER.
- 3 In the **Schema Repository Properties** area, enter the properties of the vendor database designated for the schema repository.
 - a Choose a vendor database from the drop-down list.
 - b Enter the required vendor database properties.
- 4 Click **Next**.
- 5 In the **Sample Database** area:
 - a Select the **Create sample database** check box.
 - b In the **Sample DB Schema** list, choose a schema to use for your sample database.
 - c Optionally, in the **Description** box, enter a description.

- 6 In the **Sample User Database** area, enter the properties of the vendor database designated for the sample database.
- 7 Click **Finish**.

Creating a new user database

Before you create a user database, you must create a vendor database to use for your user database. If you are using Microsoft Access or SQL Anywhere, you don't need to have already created a vendor database.

To create a new user database, use the ClearQuest Designer.

- 1 In the ClearQuest Designer, select **Database > New Database**.
- 2 In the **Logical Name** box, provide a name for your user database. The name cannot be more than five characters long and can use only alphanumeric characters.
- 3 Click **Next**.
- 4 In Step 2 of the New Database Wizard, enter the properties of the vendor database designated for the user database.
 - a Choose a vendor database from the drop-down list.
 - b Enter the required vendor database properties.
 - c Select whether this user database will be used in production or as a test database.
- 5 Click **Next**.
- 6 Enter a value for timeout and poll interval, then click **Next**.
- 7 Select a schema from the **Schema** list, then click **Finish**.

Connecting to ClearQuest

Use the ClearQuest Maintenance Tool to connect to an existing schema repository and user databases in the following circumstances:

- To create a connection to a newly-created schema repository.
- To re-connect to a schema repository after it has been moved or changed.

- 1 Select **Connection > New**.
- 2 In the **Existing Connections** area, name the alias by typing a name in the highlighted item.
- 3 In the **Schema Repository Properties** group box, enter the properties of the vendor database designated for the schema repository.
 - a Choose a vendor database from the drop-down list.
 - b Enter the required vendor database properties.
- 4 Click **Finish**.

Note: Your ClearQuest administrator might have saved a connection profile that includes all of the database information you need, see “Using connection profiles” on page 19.

Modifying connections

You can modify existing connections at anytime. You can rename, edit the database properties, and create duplicate connections.

Renaming a connection

You can rename a connection at any time. Renaming a connection does not change the database information associated with that connection. Keep in mind that connections are maintained on a machine-basis and that the same schema repository connection can have different names on different machines.

To rename a connection:

- 1 In the **Existing Connections** area, select the connection you would like to rename.
- 2 Click **Connection > Rename**.
- 3 Type the new name, then press ENTER.

Editing a connection

When you edit a connection, you change the database properties of the associated schema repository. You need to edit a connection if the schema repository has been moved or changed by your ClearQuest administrator.

To edit a connection:

- 1 In the **Existing Connections** area, select the connection you would like to edit.
- 2 Click **Connection > Edit**.
- 3 In the **Schema Repository Properties** group box, enter the new properties of the schema repository.
 - a Choose a vendor database from the drop-down list.
 - b Enter the required vendor database properties.
- 4 Click **Finish**.

Deleting a connection

You can delete a connection when you no longer need it. Remember that connections are kept on a machine-basis and do not effect other ClearQuest users.

If you need to retrieve a deleted connection, use the **Connection > New** option or import an existing Connection Profile that contains the connection you need.

To delete a connection:

- 1 In the **Existing Connections** area, select the connection you would like to delete.
- 2 Click **Connection > Delete**.
- 3 When asked if you want to delete the connection, select **Yes**.

Duplicating an existing connection

In some cases, you may need to duplicate an existing connection. For example, if you need to connect to a new schema repository whose properties are similar to an existing connection, you can duplicate an existing connection and then you use the **Connection > Edit** option to modify the properties of the duplicated connection to match the new, but similar schema repository.

To duplicate an existing connection:

- 1 In the **Existing Connections** area, select the connection you would like to duplicate.
- 2 Click **Connection > Duplicate**.
- 3 Optionally, type a new name for the duplicate connection and then press ENTER.

Using connection profiles

Rather than re-entering database connection information each time you need to re-connect to a schema repository, you can create connection profiles that can be used as shortcuts.

Use the ClearQuest Maintenance Tool to export and import connection profiles.

Connection profiles allow you to save database settings for later use, or for distribution to a team, to make it easier to reconnect to a ClearQuest schema repository.

Both ClearQuest administrators and ClearQuest client users can import and export connection profiles for later use.

Creating a connection profile

When you create a connection profile, you “export” your connection information to a file that can later be imported.

- 1 Select **File > Export Profile**.
- 2 In the **Export Connection** area, select the connections that you would like to include in your profile.
- 3 In the **File Name** box, enter the pathname and file name for the profile. Connection profiles must be saved with a .ini extension.

Optionally, you can use the browse button to browse to a location to which to save the profile.

Importing a connection profile

To use a previously-created connection profile, you need to import a connection profile.

- 1 Select **File > Import Profile**.
- 2 In the **Import Profile Information** area, enter the path and file name of the profile you would like to import.

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.

Note: Rational recommends that you use only ClearQuest tools to move and manipulate ClearQuest databases. Clear Quest stores the location of the schema repository and its user databases in the schema repository.

If you use vendor database tools to move the physical location of a schema repository, ClearQuest will lose the connection to the schema repository. In this case, you must use the Maintenance Tool to update the properties of the moved schema repository.

Preparing to move a ClearQuest database

When you move a schema repository or user database, either to a new location or to a different vendor database (or both at once), ClearQuest copies the schema repository or user database to the new location, locks the old schema repository and marks the copy in the new location as the active schema repository or user 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 the *Installation Guide for 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 of your databases. Even if you are moving just one user database or only your schema repository. You **MUST** back up **ALL** of your ClearQuest databases (both the schema repository and all associated user databases) to ensure data integrity if data loss occurs during the move due to unforeseen circumstances (network failure, etc.).

Moving a schema repository

To move a schema repository, use the ClearQuest Maintenance Tool.

- 1 In the **Existing Connections** area, select the schema repository that you want to move.
- 2 Click **Schema Repository > Move**.
- 3 In the **Login** area, log in to the schema repository by entering your user name and password. You must have Super User privileges to move a schema repository.
- 4 Click **Next**.
- 5 In the **New Schema Repository Properties** area, enter the properties of the vendor database designated for the schema repository.
 - a Choose a vendor database from the drop-down list.
 - b Enter the required vendor database properties.
- 6 Click **Finish**.
- 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 *Installation Guide for Rational ClearQuest* for details on the information needed to connect.

Updating schema repository properties

If you move a schema repository using vendor database tools instead of the ClearQuest Maintenance Tool, you must update the connection to the schema repository using the ClearQuest Maintenance Tool.

If you are already connected to the schema repository:

- 1 In the **Existing Connections** area, select the connection you need to update.
- 2 In the ClearQuest Maintenance Tool, select **Schema Repository > Update Properties > Selected Connection**.
- 3 In the **Schema Repository Properties** area, enter the properties of the vendor database designated for the schema repository.
 - a Choose a vendor database from the drop-down list.
- 4 Enter the required vendor database properties. Click **Next**.
- 5 In the **Login Information** area, enter your login name and password. You must have Super User privileges to upgrade a schema repository and/or user databases.
- 6 Click **Next**.

Note: When you relocate a schema repository, the ClearQuest Maintenance Tool automatically creates a connection to the schema repository. For example, **Connection 1**. If you want to rename this connection, use the **Connection > Rename** option (see “Renaming a connection” on page 17).

- 7 In the **New Schema Repository Properties** area, enter the properties of the vendor database designated for the schema repository.
 - a Choose a vendor database from the drop-down list.
 - b Enter the required vendor database properties.
- 8 Click **Finish**.
- 9 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 *Installation Guide for Rational ClearQuest* for details on the information needed to connect.

If you are not currently connected to the schema repository:

- 1 In the ClearQuest Maintenance Tool, click **Schema Repository > Update Properties > Other**.
- 2 In the **Schema Repository Properties** area, enter the properties of the vendor database designated for the schema repository.
 - a Choose a vendor database from the drop-down list.
- 3 Enter the required vendor database properties. Click **Next**.
- 4 In the **Login Information** area, enter your login name and password. You must have Super User privileges to upgrade a schema repository and/or user databases.
- 5 Click **Next**.
- 6 In the **New Schema Repository Properties** area, enter the properties of the vendor database designated for the schema repository.
 - a Choose a vendor database from the drop-down list.
 - b Enter the required vendor database properties.
- 7 Click **Finish**.
- 8 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 *Installation Guide for Rational ClearQuest* for details on the information needed to connect.

Moving a user database

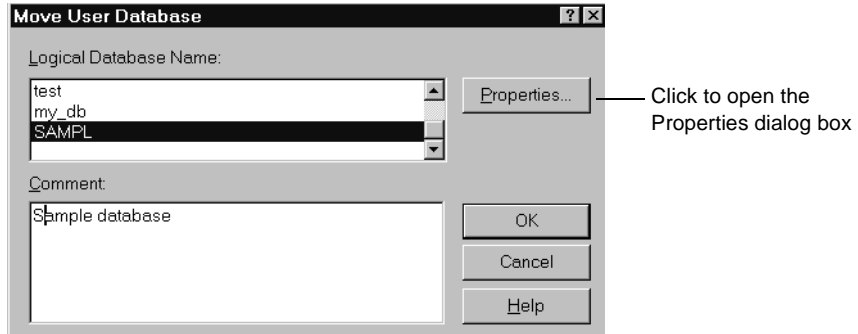
To move a user database, either to a new location or to a different vendor database (or both), use the ClearQuest Designer.

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

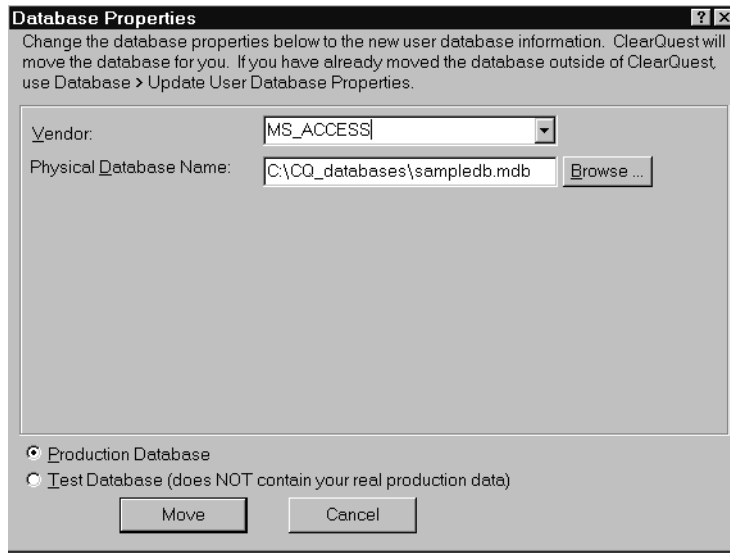
Note: If you have already moved the database outside of ClearQuest and only need to change the database properties, follow the steps described in “Updating the properties of a user database” on page 26.

- 1 Complete the steps in “Preparing to move a ClearQuest database” on page 20.
- 2 Select **Rational ClearQuest Designer** from the Start menu.

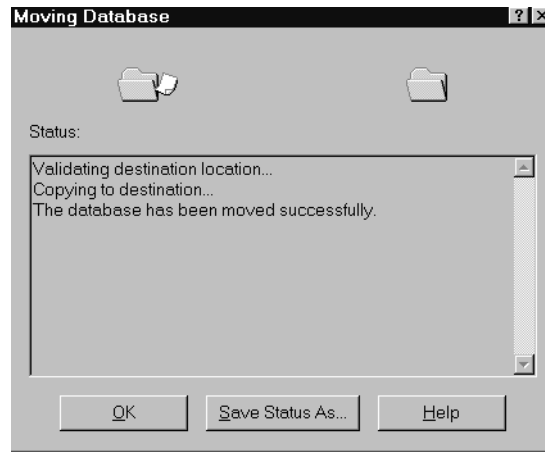
- 3 In ClearQuest Designer, select **Database > Move User Database**.
- 4 In the Move User Database dialog box, select the user database and click **Properties**. (Note that if you click **OK** at this point you will only be updating any changes you have made in the **Comment** text box of the dialog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



- 5 In the Properties dialog box, 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**.



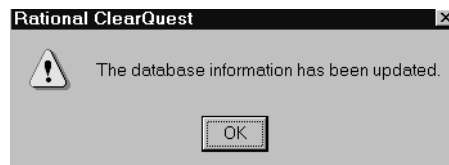
- 6 In the Moving Database dialog box, the message in the Status area informs you whether the database move is successful.



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

Updating the properties of a user database

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, follow the steps described in “Moving a database to a new location or to a new vendor” on page 20 to change properties prior to moving the database.)

- 1 In ClearQuest Designer, select **Database > Update User Database Properties**.
- 2 Select the desired database name.

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

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

Deleting databases

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 28.

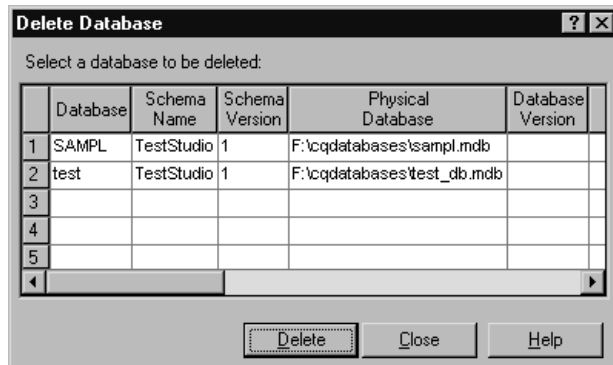
Note: The link to a deleted database is permanently lost if you delete the schema version associated with the database or if you upgrade to a newer version of ClearQuest.

ClearQuest does not allow you to delete a schema repository.

Deleting a user database

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 box.



- 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.

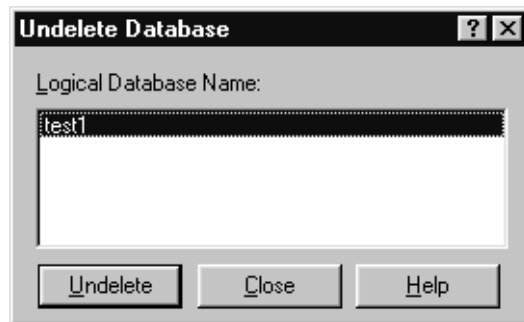
Undeleting a user database

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 box.



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

Viewing database properties

To view user database properties, use the ClearQuest Designer.

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

Note: This window is read-only. To move and/or edit database properties, see “Moving a database to a new location or to a new vendor” on page 20 or “Updating the properties of a user database” on page 26.

A ClearQuest schema contains the metadata (the record types, actions, states, fields, forms, and hooks) that define how you work with records in ClearQuest. 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 schema procedures
- Working with a test database
- Selecting a ClearQuest schema
- Changing a database to a different schema
- Working with packages

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.”

For information on how to customize a schema, see Chapter 4, “Customizing a schema.”

ClearQuest includes several predefined schemas that provide common workflow models. For a list of predefined schemas and packages, see Appendix A, “ClearQuest schemas and packages.”

For a description of the relationship between schemas and databases, see “Understanding ClearQuest schemas and databases” on page 7.

Overview of schema procedures

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

- 1 Create a test database and associate it with the schema you want to customize. See “Working with a test database” on page 33.
- 2 Check out a schema from the schema repository, or create a new schema. See “Checking out a schema” on page 35 or “Creating a new schema” on page 37.
- 3 Set the test database for the schema by selecting **Database > Set Test Database**. See “Working with a test database” on page 33.
- 4 Select the scripting language to use for hook code. See “Selecting a scripting language” on page 38.
- 5 Customize the schema as desired. For complete instructions on how to customize a schema, see Chapter 4, “Customizing a schema.”

Note: A quick way to customize a schema is to add ClearQuest packages to get the functionality you need. See “Working with packages” on page 46.

- 6 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 33.
- 7 Validate the schema often, especially when making complex changes. See “Validating schema changes” on page 39.
- 8 Save work in progress, if desired, by selecting **File > Save Work**. At any time before checking in a schema, you can save your work in progress. You can then log off while the schema is still 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. See “Saving work in progress” on page 42.
- 9 Check in the schema. Once you have tested and validated your customizations and are satisfied that the schema will work as intended, you are ready to check the schema back into the schema repository. See “Checking in a schema” on page 40.
- 10 Apply the new schema version to the user database by selecting **Database > Upgrade Database**. See “Applying schema changes to a user database” on page 41.

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. Working with a test database involves:

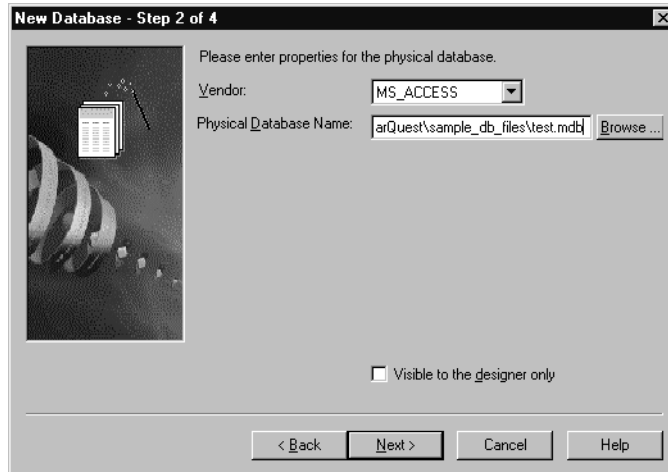
- Creating a test database
- Setting the test database for the schema
- Testing your work

Creating a test database

To create 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.



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 the ClearQuest Import Tool.

- 2 Next, you must check out the schema you want to customize and set the test database. See “Checking out a schema” on page 35 and “Setting the test database for the schema” on page 34.

Setting the test database for the schema

Before you can set the test database for the schema, you must first create a test database and associate it with the schema you want to modify. See “Working with a test database” on page 33.

To set the test database for a schema:

- With the schema checked out, select **Database > Set Test Database** and select the test database you created to use with this schema.

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.

Testing your work

At any time before you check in a schema you can test your work in progress without actually affecting your user database:

- 1 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 39.

- 2 ClearQuest 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.) When testing your schema on UNIX, pay special attention to the layout of forms and the execution of scripts.

- 3 Fix any validation errors. You cannot check in a schema that has validation errors.

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.

Working with schemas

This section describes procedures for working with schemas.

Checking out a schema

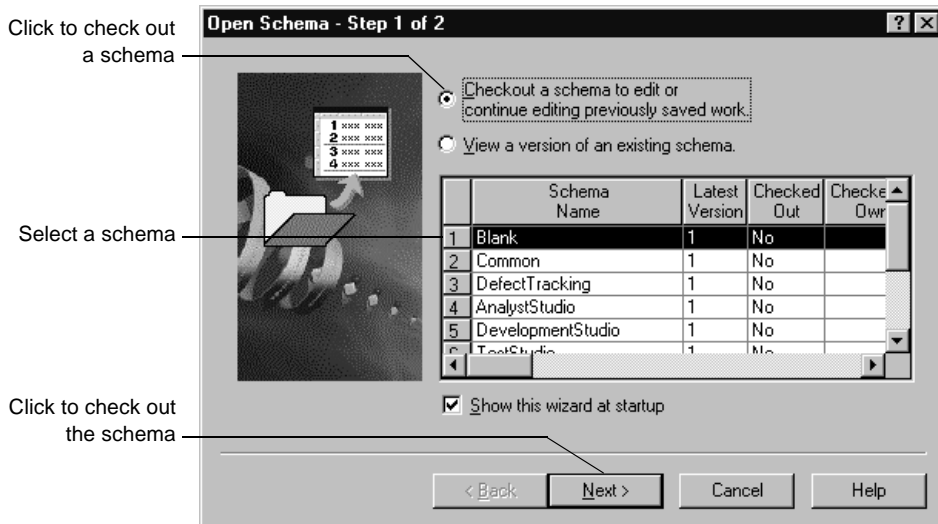
ClearQuest maintains schemas under version control in the schema repository. To customize a schema, you must first check out the schema from the schema repository. When you check out a schema, ClearQuest creates a new version of the schema for you to edit.

Checking out a schema when you log in

After you log into ClearQuest Designer, the Open Schema dialog box is automatically displayed. You can check out a schema at this time or you can open ClearQuest Designer without checking out a schema.

To check out a schema:

- 1 In the Open Schema dialog box, select **Check out a schema to edit...**
- 2 Select a schema version and click **Next**.
- 3 Enter comments in the **Comment** text box, if desired, then click **Finish**.



Opening a schema for viewing only

To open a schema for viewing only:

- 1 In the Open Schema dialog box, click **View a version of an existing schema**.
- 2 Select a schema version to view, then click **Next**.

The selected schema opens as read-only; you cannot customize it.

If you open another schema while you are viewing a schema, ClearQuest closes the schema you are viewing.

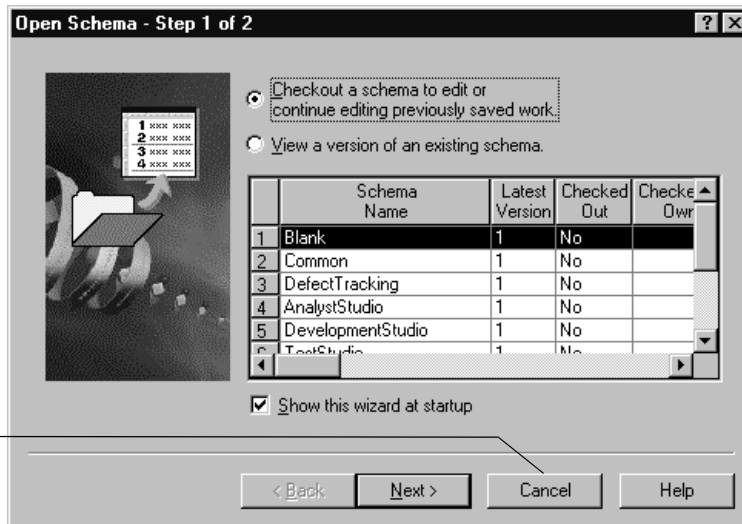
To close a schema that you are viewing:

- Select **File > Close Work**.

Starting ClearQuest Designer without checking out a schema

To start ClearQuest Designer without checking out a schema:

- In the Open Schema dialog box, click **Cancel**.



Checking out a schema from ClearQuest Designer

To check out a schema from within ClearQuest Designer:

- 1 Select **File > Open Schema**.
- 2 Select **Check out a schema to edit**.

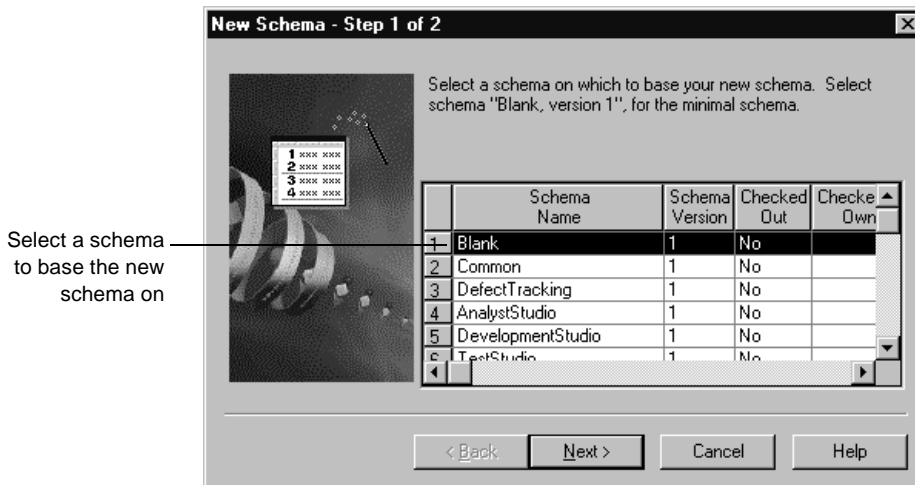
- 3 Select a schema and version, then click **Next**.
- 4 Enter comments in the **Comment** text box, if desired, then click **Finish**.

Creating a new schema

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

- 1 In ClearQuest Designer, select **File > New Schema**.
- 2 In the New Schema dialog box, select a schema to base the new schema on and click **Next**.

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



- 3 Type a **Schema Name**, enter **Comments** if desired, then click **Finish**.
- 4 ClearQuest Designer creates Version 1 of the new schema and asks if you want to associate a database with the new schema. Click **Yes** to open the New Database Wizard and associate a database. Click **No** if you do not want to associate a database at this time.
- 5 ClearQuest Designer asks if you want to check out the new schema for editing. Click **Yes** to check out the schema, enter **Comments** if desired, and click **OK**.

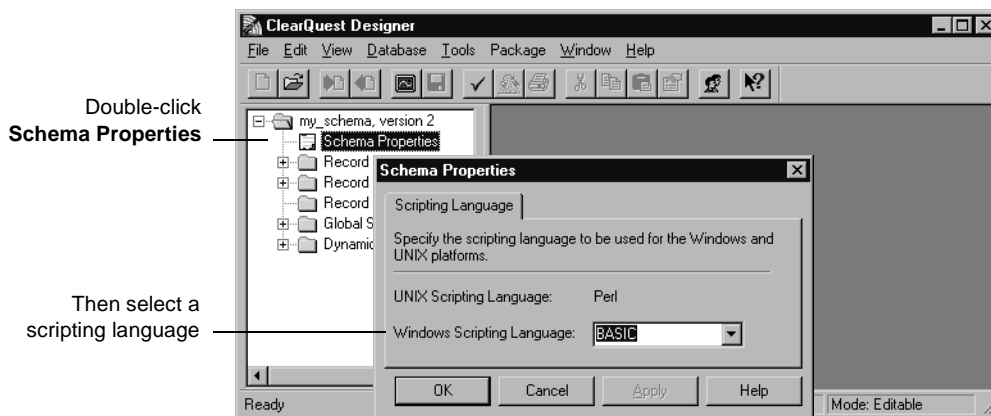
Selecting a scripting language

Hooks are triggers for pieces of code that ClearQuest executes at specified times to implement your workflow. You can write hooks in BASIC (VBScript) for Windows, and in Perl for both Windows and UNIX. (You must use Perl when scripting for UNIX.)

By default, ClearQuest runs scripts in BASIC on Windows and in Perl on UNIX. After checking out a schema, you can select the scripting language that you want ClearQuest to use to run scripts on Windows.

To select a scripting language:

- 1 In the Workspace, double-click **Schema Properties**.
- 2 In the Schema Properties dialog box, select **BASIC** or **PERL**.



For additional information, see Chapter 8, “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

When you attempt to check in a schema, ClearQuest automatically validates the schema to ensure that it does not contain errors. ClearQuest verifies that the state transition matrix is valid, and also tests that each control on the form is associated with a field. Schema validation does not catch errors in hooks.

You cannot check in a schema if there are any validation 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.

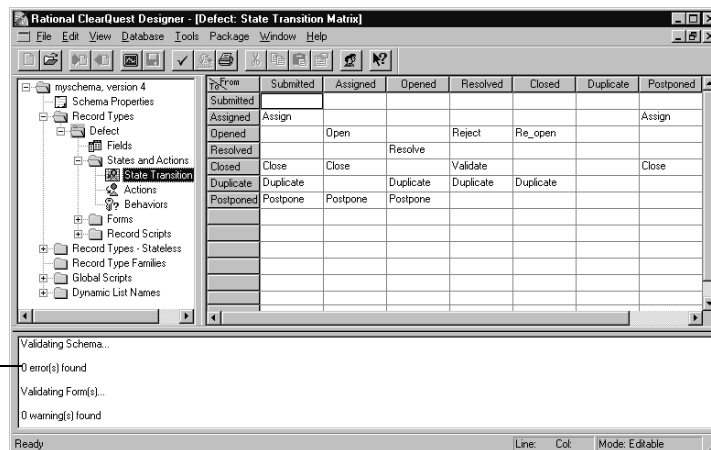
Note: Validation does not guarantee that a schema will function as desired. Be sure to test a schema with a test database before checking it in, and back up your user database before applying schema changes to it. See “Working with a test database” on page 33. Hook code, in particular, can introduce subtle errors at runtime if not written correctly. To test hooks, select **Hooks > Compile** in the Scripts Editor and debug your hook code as needed.

To validate a schema:

1 Select **File > Validate**.

ClearQuest displays the validation results in the Validation pane at the bottom of the ClearQuest Designer window.

Right-click an error and select **What's This?** to get more information on the error



2 Review the validation results and modify the schema as needed to correct any errors.

To get more information about an error, select the error message in the Validation pane, right-click the error message, and select **What's This** from the shortcut menu.

If you cannot validate all of your schema changes, you can save the 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 41.

Checking in a schema

Before checking in a schema, you should test the schema on a test user database. See “Working with a test database” on page 33.

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.

Once the new schema version is checked in, you can apply it to an existing user database or to a new database. ClearQuest keeps a history of each schema version, so you can view a prior version and use it in a new database.

To check in a schema:

- 1 Select **File > Check In**.

ClearQuest validates the schema, displaying any errors in the Validation pane at the bottom of the ClearQuest Designer window.

- 2 Review the validation results.

If there are validation errors, you must change the schema as necessary to correct the validation errors. You cannot check in a schema if there are validation errors.

For help on a validation message, right-click the message and select **What's This** from the shortcut menu. See “Validating schema changes” on page 39.

- 3 When the schema passes validation, ClearQuest Designer displays the Check In dialog box with your original check-out **Comments** displayed. Edit the comments if desired, then click **OK**.

Note: Checking in a schema does not affect the user database; you must apply the new version of the schema to the user database to have your changes take effect. See “Applying schema changes to a user database” on page 41.

Undoing a schema checkout

You can return a schema to its previous version, even after saving any schema 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 undo a schema checkout:

- Select **File > Undo Check Out**.

Note: If you test your work by selecting **File > Test Work**, ClearQuest automatically updates 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.

Applying schema changes to a user database

After checking in a schema, you must apply the new schema version to the user database to have your changes take effect. Keep in mind the following restrictions when you apply the schema changes to a user database:

- Once a user database is associated with a specific schema version, you can only apply newer versions of the same schema to that database. You cannot apply earlier versions or a different schema to the database.
- Before applying schema changes to a user database, back up the user database.

Warning: Before applying changes to a user database, make sure that there are no users logged in to the database. If users are connected to the user database, the upgrade will fail. ClearQuest prevents new users from logging in to a database while you are updating it, but it does not disconnect users who are currently logged in.

To apply schema changes to a user database:

- 1 Select **Database > Upgrade Database**.

ClearQuest prompts you to back up the schema repository and the user database before updating. If you have not already backed up the database, click **No**; otherwise, click **Yes**.

- 2 Select a database from the **Database** list and click **Next**.
- 3 Select a schema version from the **Versions** list and click **Finish**.

For more information, see Chapter 2, “Managing databases.”

Saving work in progress

You can edit the same schema version over multiple editing sessions without committing changes or incrementing the schema version number. You can check out a schema and edit it, save your work in progress, then log off. The next time you log in you can continue working on the same schema version. This feature is useful when you want to save your work so far, but are not yet ready to commit the changes to the database.

To save your work in progress during a schema editing session:

- 1 Select **File > Save Work**.
- 2 Select **File > Close Work** or **Exit** to end the editing session.

To reopen the schema and continue editing:

- 1 If you are logging in to ClearQuest Designer, the Open Schema dialog box is automatically displayed. Or, from within ClearQuest Designer, select **File > Open Schema** to open the Open Schema dialog box.
- 2 In the Open Schema dialog box, click **Open a schema for editing or continue editing a previously saved work** and select the schema version you previously worked on.

ClearQuest Designer opens the schema version for editing.

Note: Before checking the schema into the schema repository, be sure to test your work and validate the schema. See “Working with a test database” on page 33, “Validating schema changes” on page 39, and “Checking in a schema” on page 40.

Deleting a schema or schema version

Note: All schema deletions are final. You cannot retrieve a deleted schema or schema version.

You can delete a single version of a schema or all versions of a schema. Before you attempt to delete a schema or schema version, make sure:

- The schema is not currently associated with a user database. If it is, you must first delete the user database.
- The schema is not currently checked out of the schema repository. If it is, you must first check in the schema.

To delete a schema or schema version from the schema repository:

- 1 In ClearQuest Designer, select:
 - **File > Delete Schema** to delete all versions of the schema.
 - **File > Delete Schema Version** to delete a single version of a schema.
- 2 Select the desired schema or schema version and click **Delete**. Click **Yes** to confirm the delete.

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 “Working with packages” on page 46.

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 PureCoverage.
UnifiedChangeManagement	Supports the UCM process by providing integration with Rational ClearCase. See “Adding a Rational UCM integration” on page 284.

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

Changing 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 user 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 change a user database to a different schema:

- 1 Before you begin, read “Moving a database to a new location or to a new vendor” on page 20.

- 2 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.

- 3 Use the ClearQuest Export Tool to export the data from your current user database. For more information, see “Using the ClearQuest Export Tool” on page 252.

- 4 Use the ClearQuest Import Tool to import the data into the new database. For more information, see Chapter 11, “Importing and exporting with ClearQuest.”

Working with packages

ClearQuest's schema packages contain metadata such as records, fields, and forms that define specific functionality. Adding a package to 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 Appendix A, "ClearQuest schemas and packages." To view a list of the packages that are installed in a schema, see "Viewing the packages in your schema" on page 275.

Keep in mind the following when adding packages to schemas:

- You can add one or more packages to a schema.
- You cannot install a package version into a schema that already contains that package version.
- Some packages modify specific record types. If you attempt to add a package that modifies a record type to a schema that does not already include that record type, the addition will fail.
- You can specify whether a package should modify an existing record type.
- If you are upgrading to a new version of ClearQuest, add the latest version of the package(s). See the *Installation Guide* for Rational ClearQuest.
- 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.

Warning: Be sure to plan carefully before adding packages. Once you add a package to a schema, you cannot remove the package. You must delete all schema versions in which the package exists. You can delete schema versions only if you have not applied them to a user database.

Adding packages to a schema

To add a package to a schema, you use the Package Wizard:

- 1 In ClearQuest Designer, make sure that the schema you want to add the package to is checked in. To check in a schema, select **File > Check In**.
- 2 Select **Package > Package Wizard**.



- 3 In the Package Wizard, select a package to add, then click **Next**.

Note: If the schema you are attempting to add a package to is not checked in, the Package Wizard will not list the packages that are already part of your schema. Be sure to check in your schema before adding packages.

Also, packages that are not already installed in the schema repository are not listed. If you need a package or package version that is not listed, click **More Packages** to first install packages into the schema repository. After you do this, the package will be available to add to your schema. (If you are looking for the latest version, click **More Packages** to see if a newer version exists.)

- 4 You might also need to enable record types and set up state types for the package you are installing. See “Enabling record types” on page 48, and “Setting up state types” on page 49.

Enabling record types

Some packages modify record types that already exist in your schema. You can decide whether or not the package should modify these record types. For example, the package you are adding might add additional fields to existing record types, so you must indicate which record type(s) you want these fields added to. The package might also add actions, scripts, and so on.

For descriptions of packages, including the record types they modify, see “ClearQuest packages” on page 259.

To enable record types:

- When you install a package, if the list box of the Setup Record Types for Package dialog box appears, then enable the record types that you want the package applied to.



Note: If you decide not to enable a record type when you install the package, you can do so later by selecting **Package > Setup Record Types for Packages**. You can also do this to enable this package for record types you add later.

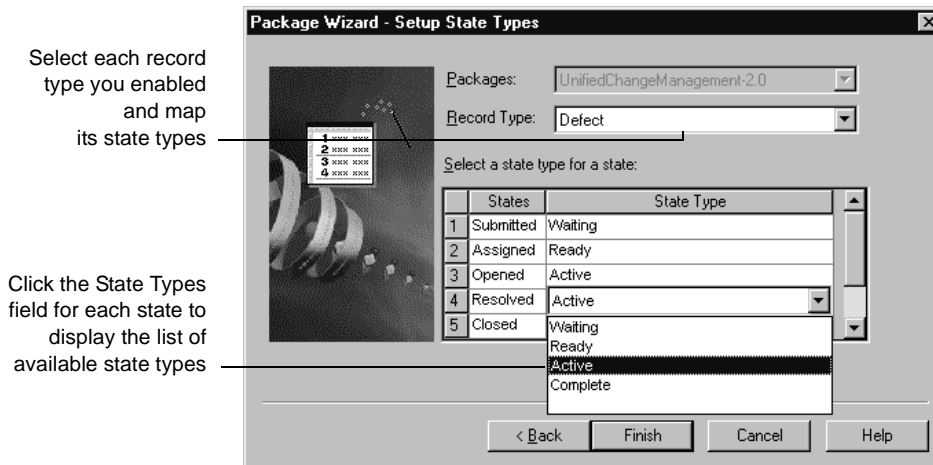
Setting up state types

Some packages modify a schema by adding hooks that execute when a record type is in specific states. Such packages use state types to identify when the hooks should run. Packages such as the UnifiedChangeManagement package and the Resolution package use state types to define a state's role in your state model. The Resolution package adds a field called "Resolution" and uses state types to determine whether the Resolution field is mandatory or optional in each state.

When you add a package that uses state types, the Setup State Types dialog box is automatically displayed and you must 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.

To set up state types:

- 1 When you install a package, in the Setup State Types dialog box select a **Record Type**.



- 2 For each state in the record type, click in the **State Type** field and select the appropriate state type.

- 3 Repeat the state type mapping for each record type you enabled. When you are done, click **Finish**.

Note: If you need to edit the state types later, you can do so by selecting **Package > Setup State Types** from the main menu.

State-type models are listed in “State-type models for packages” on page 268. To learn how to map state types, see “Mapping state types” on page 78.

- 4 If you are installing the UnifiedChangeManagement package, you must also create default actions. See “Using default actions” on page 87.

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

- Overview of customizing a schema
- Working with record types
- Working with fields
- Defining your state model
- Working with actions and action types

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 3, “Working with ClearQuest schemas.”

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

Overview of customizing a schema

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 types 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 54.
- 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 61.

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 “Working with packages” on page 46.

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 75.
- 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 81 and “Customizing actions by adding hooks” on page 86.
- 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 66.

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 78.

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 forms" on page 89.

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 74.

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 86.

See Chapter 8, "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 Enhancement Requests.

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 259.

State-based record types

State-based record types flow through a process. Users work with these record types by moving them through various states such as Submitted, Assigned, and Resolved. A record of this type is always associated with a state, so it's easy to track the status of the record as it moves through your system. The record moves from state to state through the actions that users perform on it.

You can group two or more state-based record types that have common fields into a record type family, to allow ClearQuest users to query across multiple record types. For example, you might group the Defect record type with the Enhancement Request record type, specifying the Owner field as the common field. ClearQuest client users can then create a query asking for all the defects and enhancement requests owned by a specific engineer. See “Working with record type families” on page 57.

Stateless record types

Stateless record types are typically used for data that is referenced from many different records or record types, such as a list of users, projects, or customers. 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. For example, in the DefectTracking schema, the Defect state-based record type references the Project stateless record type. The ClearQuest user can create a number of projects and then assign defects to the appropriate project.

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

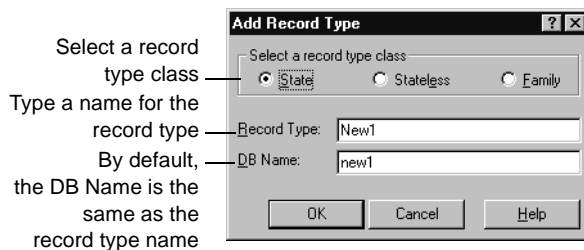
Adding a new record type

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 box, select the class for the record type (**State** or **Stateless**).
- 3 In the **Record Type** text box, type a name for the record type.

Note: ClearQuest uses the DB Name for the table. By default, it's the same as the record type name.



- 4 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 56.
- 5 Click **OK** to close the Record Type dialog box.
- 6 Each schema must have a default record type, which ClearQuest uses to create a shortcut for submitting records. If your schema does not already have a record type specified as the default, you must specify one. See “Selecting a default record type” on page 57.

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 “Working with packages” on page 46. If your schema contains a package, which modifies existing record types, you can explicitly enable the new record type for the package functionality. See “Enabling record types for integrations” on page 274.

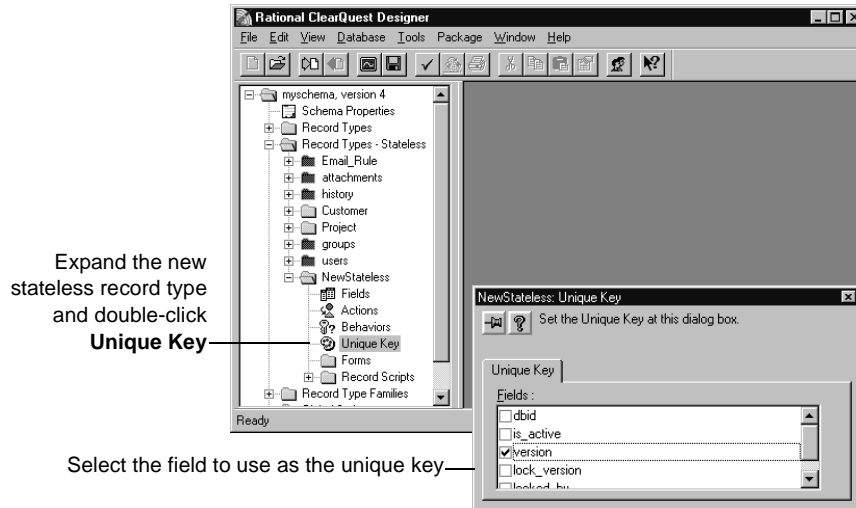
Selecting a unique key for a stateless record type

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.

Note: If your schema contains hook code that explicitly refers to the unique key, and then you change the unique key, be sure to modify your code to refer to the name of the new unique key fields.

To select the unique key for a stateless record type:

- 1 In the Workspace, expand **Record Types - Stateless**, then expand the new record type and double-click **Unique Key**.
- 2 In the Unique Key dialog box, check one or more field names to be used as the unique key.



- 3 Close the Unique Key dialog box.

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:

- In the Workspace, right-click the record type and select **Default Record Type** from the shortcut menu.

Working with record type families

You can group together two or more state-based record types that have common fields to create a record type family. 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.
- 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.

Creating a record type family, involves:

- Adding a record type family
- Adding members to a record type family
- Creating common fields

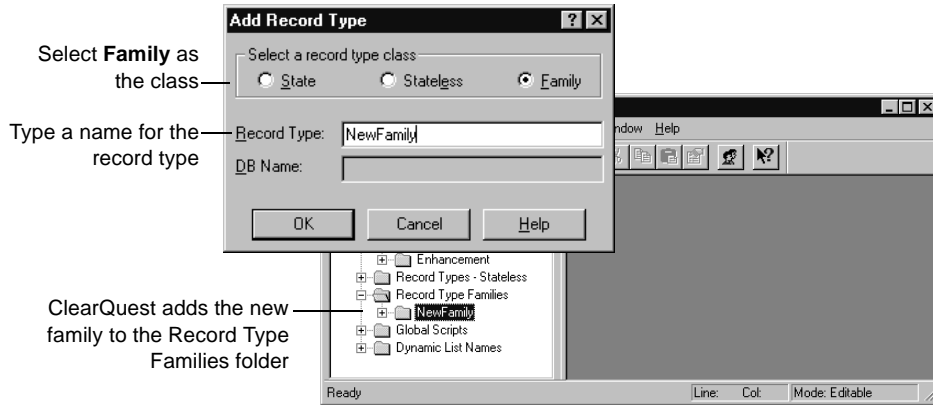
Adding a record type family

To create a new record type family to a schema:

- 1 Select **Edit > Add Record Type/Family**.
- 2 In the Add Record Type dialog box, select **Family** as the record type class.

- 3 In the **Record Type** text box, type a name for the record type family, then click **OK**. ClearQuest Designer adds the new family to the Record Type Families folder.

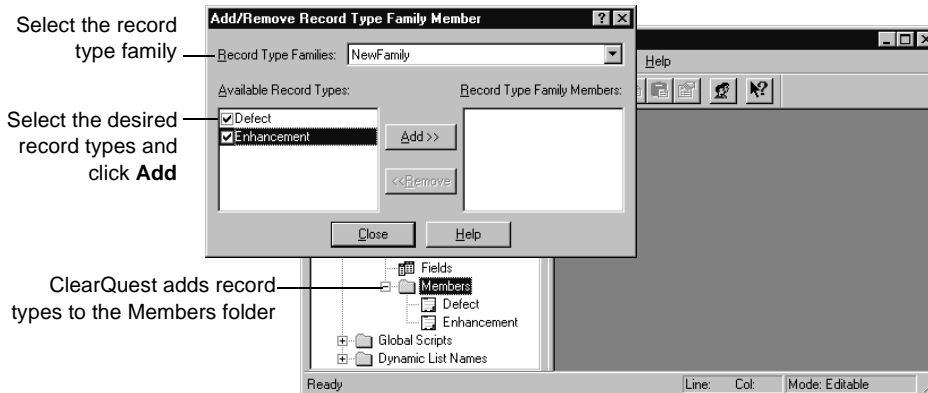
Note: ClearQuest uses the DB Name for the table. By default, it's the same as the record type family name.



Adding members to a record type family

To add the desired record types to the new record type family:

- 1 Select **Edit > Record Type Family Members**.



- 2 In the Add/Remove Record Type Family Member dialog box, select the **Record Type Family** you just created.

- 3 In the **Available Record Types** list, select the record types to add to the family, then click **Add**. When you're done, click **Close**.

Note: To quickly add record types to a family, expand the **Record Types** folder and the family folder, then drag a record type to the family's **Members** folder.

ClearQuest Designer adds the record types to the **Members** folder of the record type family.

Creating common fields

All of the member record types in a record type family must contain one or more common fields. If the record type members do not already contain common fields, you must add common fields.

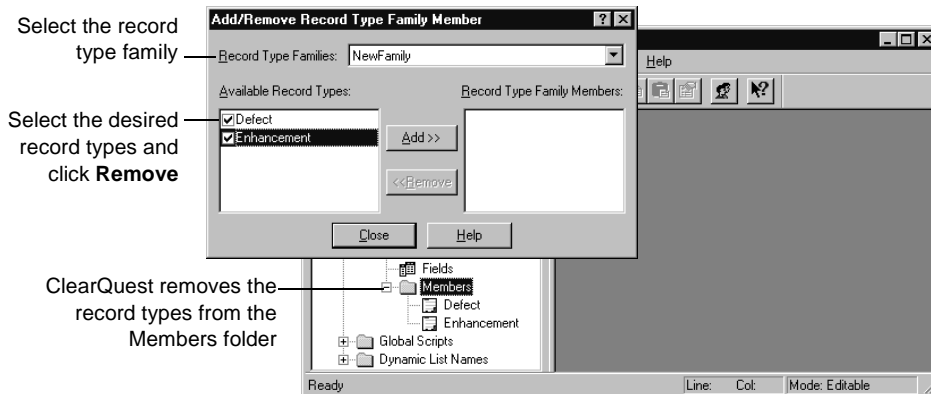
Common fields must have the same name and be of the same data type. For example, to group a Defect record type and an Enhancement record type, you can use the Description field in each record type as the common field as long as that field is the same data type (perhaps short string) in both record types.

To learn how to add fields to a record type and to define their data type, see “Working with fields” on page 61.

Removing members from a record type family

To remove record types from a record type family:

- 1 Select **Edit > Record Type Family Members**.



- 2 In the Add/Remove Record Type Family Member dialog box, select the desired **Record Type Family**.

- 3 In the **Available Record Types** list, select the record types to remove and click **Remove**. When you're done, click **Close**.

Renaming a record type or family

If 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 the hooks, your scripts will not work as intended.

Note: You cannot rename the stateless system record types: `ratl_replicas`, `history`, `attachments`, `groups`, and `users`.

To rename a record type or family:

- 1 Select **Edit > Rename Record Type/Family**.
- 2 Select the class of the record type that you want to rename: **State**, **Stateless**, or **Family**.
- 3 Select the record type from the **Current** drop-down list.
- 4 Type a new name in the **New Name** text box. Within a schema, each record type and record type family name must be unique.

Note: You can also right-click the record type or family in the Workspace and select **Rename** from the shortcut menu.

Deleting a record type or family

You cannot delete the following record types:

- The stateless system record types: `ratl_replicas`, `history`, `attachments`, `groups`, and `users`.
- Record types added by read-only packages.
- The default record type. You must first assign another record type to be the default.

If you explicitly referred to the name of a record type in a hook, you must modify the script code to remove references to that name.

To delete a record type or family:

- 1 Select **Edit > Delete Record Type/Family**.
- 2 Select the class of the record type to delete: **State**, **Stateless**, or **Family**.
- 3 Select the record type or family from the **Record Type** list.

Note: You can also right-click the record type or family in the Workspace and select **Delete** from the shortcut menu.

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

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

Field Name	Type	Default Value	Permission	Value Changed	Validation	Choice List
record_type	RECORD TYPE					
id	ID					
is_active	INT					
id	ID					
State	STATE					
version	INT					
lock_version	INT					
locked_by	INT					
history	JOURNAL					
is_duplicate	INT					
duplicate_state	SHORT_STRING					
Message	SHORT_STRING					
Description	MULTILINE_STRING					
Priority	SHORT_STRING					
Severity	SHORT_STRING					CONSTANT_LIST
Submitter	REFERENCE	BASIC.PERL				CONSTANT_LIST
Submit_Date	DATE_TIME	BASIC.PERL				DEFAULT
Owner	REFERENCE					DEFAULT
old_id	SHORT_STRING					DEFAULT
Keywords	MULTILINE_STRING					CONSTANT_LIST
Symptoms	Add Field...			BASIC.PERL		CONSTANT_LIST
Note_Entry	Delete Field					
Notes_Log	Rename Field...					
Resolution_Statetype	Field Properties			BASIC.PERL		CONSTANT_LIST
Attachments	What's This?					N/A
Project						DEFAULT
customer_severity	SHORT_STRING					CONSTANT_LIST
customer	REFERENCE_LIST					DEFAULT

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.

Note

- To make a new field available to your users or to hide a deleted field from your users, you must also modify any forms that refer to that field.

- After adding or modifying a field, you must check the schema back into the schema repository and apply this version of the schema to the user database.
- Once you check in your schema, you cannot change the type, size, or DB Name properties of the field. However, you can change the name that you use to refer to the field.

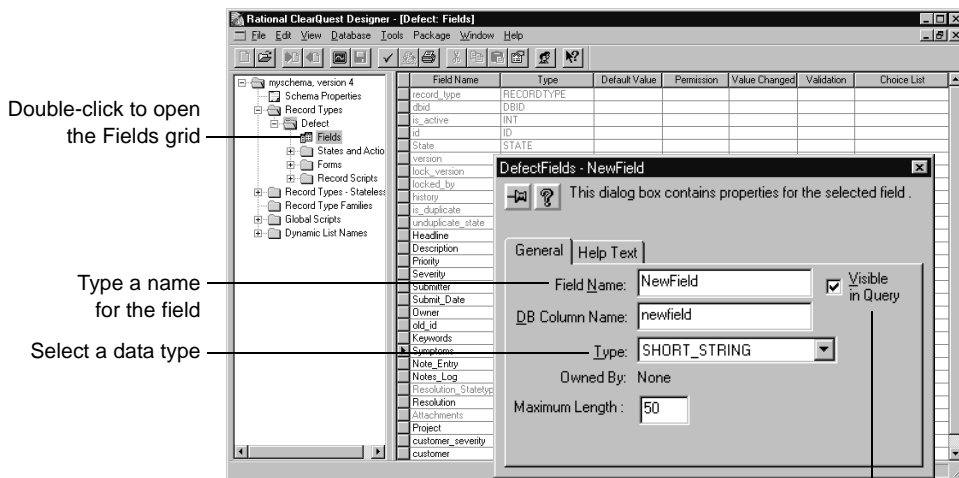
Adding a new field

To add a new field, you add the field to the Fields grid. In addition, you must also:

- Associate the new field with a form control to make the field available to ClearQuest client users. See “Adding controls to a form” on page 98.
- Test the schema, check the schema back into the schema repository, and apply the schema changes to the user database. See “Overview of schema procedures” on page 32.

To add a new field to the Fields grid:

- 1 In the Workspace, expand **Record Types**, then expand the desired record type and double-click **Fields** to display the Fields grid.
- 2 Select **Edit > Add Field**.



Select **Visible in Query** to ensure that the field is included in query runs in the ClearQuest client

3 In the New Field dialog box, type a **Field Name**.

Warning: When naming fields, be sure not to use any vendor database reserved keywords. See your vendor documentation for a list of keywords.

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

4 Select a field data **Type**. The data type you select determines the type of data the user can enter in the field. See, “Selecting a field data type” on page 65.

- For fields of type `SHORT_STRING`, enter a value for **Maximum Length**. The value must be greater than zero, but cannot exceed 254 characters. You cannot modify the maximum length after you check in your schema.

ClearQuest reserves storage space in the database for the maximum length of a field. To save storage space, set the maximum length as low as possible. ClearQuest truncates values that exceed the maximum field length.

- For fields of type `REFERENCE` and `REFERENCE_LIST`, select a record type to refer to. The record type can be state-based or stateless. You can also enter an optional back-reference field to create a link from the referenced record back to this field's record. For more information, see “Using fields to link records” on page 70.

Note: You cannot modify the field **Type** or **DB Column Name** after you check in the schema. For fields whose type is `SHORT_STRING`, you cannot modify the Maximum Length property. To change any of these properties, you must delete the field and create a new field with similar properties.

5 Select the **Visible in queries** checkbox to ensure that this field is included in queries run from ClearQuest client. If you do not want this field included in queries, unselect the **Visible in queries** checkbox.

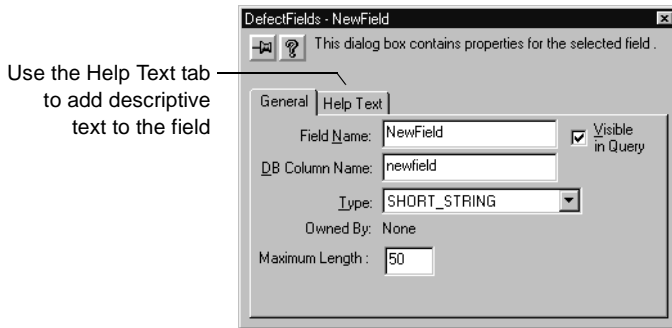
Note: After you add the field to the Fields grid, you must associate the new field with a form control so that it is available to ClearQuest users. See “Adding controls to a form” on page 98.

Adding Help text to a field

You can provide field-level help for ClearQuest client users by adding Help text to a field. Help text can describe the field or provide special instructions on how to use the field. ClearQuest client users can then right-click the field on the record form and select **Help** from the shortcut menu to view the Help text.

To add Help text to a field, use the Help Text tab in the New Field dialog box.

The Help text limit is 254 characters, approximately two-thirds of the space available on the Help Text tab.



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:

Data	Description/Comments
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	A reference to a unique key in a record type. 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 and can specify that the referenced record type is under security control.
REFERENCE_LIST	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 with a parent/child control to link related records. 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. Note: You cannot use the REFERENCE_LIST type when creating a report. Multiple record references within a field will return a report error.
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.
DBID	Reserved for system fields
ID	Reserved for system fields
JOURNAL	Reserved for system fields
STATE	Reserved for system fields

Note: You cannot modify the data type or the DB Column Name 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, but 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. Note: Optional is the default setting for new fields.
Read only	The user can view the contents of the field but cannot modify it. Note: 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:

- 1 In the Workspace, expand **Record Types** and the desired record type.
- 2 Expand **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

record_type	Submitted	Assigned	Opened	Resolved	Closed	Duplicate	Postponed	Default Behavior
id	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY
is_active	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY
id	READONLY	OPTIONAL	READONLY	READONLY	READONLY	READONLY	OPTIONAL	OPTIONAL
State	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY
version	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY
lock_reason	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY
locked_by	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY
history	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY
is_duplicate	READONLY	OPTIONAL	READONLY	READONLY	READONLY	READONLY	OPTIONAL	OPTIONAL
unduplicate_state	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY
Headline	MANDATORY	MANDATORY	MANDATORY	MANDATORY	MANDATORY	MANDATORY	MANDATORY	MANDATORY
Description	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL
Priority	OPTIONAL	MANDATORY	MANDATORY	MANDATORY	MANDATORY	OPTIONAL	OPTIONAL	MANDATORY
Seventy	MANDATORY	M	Read Only	ATORY	MANDATORY	MANDATORY	OPTIONAL	OPTIONAL
Submitter	READONLY	RE	Mandatory	ONLY	READONLY	READONLY	READONLY	MANDATORY
Submit_Date	READONLY	RE	Optional	ONLY	READONLY	READONLY	READONLY	READONLY
Owner	OPTIONAL	M	Use Hook	ATORY	MANDATORY	MANDATORY	OPTIONAL	MANDATORY
oid_id	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY
Keywords	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL
Symptoms	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL
Note_Entry	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL
Notes_Log	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY

Set the field behavior to "Mandatory"

Line: Col: Mode: Editable

- 3 In the Behaviors grid, right-click in the state column of the field you want to define and select a desired behavior for the field.

Defining default field behavior

You can use the Default Behavior column in the Behaviors grid to define the 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.

To define the default behavior for a field:

In the Behaviors grid, click the Default Behavior column of the desired field and select a behavior to use as the default.

Double-click to open the Behaviors grid

Click the Default Behavior column of the desired field and select a default behavior

	Submitted	Assigned	Opened	Resolved	Closed	Duplicate	Postponed	Default Behavior
record_type	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY
uuid	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY
is_active	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY
id	READONLY	OPTIONAL	READONLY	READONLY	READONLY	READONLY	OPTIONAL	OPTIONAL
date	READONLY	OPTIONAL	READONLY	READONLY	READONLY	READONLY	OPTIONAL	OPTIONAL
version	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY
lock_version	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY
locked_by	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY
history	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY
is_duplicate	READONLY	OPTIONAL	READONLY	READONLY	READONLY	READONLY	OPTIONAL	OPTIONAL
unduplicate_date	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY
Headline	MANDATORY	MANDATORY	MANDATORY	MANDATORY	MANDATORY	MANDATORY	MANDATORY	MANDATORY
Description	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL
Priority	OPTIONAL	MANDATORY	MANDATORY	MANDATORY	MANDATORY	OPTIONAL	OPTIONAL	MANDATORY
Severity	MANDATORY	No	Read Only	MANDATORY	MANDATORY	OPTIONAL	OPTIONAL	MANDATORY
Submitter	READONLY	RE	Mandatory	ONLY	READONLY	READONLY	READONLY	READONLY
Submit_Date	READONLY	RE	Optional	ONLY	READONLY	READONLY	READONLY	READONLY
Owner	OPTIONAL	N/	Use Hook	JATORY	MANDATORY	MANDATORY	OPTIONAL	MANDATORY
old_id	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY
Keywords	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL
Symptoms	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL
Note_Entry	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL
Notes_Log	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY

Set the field behavior to "Mandatory"

Line: Col: Mode: Editable

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.

Modifying a field

You can modify some field properties using the field's Properties dialog box, but other properties must be modified directly from the fields grid.

Note: You cannot modify the field data type or DB Column Name after you check in the schema. For fields whose type is `SHORT_STRING`, you cannot modify the Maximum Length property. To change any of these properties, you must delete the field and create a new field with similar properties.

To modify a field:

- 1 In the Workspace, expand **Record Types**, then expand the desired record type and double-click **Fields** to display the Fields grid.
- 2 Right-click the field you want to modify and select **Field Properties** from the shortcut menu. (You can also select the field and select **Edit > Field Properties**.)

Changing the name of a field

Note: 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:

- 1 Right-click the field and select **Rename Field** from the shortcut menu.
- 2 Type a new **Field Name**.

When naming fields, be sure not to use any vendor database reserved keywords. See your vendor documentation for a list of keywords.

Deleting a field

The following restrictions apply to deleting fields:

- 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 and then delete that field, you must also remove references to the deleted field from the script code.
- In the ClearQuest client, any queries that use a deleted field will become invalid.

To delete a field you must first delete the field from the Fields grid and then delete the field from the record form:

- 1 Display the Fields grid.
- 2 Right-click the field and select **Delete Field** from the shortcut menu.
- 3 Delete the field from the record form. See “Deleting a control from a form” on page 104.

Using fields to link records

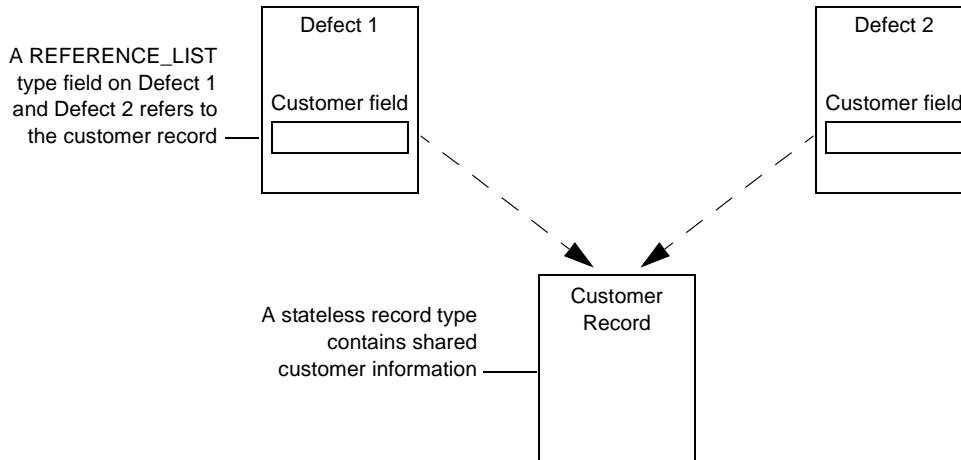
You can use fields to link records of the same type or records of different types. You can link records to:

- Share common data
- Create a parent-child hierarchy

Linking records to share common data

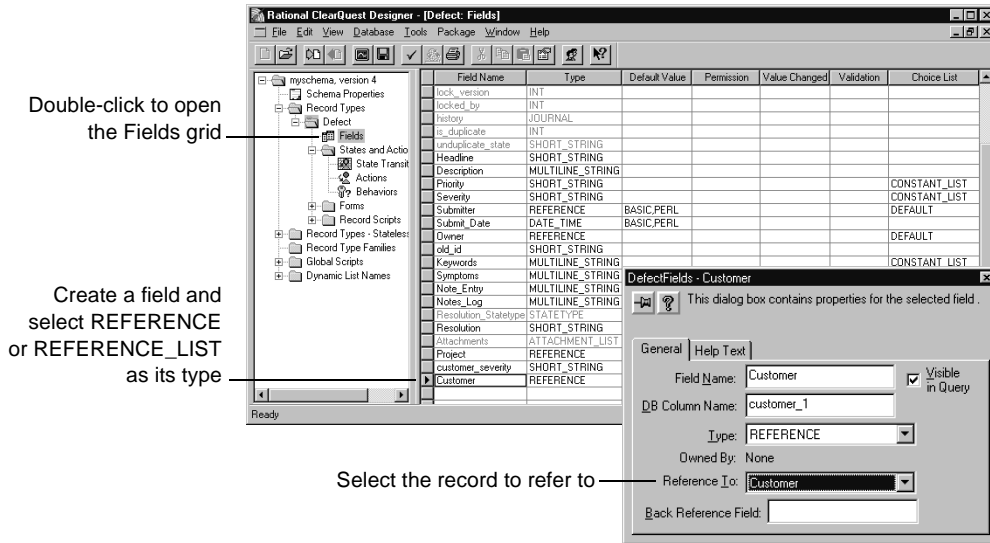
You can use REFERENCE or REFERENCE_LIST type fields to link records in order to share common data. Use a REFERENCE field to link one record; use a REFERENCE_LIST field to link multiple records.

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



To link records to share common data:

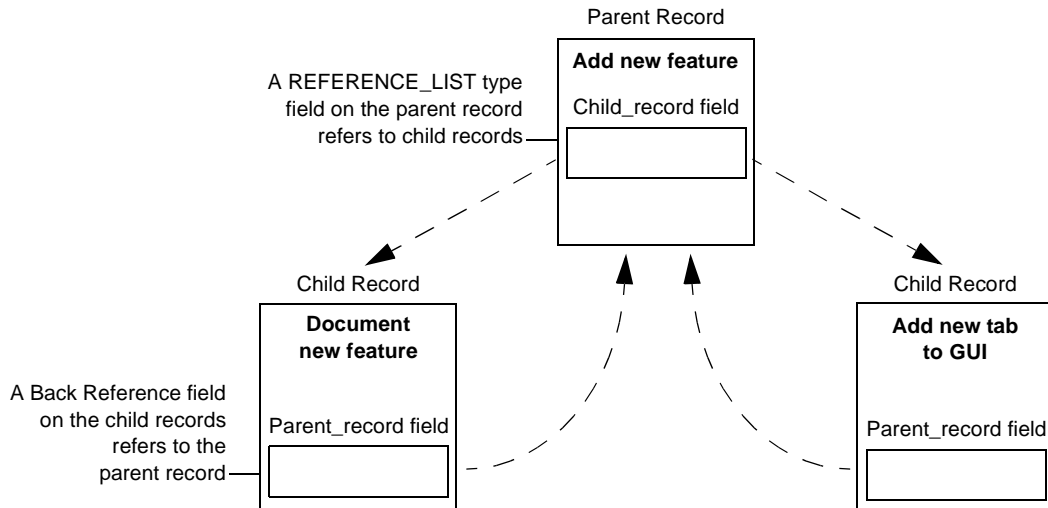
- 1 Display the Fields grid for the record type and create a new field, selecting REFERENCE or REFERENCE_LIST as its type.
- 2 Right-click the new field and select **Field Properties** from the shortcut menu.
- 3 In the Field Properties dialog box, select the record type to refer to from the **Reference To** list.



- 4 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 controls to a form” on page 98.

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 Right-click the field and select **Field Properties** from the shortcut menu.

- 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.

The screenshot shows the Rational CleanQuest Designer interface. On the left, a tree view shows a schema named 'myschema, version 4' with a 'Record Types' folder containing a 'Detect' record type. The 'Fields' folder under 'Detect' is expanded, showing a list of fields. A 'Parent_record' field is highlighted in the list, and its type is 'REFERENCE_LIST'. A dialog box titled 'DefectFields - Child_record' is open, showing the configuration for this field. The 'Field Name' is 'Child_record', the 'DB Column Name' is 'child_record', and the 'Type' is 'REFERENCE'. The 'Reference To' is 'Defect', and the 'Back Reference Field' is 'Parent_record'. The 'Visible in Query' checkbox is checked.

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.

Field Name	Type	Default Value	Permission	Value Changed	Validation	Choice List
is_duplicate	INT					
is_duplicate_state	SHORT_STRING					
Headline	SHORT_STRING					
Description	MULTILINE_STRING					
Priority	SHORT_STRING					CONSTANT_LIST
Severity	SHORT_STRING					CONSTANT_LIST
Submitter	REFERENCE	BASIC.PEPL				DEFAULT
Submit_Date	DATE_TIME	BASIC.PEPL				
Owner	REFERENCE					DEFAULT
old_id	SHORT_STRING					
Keywords	MULTILINE_STRING					CONSTANT_LIST
Symptoms	MULTILINE_STRING					CONSTANT_LIST
Note_Entry	MULTILINE_STRING					CONSTANT_LIST
Notes_Log	MULTILINE_STRING					BASIC.PEPL
Resolution_StateType	STATE_TYPE					
Resolution	SHORT_STRING					
Attachments	ATTACHMENT_LIST					
Project	REFERENCE					
customer_severity	SHORT_STRING					
Customer	REFERENCE					
Child_record	REFERENCE					
Parent_record	REFERENCE_LIST					

- Add the new fields to the record forms. Use a parent/child control with a REFERENCE_LIST field type. See “Adding controls to a form” on page 98.

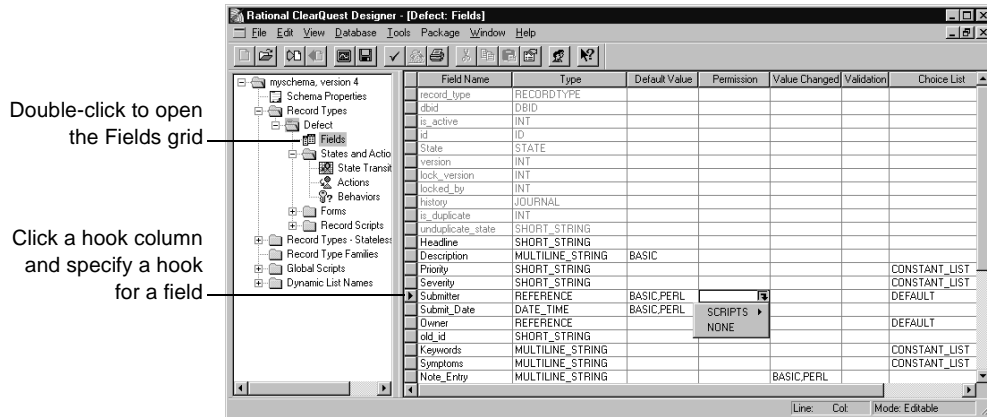
For more information, see “Action hook for setting the value of a parent record” on page 212 of this guide.

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. See, “Validating user input in a field” on page 185.

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

To define a field hook, use the Fields grid.



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.

See also

For a complete description of field hooks and information about how they work with action hooks, see Chapter 8, “Using hooks to customize your workflow.”

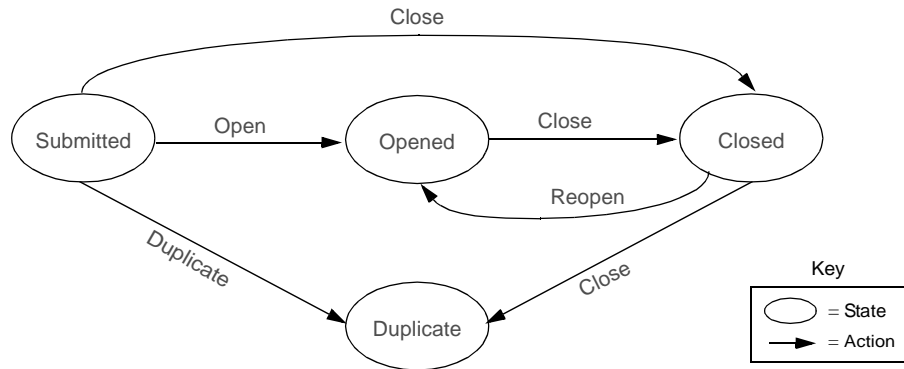
- “Working with field hooks” on page 175
- “Creating a dependency between fields” on page 178
- “Creating a choice list for a field” on page 180
- “Defining default field behavior” on page 67
- “Validating user input in a field” on page 185

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.



The best way to begin designing a state model is by listing and describing all the states you want to have for a specific record type. For example, the table below 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.

For example, to display the State Transition Matrix for the Defect record type:

- In the Workspace, expand **Record Types > Defect > States and Actions** and double-click **State Transition Matrix**.

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		Reject	Re_open
Resolved			Resolve		
Closed	Close	Close		Validate	
Duplicate	Duplicate		Duplicate	Duplicate	Duplicate
Postponed	Postpone	Postpone	Postpone		

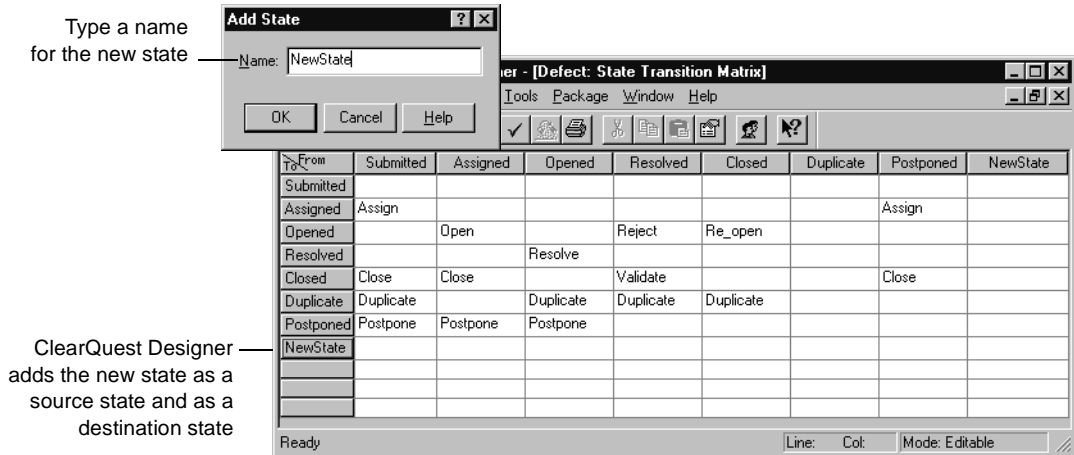
In the State Transition Matrix, source states are listed in columns (From), and destination states are listed in rows (To). The action necessary to move from a particular source state to a destination state is listed at the intersecting cell. If there is no action listed in the cell, a state transition 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, you first add the state to the State Transition Matrix, then create a state transition:

- 1 In the Workspace, expand **Record Types** > <the record type> > **States and Actions**, then double-click **State Transition Matrix**.
- 2 Select **Edit > Add State**.
- 3 In the Add State dialog box, type a name for the new state and click **OK**.



ClearQuest Designer automatically adds the new state as a source state and as a destination state in the row and column headers of the State Transition Matrix.

- 4 Next, you must create a state transition that defines how this new state will be used in your state model. See “Creating a state transition” on page 84.

Note

- Once you add a state to 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” on page 78.

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.

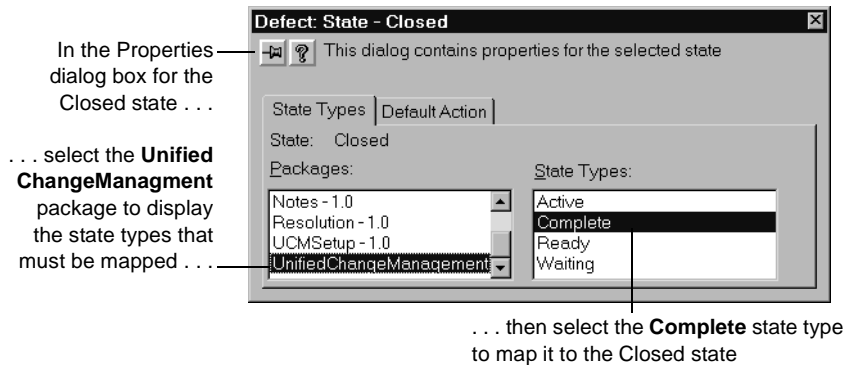
To ensure that a package with state types works properly with your schema, you need to map each state in each record type to an appropriate package state type. You can map more than one state to a state type, but each state type requires at least one state.

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 Display the State Transition Matrix for the desired record type.
- 2 Right-click a state and select **Properties** to display the Properties dialog box 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:



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

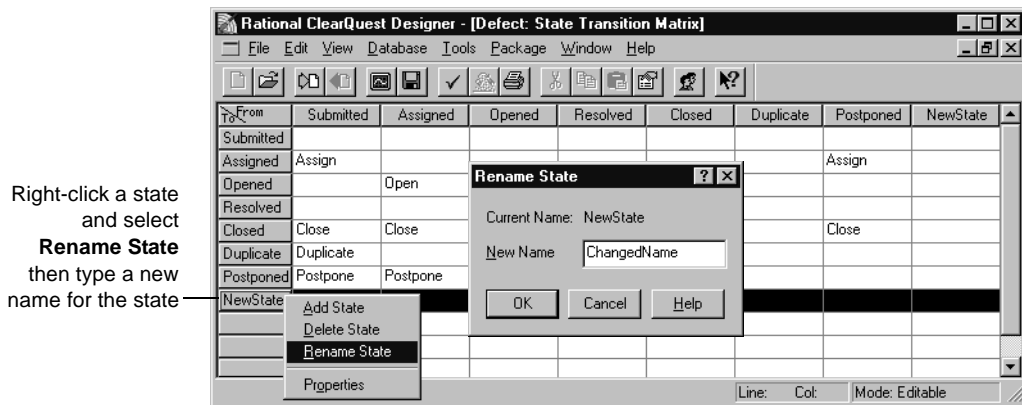
Note: If you are using the UCM schema or package, you must also assign default actions for your states; see “Setting the default actions for UCM” on page 287.

Changing the name of a state

You can change 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:

- 1 In the Workspace, expand **Record Types** > <the record type> > **States and Actions**, then double-click **State Transition Matrix**.
- 2 In the State Transition Matrix, click the state that you want to rename and select **Edit > Rename State** (or right-click and select **Rename State** from the shortcut menu).



- 3 In the Rename State dialog box, type a **New Name** and click **OK**.

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.

Deleting a state

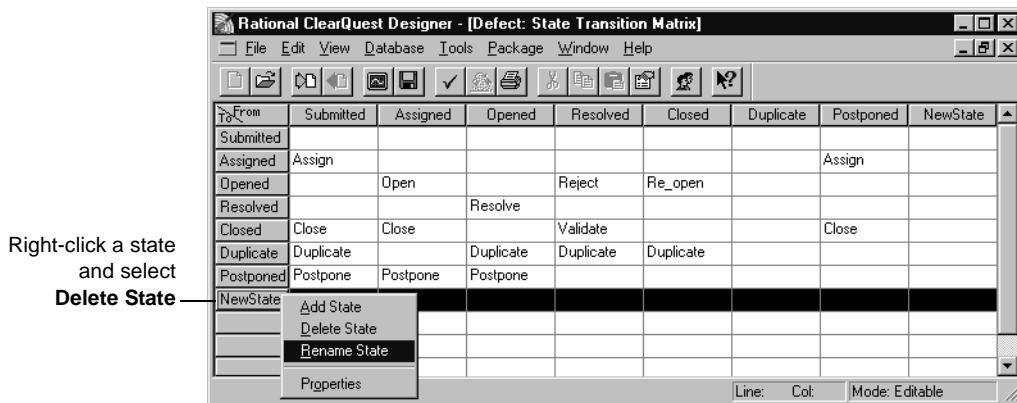
You should delete any states that you do not plan to use. It is a validation error to define a state that cannot be reached by one or more actions.

Before deleting a state, keep in mind the following:

- If you delete a state, you must edit any actions that refer to that state. ClearQuest does not reassign the source or destination states of an action.
- If you explicitly refer to a state in a script, you must modify the script to remove any references to the state.
- Do not delete a state if you plan to upgrade a database that currently uses that state. ClearQuest will not let you upgrade the database if there are records that use the state.

To delete a state:

- 1 In the Workspace, expand **Record Types** > <the record type> > **States and Actions**, then double-click **State Transition Matrix**.
- 2 In the State Transition Matrix, click the state that you want to rename and select **Edit > Delete State** (or right-click and select **Delete State** from the shortcut menu).



Working with actions and action types

After defining your states, you are ready to define the actions that ClearQuest client users will use to submit new records to the database, to modify or delete records, and to move records from one state to another.

Understanding actions and action types

Actions are how ClearQuest client users submit new records to the database, move records from one state to another, and modify or delete records. Two of the most common actions used in ClearQuest are the Submit action and the Change State action. ClearQuest includes predefined action types that allow you to work with your database records. See “Supported action types” on page 83.

The ClearQuest client displays action names in a drop-down list attached to the **Actions** button of a record's form and in the **Actions** menu. As the ClearQuest administrator, you can choose a default action for a state. In ClearQuest, the default action appears in bold prompting the user through your process. Also, a default action can be called from a hook that uses a script.

An action can:

- Create a new record and add it to the database.
- Modify information in the record. (The behaviors associated with each field can also limit access to particular fields of the record.)
- Move a record from one state to another.
- Mark one record as a duplicate of another.
- Execute a hook. Action hooks can handle access control, initialization, validation, and notification. See “Customizing actions by adding hooks” on page 86.
- Delete a record from the database.
- You can customize which users have access to specific actions, as well as when actions can be performed.

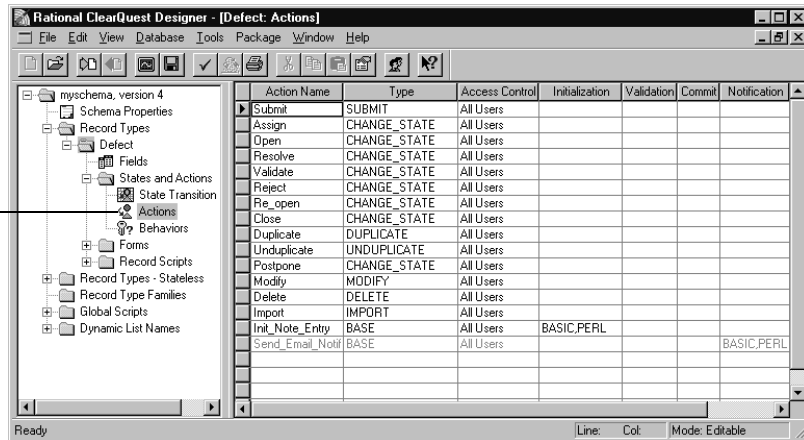
In ClearQuest Designer, each record type has an Actions grid that defines the actions available for records of that type. You can use the Actions grid to add, modify, and delete actions, and to create state transitions.

To display the Actions grid:

- 1 In the Workspace expand **Record Types** or **Record Types - Stateless**, then expand the desired record type.
- 2 For state-based record types, expand **States and Actions**. 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



Supported action types

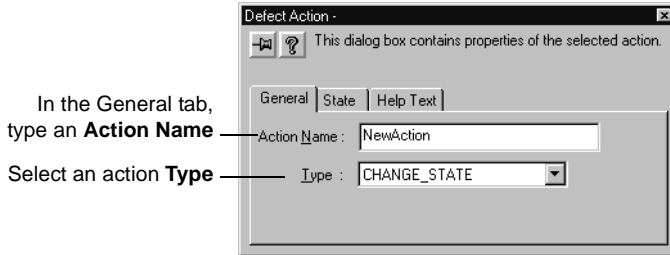
ClearQuest supports the following types of actions:

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>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>
Import	<p>Allows ClearQuest to import records from another source.</p> <p>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.</p> <p>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>Available for state-based record types. Removes the link between duplicate records.</p>

Adding a new action

To add a new action:

- 1 Display the Actions grid.
- 2 Select **Edit > Add Action**.
- 3 In the General tab of the Actions Properties dialog box, type an **Action Name**.



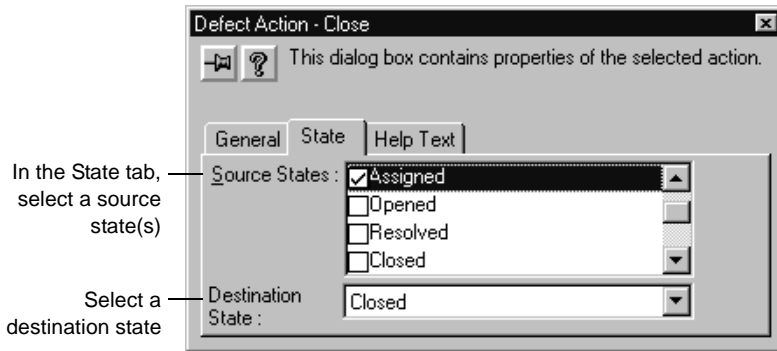
- 4 Select an action Type. You must select a type for the action. If you plan to use this action to initiate a state transition, select **CHANGE_STATE** as the type.

Creating a state transition

As the ClearQuest administrator, you define the rules for how users move records between states. State transitions are implemented by actions. 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.
In the Workspace expand **Record Types > < the record type > > States and Actions**, then double-click **Actions**.
- 2 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**.

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



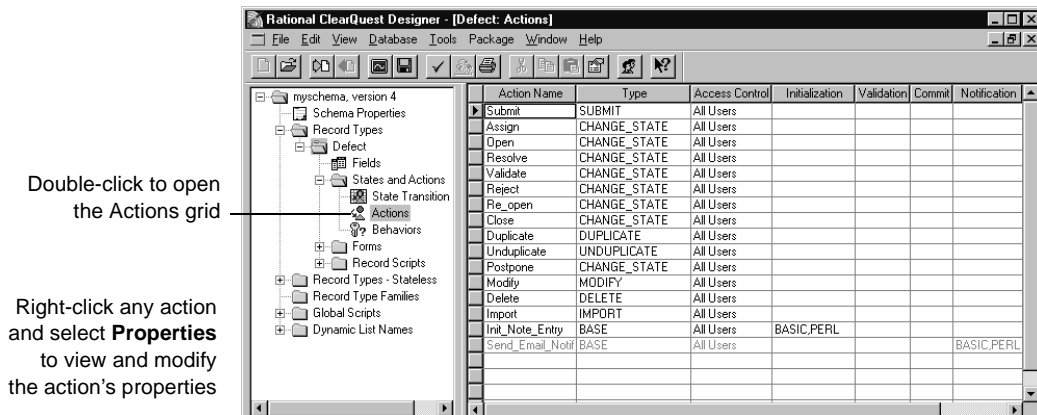
Each CHANGE_STATE action must have at least one source state and a destination state. If none are defined, ClearQuest reports an error during validation.

- 4 After creating the state transition, display the State Transition Matrix to verify that ClearQuest applied the transitions to the states. See “Using the State Transition Matrix” on page 76.

Modifying actions

To modify an action:

- 1 In the Workspace expand **Record Types** or **Record Types - Stateless**.
- 2 Expand the desired record type.
- 3 For state-based record types, expand **States and Actions**. Double-click **Actions**.

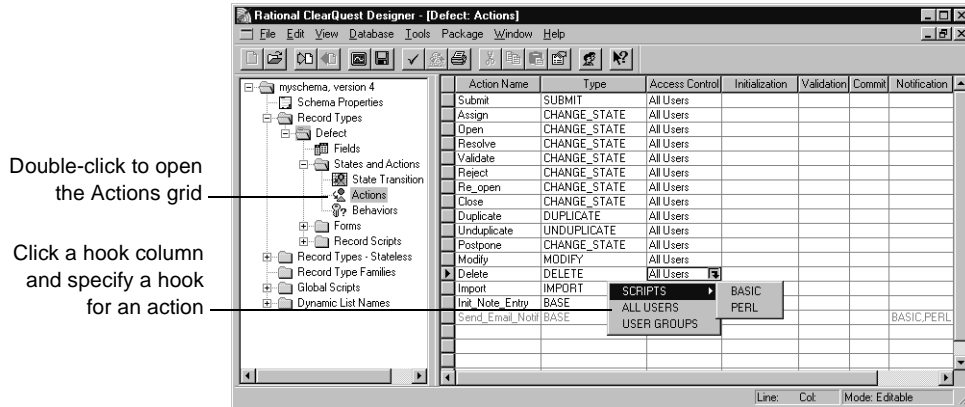


Customizing actions by adding hooks

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.

For a description of action hooks and information about how they work with field hooks, see Chapter 8, “Using hooks to customize your workflow.” To learn how to create an access control action hook, see “Restricting user access to actions” on page 151. Also see “Selecting a scripting language” on page 38.

Using default actions

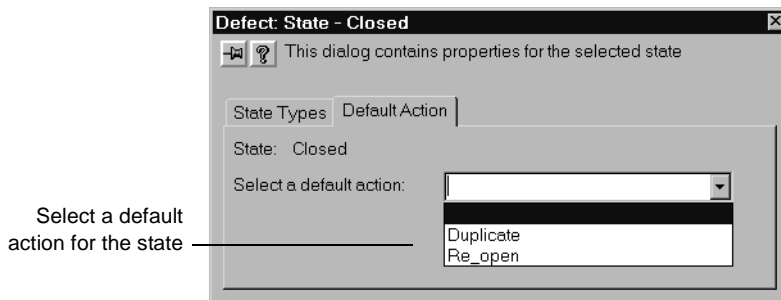
You can define default actions for states. A default action for a state appears in bold in the ClearQuest client **Actions** menu. Default actions:

- Are useful for guiding users through your state model.
- Are required for certain schemas and packages, such as the UCM schema and package. 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 the default actions for UCM” on page 287.
- Can be called from hook code.

Note: Before you can define the default action for a state, you must first create a state transition. See “Creating a state transition” on page 84.

To define the default action for a state:

- 1 Display the State Transition Matrix for the record type.
- 2 In the State Transition Matrix, right-click the state and select **Properties** from the shortcut menu.
- 3 In the Default Action tab of the State Properties dialog box, select a default action for the state.



Deleting an action

Deleting an action might require other changes to a schema. For example, if you delete a `CHANGE_STATE` action, you may need to modify the State Transition Matrix to compensate for the lost action.

To delete an action:

- 1 Open the Actions grid.

In the Workspace expand **Record Types** > < the record type> > **States and Actions**, then double-click **Actions**.

Or, expand **Record Types - Stateless** > < the record type> and double-click **Actions**.

- 2 Select the row of the action you want to delete.
- 3 Select **Edit** > **Delete Action** (or right-click and select **Delete Action** from the shortcut menu).

Note: If you have explicitly referred to a deleted action in a script, you also need to modify your script to delete any references to the action.

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.

This chapter describes how to create and modify forms. The topics covered include:

- Using the Forms window
- Working with forms
- Adding tabs to a form
- Working with form controls
- Creating forms for multiple platforms
- Creating forms to view and modify imported data
- Reusing forms
- Form controls reference

For detailed descriptions of the properties for each control type, see “Form controls reference” on page 111.

Using the Forms window

You use the Forms window to create and customize forms.

To display the Forms window:

- 1 In the Workspace, expand the **Record Types** or **Record Types - Stateless** folder.
- 2 Expand the record type.
- 3 If the record type has an existing form, expand the **Forms** folder and double-click the form. If the record type does not have a form, right-click the Forms folder and select **Add** to add a form.

The Forms window includes a Control palette, a Field List, and a Form toolbar.

- Use the Control palette to add controls to the form. Select appropriate controls for the data you expect users to enter in the ClearQuest database.
- Use the Field List to add a selected field to the form. ClearQuest adds a control that is appropriate for the field type.
- Use the Form toolbar to align, resize, and center controls on a form.

You can hide and display these parts of the window using commands on the View menu. When viewing a form, the Edit and View menus change and new menus appear that apply to the form.

Working with forms

You must define at least one record form for each record type you expect your users to display or modify. You can also define a Submit form for each record type. Every form, whether a Record form or a Submit form, has at least one tab; you can add additional tabs. Records can be created without forms, but only using the ClearQuest API or the Import Tool.

Creating a new form

When you create a form, ClearQuest automatically adds three controls to the form: an Apply button, a Revert button, and an Actions button. ClearQuest uses the buttons to initiate and manage actions. You cannot delete these buttons, but you can change their properties, such as the label on the button (for example, you can change the label on the Apply button to OK and the label on the Revert button to Cancel).

To create a form:

- 1 In the Workspace, expand **Record Types** or **Record Types - Stateless** and the desired record type.

- 2 Right-click the **Forms** folder and select **Add** from the shortcut menu.

ClearQuest displays a new empty form in the right pane 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.

The name must be between 1 and 25 characters and contain only upper- and lowercase letters, numbers, and underscores.

- 4 Right-click the new form name and select **Record Form** or **Submit Form**.

After creating a form, you can begin adding controls to it. If the record that owns the form contains too many fields to display on one tab or if you want to group related fields on separate tabs, you can add tabs to the form. Every form must have at least one tab.

Note: Font and language differences can cause forms to look different on the various ClearQuest Windows, UNIX, and Web clients. Be sure to test the form on the specific clients you are using and make any necessary changes.

Changing the name of a form

To change the name of a form:

- In the Workspace, right-click the form name and select **Rename** from the shortcut menu.

Changing the size of a form

You can change the size of a form to optimize the space used by the controls. Changing the proportions of the form also changes the size and position of the tabs and the default buttons on the form.

If a form requires more controls than can be accommodated on the main tab, you can add additional tabs to the form. See “Adding tabs to a form” on page 93.

To change the size of a form:

- 1 Expand **Record Types** or **Record Types - Stateless** in the Workspace, until you see the form you want to resize.
- 2 Double-click the form to display it.
- 3 Click a corner of the form window and drag it to the desired shape and size.

To set the width and height of the form to specific values, edit the property sheet of the form:

- 1 Right-click in a tab on the form and select **Form Properties** from the shortcut menu.
- 2 In the Form Property Sheet, enter the width and height values, then click **OK**.

Changing the font on forms

You can change the font and font properties for all forms in a schema. Font changes apply to all text on all tabs, in every form in the schema, including labels for controls and tab names.

To change the font and font properties for all forms in a schema:

- 1 In the Workspace, right-click the form and select **Fonts** from the shortcut menu.
- 2 In the Font dialog box, select the desired font and font properties.
- 3 Close the form and then re-open it to display the new font.

Deleting a form

If you no longer need a form, you can delete it; however, if a record type has only one form, that form must be a record form. You cannot delete the last form of a record type.

To delete a form:

- 1 In the Workspace, expand **Record Types** or **Record Types – Stateless**, then expand the desired record type and expand the **Forms** folder.
- 2 Right-click the form and select **Delete** from the shortcut menu.

Adding tabs to a form

Tabs allow you to organize controls into groups. If your form has more controls than can be easily displayed on the main tab, you can add additional tabs and use them to group the controls on the form. You can add as many tabs as you need.

To add a tab to a form:

- 1 Expand **Record Types** or **Record Types – Stateless**, in the Workspace until you see the desired form.
- 2 Double-click the form to display it.
- 3 Select **Edit > Add Tab**.

After adding a tab, you can change the tab name (and specify an access key for the tab), restrict user access to the tab, change the tab position, delete unused tabs, and copy tabs.

Changing the name of a tab

You can rename the tab and include an access key so that users can display the tab by pressing the corresponding key on the keyboard. Access keys are underlined in the tab name.

To change the name of a tab:

- 1 Right-click the tab and select **Tab Properties** from the shortcut menu, or double-click a blank portion of the tab to display the shortcut menu.
- 2 In the Tab Properties dialog box, type the new name in the Tab Caption text box. To specify an access key for the tab, type an ampersand (&) before any letter in the tab name.

The access key allows users to display the tab by typing the access key. For example, to specify the letter T as the access key for a tab named Attachments, type: `A&ttachments`

Restricting access to a tab

By default, a tab is visible to all users, but you can restrict access to a tab so that it is visible to only users in selected groups.

To restrict access to a tab:

- 1 Right-click the tab on the form and select **Tab Properties** from the shortcut menu, or double-click a blank portion of the tab.
- 2 In the Tab Properties dialog box, do any of the following:
 - To display the tab to all users, select **All Users**.
 - To display the tab to users in selected groups, clear the **All Users** check box, select the groups in the **Available Groups** list, and click **Add**.
 - To remove a group's access to the tab, select the group in the **Selected Groups** list and click **Remove**.

Changing the order of tabs

By default, ClearQuest displays tabs on the form in the order in which you create them. You can change the tab order by assigning a new index number to a tab. The index number identifies the order of the tab on the form; the first tab has index number 0, the second tab has index number 1, and so on.

To change the order of tabs, you must change the index number of each tab individually, then close and re-open the form to see your changes:

- 1 Open the form and click the tab whose order you want to change.
- 2 Right-click the tab and select **Tab Properties** from the shortcut menu.
- 3 In the Tab Properties dialog box, select the desired index number for the tab.

Select 0 for the tab you want to display first on the form; select 1 for the second tab, and so on.

- 4 Close the form and re-open it to see the change.

Deleting tabs

You can delete a tab from a form. Deleting a tab also deletes all of the controls on that tab.

Warning: Deleting a tab cannot be undone.

To delete a tab:

- 1 Expand **Record Types** or **Record Types – Stateless** in the Workspace until you see the desired form.
- 2 Double-click the form to display it.
- 3 Click the tab and select **Edit > Delete Tab**.

Copying tabs

You can copy a tab, including all of its controls, and paste it into another tab in the same form or into a different form. After pasting the tab into a form, you can modify the tab as you normally would.

To copy a tab:

- 1 Open the form and display the tab that you want to copy.
- 2 Select **Edit > Copy Entire Dialog to Clipboard**.

To paste the tab:

- 1 Click the tab you want to paste the controls.
- 2 Select **Edit > Paste**, or right-click in the tab and select **Paste** from the shortcut menu.
- 3 Edit the tab as desired.

Working with form controls

You use controls to display fields on the form. ClearQuest Designer 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, the list-view control and the parent/child control can only 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 Designer also provides an ActiveX control that you use to incorporate any registered ActiveX control into a form. For example, you might use an ActiveX control to interact with an external database. Before using this control, you should be familiar with ActiveX functionality and how to register your controls.

Note: You can use the ClearQuest API to submit, view, and edit records programmatically instead of through forms.

ClearQuest supports the following form controls:

Form control	Description
ActiveX	Incorporates any registered ActiveX control into a form. You write the initialization record script and the action record script.
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 Properties.
Combo Box	Combines an editable text field with a list box.
Drop-down List Box	Displays a list of possible values for a particular field.
Drop-down Combo Box	Combines an editable text field with a drop-down list box.
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	Visually groups together one or more other controls.
History	Displays information about the actions that have been applied to a record.

Form control	Description
List Box	Displays a list of possible values for a particular field. List boxes include an additional control for selecting one or more items from a choice list.
List View	Allows you to display the records associated with a field of the REFERENCE_LIST type. Displays the associated reference list in a multi-column format.
Option Button	Option button controls are used in groups to represent a set of mutually exclusive choices. Allows the selection of only one option in a group.
Parent/child	Allows you to set up a form to link associated records. Used with REFERENCE_LIST field type. The parent/child control consists of both a list view control and three push buttons. The list view control and push buttons are automatically associated using a unique list view ID. If you change the ID of the list view, you must also update the push buttons.
Picture	Allows you to 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.

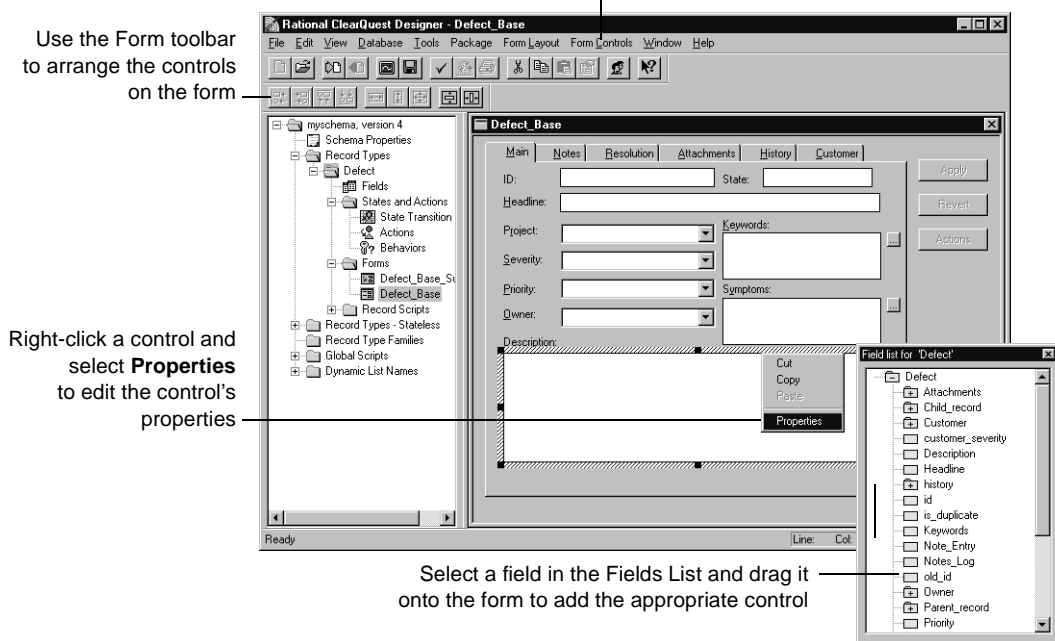
For detailed descriptions of the properties for each control type, see “Form controls reference” on page 111.

Adding controls to a form

Before you can add a field to a form, you must first add the field in the Fields grid. See “Adding a new field” on page 62.

You can add controls to a form using the Control palette, the Field List, or the Form Controls menu. The Control Palette provides a visual cue as to the type of control you are adding. The Field List automatically creates the control that is appropriate for the type of field you have chosen. The Form Controls menu allows you to create the same controls as the Control palette; in addition, it contains a command to create ActiveX controls.

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



Note: You can add a field to a form more than once, but all instances of the field must have the same value.

Opening the form

To open a form for editing:

- 1 Expand **Record Types** or **Record Types – Stateless** in the Workspace until you see the desired form.
- 2 Double-click the form to display it.

The form opens with the Control Palette and Field List displayed. If they are not visible, select **View > Control Palette** or **View > Field List**.

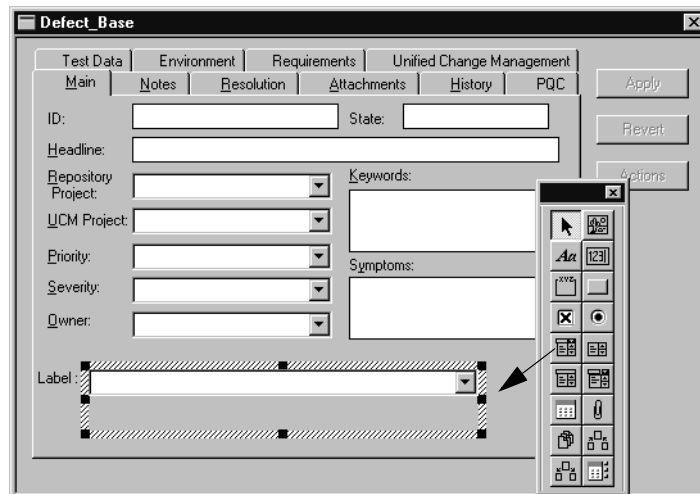
Adding a control using the Control Palette

If the Control Palette is not visible when you open the form, select **View > Control Palette** to display it.

To add a control using the Control palette:

- 1 In the Control palette, click the control you want to add.
- 2 Click where you want to place the control on the form and drag until the control is the desired size.

Once the control is on the form, ClearQuest Designer changes from the current tool to the selection arrow so you can move and size the control as desired.



Click a control in the Control Palette, click where you want the control on the form, then drag the control to the desired size

- 3 After you add the control to the form, you must associate an existing field with the new control. Double-click the control to open its properties sheet, then select the field to associate with the control.

You can also set the display options of the control if desired. See “Editing control properties” on page 103.

Adding a control using the Field List

The advantage of using the Field List to add a control to a form is that when you select a field from the list and drag it to the form, ClearQuest automatically adds the appropriate control for the field type you select. You do not have to edit the control properties to associate the field with the control.

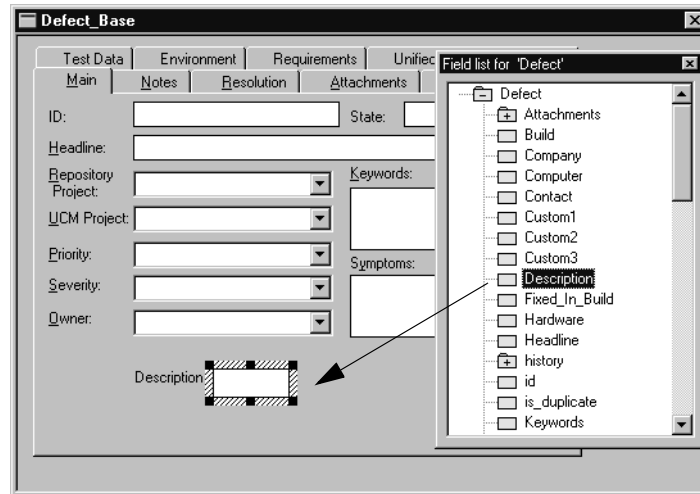
The following table lists the default control that ClearQuest Designer creates for each field type.

Field type	Default control
Attachment List	Attachment
Constant List Choice List	Drop-down List Box
Date-Time	Text Box
Integer	Text Box
Multiline String	Text Box
Reference List	Parent/Child
Reference List Constant	List View
Reference Choice List=Default	Drop-down List Box
Short String Constant List	Drop-down List Box
Short String	Text Box

To add a control using the Field List:

- 1 If the Field List is not visible when you open the form, select **View > Field List** to display it.
- 2 Drag a field from the Field List to a tab.

Once the control is on the form, ClearQuest Designer changes from the current tool to the selection arrow so you can move and size the control as desired.



Select a field in the Field List and drag it to the desired location on the form. ClearQuest Designer automatically creates the appropriate control.

- 3 After adding the field to the form, you can edit the control properties to set the display options of the control if desired. Double-click the control to open its properties sheet. See “Editing control properties” on page 103.

Adding a control using the Form Controls menu

To add a control using the Form Controls menu:

- 1** Choose the type of control you want to add from the Form Controls menu.
- 2** Click where you want to place the control and drag until the control is the desired size.

Once the control is on the form, ClearQuest Designer changes from the current tool to the selection arrow so you can move and size the control as desired.

- 3** After you add the control to the form, you must associate an existing field with the new control. Double-click the control to open its properties sheet, then select the field to associate with the control.

You can also set the display options of the control if desired. See “Editing control properties” on page 103.

Selecting controls in a form

Use the Selection tool to select one or more controls and to move or change the size of controls.

To select one or more controls in the form:

- 1** Make sure that the Selection tool is selected in the Control palette.
- 2** Click a control to select it. To select additional controls, hold down the Shift key as you click.

Editing control properties

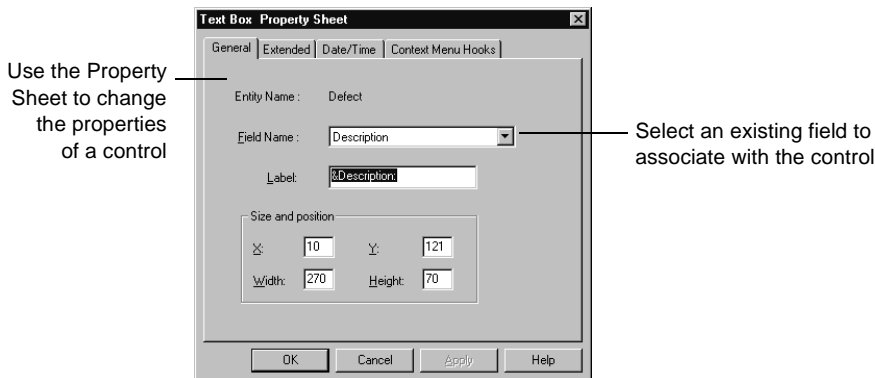
Note: If you use the Control Palette or the Form Controls menu to add a new control to a form, you must then edit the control properties to associate that control with an existing field.

After adding a control to a form, you can edit the properties of the control, including selecting the field to associate with the control and specifying display attributes for the control. Additionally, you can assign record hooks to some controls to allow the user to perform specific tasks using that control.

If you want dependent fields to be enabled in ClearQuest web, you must specify the field on which the dependency is based. To do this, use the Web Dependent Fields tab of the Control Properties sheet.

To edit the properties of a field control:

- 1 Right-click the control and select **Properties** from the shortcut menu.



- 2 Edit the control properties as desired. You can modify the name of the field and its label, the data formats, context menu hooks, and whether or not the field uses horizontal or vertical scrolling. The available options depend on the type of field and control.

For detailed descriptions of the properties for each control type, see “Form controls reference” on page 111.

Deleting a control from a form

If you no longer need a control, you can delete it from the form.

Note: Deleting a control removes the control and its label from the form but does not remove the associated field from the schema. To entirely remove the field from the schema, you must delete the field from the Fields grid. See “Deleting a field” on page 69.

To delete a control from a form:

- 1 Expand **Record Types** or **Record Types – Stateless** in the Workspace until you see the desired form.
- 2 Double-click the form to display it.
- 3 Select the control or controls to be deleted.
- 4 Select **Edit > Delete**, or press the Delete key.

Changing the size and location of controls

To change the size and location a control, you can select the control and drag it to a new location on the form or you can use the tools on the Form toolbar to adjust the position of one or more controls.

The Form Layout menu provides the same tools as the Form toolbar, plus some additional tools. The table below lists each tool and its function.

Note: The align and same-size tools use the first control you select as the basis for modifying the other controls.

Layout Tool	Description
Align left	Aligns the left edge of each control with the left edge of the control that was selected first.
Align right	Aligns the right edge of each control with the right edge of the control that was selected first.
Align top	Aligns the top edge of each control with the top edge of the control that was selected first.
Align bottom	Aligns the bottom edge of each control with the bottom edge of the control that was selected first.
Align vertical centers	Aligns the vertical center of each control with the vertical center of the control that was selected first.
Align horizontal centers	Aligns the horizontal center of each control with the horizontal center of the control that was selected first.

Layout Tool	Description
Space evenly across	Arranges the selected controls horizontally so that there is an equal amount of space between all controls. The left-most and right-most controls are not moved.
Space evenly down	Arranges the selected controls vertically so that there is an equal amount of space between all controls. The top-most and bottom-most controls are not moved.
Center vertically in dialog	Aligns the horizontal center of each control with the horizontal center of the tab.
Center horizontally in dialog	Aligns the horizontal center of each control with the horizontal center of the tab.
Make same width	Makes each control the same width as the control that was selected first.
Make same height	Makes each control the same height as the control that was selected first.
Make same width and height	Makes each control the same width and height as the control that was selected first.
Size to content	Adjusts the size of each selected control so that its entire contents can be viewed. This is useful for minimizing the size of a Static Text control while still maintaining the readability of the string.

Moving controls

To move one or more selected controls, do any of the following:

- Press Shift+Arrow key.
- Drag the control or controls to a new location.

Aligning controls

You can align controls to one another.

To align controls:

- 1 Select a control.

All other controls that you select will align with the first control that you select.

- 2 Shift-click additional controls and select an alignment command from the Form Layout menu or click an alignment button on the toolbar.

Resizing controls

You can resize individual controls, or make two or more controls the same height or width.

To resize an individual control:

- Drag a corner of the control, or select **Form Layout > Size to Content**.

To make controls the same size:

- 1 Select a control.

All other controls that you select will be the same size as the first control that you select.

- 2 Shift-click additional controls and select a **Size** command from the **Form Layout** menu or click a size button on the toolbar.

Changing the tab order of controls

The tab order of controls determines which control receives focus when a user presses the Tab key. Each time the user presses Tab, the focus moves to the next control in the tab order.

By default, the tab order of controls is the order in which you added the controls to the form. You can change the tab order so that it reflects the order in which you expect your users to use the controls.

To change the tab order of controls:

- 1 Expand **Record Types** or **Record Types – Stateless** in the Workspace until you see the desired form.
- 2 Double-click the form to display it.
- 3 Select the dialog tab containing the controls whose tab order you want to set.
- 4 Select **Form Layout > Set Tab Order**.

ClearQuest changes to tab-order mode. In this mode, each control displays a number indicating its position in the tab order.

- 5 Click the control you want to be first in the tab order.
- 6 Click each of the remaining controls in the order you want them to receive the focus.

As you click each control, its displayed number changes to match the new tab order.

After you click each of the controls once, ClearQuest exits tab-order mode. You can also exit tab-order mode by clicking in an empty portion of the dialog tab.

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; in ClearQuest Web you can either click a link or scroll to display 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 219 for information on using hooks on the Web client.

For more information, see “Administering ClearQuest Web” on page 217.

Creating forms to view and modify imported data

If you are updating a schema to accommodate imported records, you can modify existing forms by adding or removing appropriate fields. If you are creating a new schema, you can simply create a new form that displays the fields from the imported records.

When creating or modifying forms to view imported records, you might want to add a field to display the Old ID of the imported record. The Old ID field is not required for new records but is useful for identifying imported records using information from the old database.

Reusing forms

If you have a form that is common to multiple schemas, you can save time by creating the form once, then exporting the form to your local system, and importing it into another schema. Once you import the form, you can use it as is or modify it.

Note: You must import the form into a record type with a name identical to the record type that exported the form.

Exporting a form

Exporting a form saves all tabs, all controls on all tabs, and all form, tab, and font properties. But, if you want to use just a single tab in another form, you can. See “Copying tabs” on page 95.

Before exporting a form, you must close it. Closing the form saves all changes you have made to it.

To export a form:

- 1 Expand **Record Types or Record Types - Stateless** in the Workspace, until you see the form you want to export.
- 2 If the form is open, save the form by clicking its close box.
- 3 Right-click the form name in the Workspace and select **Export Form** from the shortcut menu.
- 4 In the Export Form dialog box, enter a file name for the exported form, select a location on your local system, and click **Save**.

The form file is saved with an FRM file name extension.

Importing a form

You must import the form into a record type with the same name as the record type from which you exported the 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 verify that the fields have matching types. If a field in the form does not correspond to a field in the selected record type, you must either reassign the corresponding control to a valid field or delete the control from the form.

To import a form:

- 1 Check out the schema that will receive the import data.
- 2 In the Workspace, expand **Record Types** or **Record Types - Stateless**.
- 3 Right-click the **Forms** folder of that record type and select **Import Form** from the shortcut menu.
- 4 In the Import Form dialog box, select the file containing the form and click **Save**.
- 5 After importing the form, reassign fields to controls if necessary.

Form controls reference

ActiveX control

The ClearQuest ActiveX control provides a way to incorporate any registered ActiveX control into a form. For example, you might use an ActiveX control to access and update an external database. You write the initialization record script that fires when the form containing the control is opened and the action record script that fires when a user initiates an action (such as Submit), or the action is completed or reverted.

Before using this control, you should be familiar with ActiveX functionality and how to register your controls.

The property sheet for an ActiveX control has two tabs: the General tab and the ActiveX Control tab.

General tab

- Entity Name
- Field Name
- Label
- X
- Y
- Width
- Height

ActiveX Control tab

- Registered Name
- Initialization Record
- Action Record Script

Attachment control

An attachment control displays a list of attached files and includes a set of controls that allow users to add, remove or view attached files. You associate an attachment control with an attachment field, that is, a field whose type is ATTACHMENT_LIST.

General tab

- X
- Y
- Width
- Height
- Label
- Field Name

Checkbox control

A check box control is a two-value control that you can use for Boolean values or any field that has only 2 values. A check box control can be either checked or unchecked. When you create the control, you specify the values to associate with the corresponding field when the control is checked or unchecked. You can assign any 2 strings for the checked and unchecked values of the control.

You should attach a default-value hook to the field belonging to this control. If you do not set the initial value of the control, ClearQuest sets its value to checked.

The property sheet for a checkbox control has two tabs: the General tab and the Extended tab.

General tab

- X
- Y
- Width
- Height
- Label
- Field Name

Extended tab

- Check Value
- Uncheck Value

Combo box control

A combo box control combines an editable text field with a list box. When displayed, the combo box displays the list of choices associated with the particular field. The user may enter additional choices using the editable text field of the combo box. If the number of entries in the list exceeds the size of the list, ClearQuest automatically adds a vertical scroll bar along the right edge of the control.

You can assign most field types to a combo box control. However, any field you assign to the combo box should have a choice list associated with it or it should support the addition of a choice list using the combo box. For most field types, only one item from the choice list may be selected in the combo box at any given time. However, ClearQuest allows multiple selections for fields that can support them.

The property sheet for a combo box control has three tabs: the General tab, the Extended tab, and the Context Menu Hooks tab.

General tab

- X
- Y
- Width
- Height
- Label
- Field Name

Extended tab

Contains a single check box for the Auto Sort property. If this check box is enabled, entries in the list portion of the drop-down combo box are sorted alphabetically. If this check box is disabled, entries are listed in the order in which they were added.

Context Menu Hooks tab

Displays the list of available record hooks and the ones that are currently associated with this control. To associate a hook with the control, select the hook in the Available column and click the Add button. To disassociate a hook, select the hook in the Selected column and click the Remove button.

Hooks associated with this control are displayed in the control's shortcut menu in the ClearQuest client application. To execute one of these hooks, the user must right-click the control and select the name of the hook.

Web Dependent Fields

In order to have a dependent field work in the ClearQuest web client, you use the Web Dependent Fields tab to specify the field on which the dependency is based. This is done when you add the field to a form.

To indicate the field on which the dependency is based, select a field from the **Available** box and click **Add**. To remove a field, select it from the **Selected** box and click **Remove**.

Drop-down list box control

A drop-down list box control displays a list of possible values for a particular field. Clicking on the control displays the list of values associated with the field and allows the user to choose one of the values. When the field's behavior is set to read-only, the currently selected value is displayed; you cannot edit this value. The entries in the drop-down list box are not editable by the user; however, you may update the list programmatically using a choice-list hook.

You can assign most field types to a drop-down list box control. However, any field you assign to the control should have a choice list associated with it. Only one item at a time can be selected from the list with this control.

The property sheet for a drop-down list box control has three tabs: the General tab, the Extended tab, and the Context Menu Hooks tab.

General tab

Use the General tab to define the position, size, field used, if applicable, and label of the control.

- X
- Y
- Width
- Height
- Label
- Field Name

Extended tab

Contains a single check box for the Auto Sort property. If this check box is enabled, entries in the list portion of the drop-down combo box are sorted alphabetically. If this check box is disabled, entries are listed in the order in which they were added.

Context Menu Hooks

Displays the list of available record hooks and the ones that are currently associated with this control. To associate a hook with the control, select the hook in the Available column and click the Add button. To disassociate a hook, select the hook in the Selected column and click the Remove button.

Hooks associated with this control are displayed in the control's shortcut menu in the ClearQuest client application. To execute one of these hooks, the user must right-click the control and select the name of the hook.

Web Dependent Fields

In order to have a dependent field work in the ClearQuest web client, you use the Web Dependent Fields tab to specify the field on which the dependency is based. This is done when you add the field to a form.

To indicate the field on which the dependency is based, select a field from the **Available** box and click **Add**. To remove a field, select it from the **Selected** box and click **Remove**.

Drop-down combo box control

A drop-down combo box control combines an editable text field with a drop-down list box. When displayed, the combo box displays the currently selected choice for the particular field. The user may select a new choice from the drop-down list or type in a new choice.

You can assign most field types to a drop-down combo box control. However, any field you assign to the control should have a choice list associated with it or should support the addition of a choice list using the combo box. Only one item at a time can be selected from the list with this control.

The property sheet for a drop-down combo box control has three tabs: the General tab, the Extended tab, and the Context Menu Hooks tab.

General tab

Use the General tab to define the position, size, field used, if applicable, and label of the control.

- X
- Y
- Width
- Height
- Label
- Field Name

Extended tab

Contains a single check box for the Auto Sort property. If this check box is enabled, entries in the list portion of the drop-down combo box are sorted alphabetically. If this check box is disabled, entries are listed in the order in which they were added.

Context Menu Hooks tab

Displays the list of available record hooks and the ones that are currently associated with this control. To associate a hook with the control, select the hook in the Available column and click the Add button. To disassociate a hook, select the hook in the Selected column and click the Remove button. Hooks associated with this control are displayed in the control's shortcut menu in the ClearQuest client application. To execute one of these hooks, the user must right-click the control and select the name of the hook.

Web Dependent Fields

In order to have a dependent field work in the ClearQuest web client, you use the Web Dependent Fields tab to specify the field on which the dependency is based. This is done when you add the field to a form.

To indicate the field on which the dependency is based, select a field from the **Available** box and click **Add**. To remove a field, select it from the **Selected** box and click **Remove**.

Duplicate base control

This control displays the ID of the record of which this record is a duplicate. This control is a special type of text box control because you cannot associate a field with it. Instead, ClearQuest updates the contents of this control when it determines that this record is a duplicate of another record.

Note: When creating this control, you only need to make the control large enough to hold a single record ID string. A record can be a duplicate of only one other record.

General tab

Use the General tab to define the position, size, field used, if applicable, and label of the control.

- X
- Y
- Width
- Height
- Label
- Field Name

Duplicate dependents control

This control displays the IDs of any records that are duplicates of this record. This control is a special type of list box control because you cannot associate a field with it. Instead, ClearQuest updates the contents of this control when it determines that one or more records are duplicates of this record.

Note: Because a record may have more than one duplicate, you should adjust the size of this control appropriately for your form. If there are more duplicates than can be displayed at once, ClearQuest displays a scroll bar so that you can scroll through the list.

General tab

Use the General tab to define the position, size, field used, if applicable, and label of the control.

- X
- Y
- Width
- Height
- Label
- Field Name

Group box control

A group box control is a decorative control used to visually group one or more other controls. You do not associate a field with a group box control, but you can assign a label to the control. Typically, the label of a group box control identifies the purpose of the enclosed controls.

General tab

Use the General tab to define the position, size, field used, if applicable, and label of the control.

- X
- Y
- Width
- Height
- Label
- Field Name

History control

A history control displays the actions that have been applied to a record. The history control itself is a special type of list view, each item of which contains information about a single action taken on the record. ClearQuest maintains the record's action history internally and displays it in your form's history control. You cannot associate a field with the history control.

General tab

Use the General tab to define the position, size, field used, if applicable, and label of the control.

- X
- Y
- Width
- Height
- Label
- Field Name

List box control

A list box control displays a list of possible values for a particular field. The number of entries in the list is fixed unless you change them programmatically using a choice-list hook. If the number of entries in the list exceeds the size of the list, ClearQuest automatically adds a vertical scroll bar along the right edge of the control.

You can assign most field types to a list box control. However, any field you assign to the control should have a choice list associated with it. For most field types, only one item from the list may be selected in the control at any given time. However, ClearQuest allows multiple selections for fields that can support them.

The property sheet for a list box control has three tabs: the General tab, the Extended tab, and the Context Menu Hooks tab.

General tab

Use the General tab to define the position, size, field used, if applicable, and label of the control.

- X
- Y
- Width
- Height
- Label
- Field Name

Extended tab

Contains a single check box for the Auto Sort property. If this check box is selected, entries in the list portion of the list box are sorted alphabetically. If this check box is cleared, entries are listed in the order in which they were added.

Context Menu Hooks tab

Displays the list of available record hooks and the ones that are currently associated with this control. To associate a hook with the control, select the hook in the Available column and click the Add button. To disassociate a hook, select the hook in the Selected column and click the Remove button.

Hooks associated with this control are displayed in the control's shortcut menu in the ClearQuest client application. To execute one of these hooks, the user must right-click the control and select the name of the hook.

List view control

A list view control allows you to display the records associated with a field whose type is REFERENCE LIST. One of the main uses of the list view control is to display the parent of a parent/child relationship. If you want to use a list view to link related records, use the parent/child control instead.

- To determine which fields of a record are displayed in a list view control, you need to add columns to the list view. To do this, right-click in the header bar and choose **Add Column** from the shortcut menu.
- Each column refers to a particular field of the referenced records. Right-click a column and choose **Properties** from the shortcut menu to edit the column name and the field that it references.

Note: You should use a list view control to display a back reference field in your form. The back reference field is read-only REFERENCE LIST field and is used to store the parent record information in a parent/child relationship.

The property sheet for a list view control has three tabs: the General tab, the Extended tab, and the Context Menu Hooks tab.

General tab

Use the General tab to define the position, size, field used, if applicable, and label of the control.

- X
- Y
- Width
- Height
- Label
- Field Name

Extended tab

Contains a single text box for entering the ID of the list view. This ID must be unique among any other list view controls in that tab. You will need this ID for the push buttons you associate with the list view.

Context Menu Hooks tab

Displays the list of available record hooks and the ones that are currently associated with this control. To associate a hook with the control, select the hook in the Available column and click the Add button. To disassociate a hook, select the hook in the Selected column and click the Remove button. Hooks associated with this control are displayed in the control's shortcut menu in the ClearQuest client. To execute one of these hooks, the user must right-click the control and select the name of the hook.

Option button control

You use option button controls in groups to represent a set of mutually exclusive choices. You must create each option button in the group separately and assign it the same group name. ClearQuest creates only one label for each group of option buttons you create with the same group name.

You can associate each option button with a field and provide an appropriate value to put in that field when the button is selected. You may assign a different field to each option button in the group if you wish.

The property sheet for an option button control has two tabs: the General tab and the Extended tab.

General tab

Use the General tab to define the position, size, field used, if applicable, and label of the control.

- X
- Y
- Width
- Height
- Label
- Field Name

Extended tab

The Extended tab contains the following properties:

Property	Description
Group Name	The name of the group this option button is associated with. Only one option button in a group may be selected at any given time. Selecting a different button in the group deselects the currently selected button.
Group Label	The label displayed for a group of option buttons. There is only one label for the group of buttons with the given Group Name.
Value	The value returned when this option button is selected.

Parent/Child control

A parent/child control is used when you want to set up a form to allow users to link associated records. The parent/child control is used with REFERENCE_LIST field type.

It consists of both a list view control and three push buttons. The list view control and push buttons are automatically associated using a unique list view ID. If you change the ID of the list view, you must also update the push buttons.

To determine which fields of a referenced record are displayed in a parent/child control, you need to add columns to the list view. To do this,

- Right-click in the header bar and choose **Add Column** from the shortcut menu.
- Right-click a column and choose **Properties** from the shortcut menu to edit the column name and the field that it references. Each column refers to a particular field of the referenced records.

The property sheet for a list view control has three tabs: the General tab, the Extended tab, and the Context Menu Hooks tab.

General tab

Use the General tab to define the position, size, field used, if applicable, and label of the control.

- X
- Y
- Width
- Height
- Label
- Field Name

Extended tab

Contains a single text box for entering the ID of the parent/child control. This ID must be unique among any other list view or parent/child controls in that tab, and must be the same as the as the list view ID for the associated buttons. If you change the ID of the list view, you must also update the push buttons.

Context Menu Hooks tab

Displays the list of available record hooks and the ones that are currently associated with this control. To associate a hook with the control, select the hook in the Available column and click the Add button. To disassociate a hook, select the hook in the Selected column and click the Remove button. Hooks associated with this control are displayed in the control's shortcut menu in the ClearQuest client. To execute one of these hooks, the user must right-click the control and select the name of the hook.

Picture control

A picture control lets you display a static image on your form. You can use picture controls to display any .BMP image file. The bitmap image can be obtained either from a file or from the clipboard. ClearQuest scales the bitmap proportionally to fit the width and height of the picture control.

The property sheet for a picture control contains the following properties:

Property	Description
X	The horizontal position of the upper-left corner of the picture.
Y	The vertical position of the upper-left corner of the picture.
Width	The width of the picture.
Height	The height of the picture.
File	Select this option to read the picture from a file. With this option selected, the Picture File Name text box must contain the name of the file.
Clipboard	Select this option to load the picture from the clipboard.
Picture File Name	A text box containing the path name of the file.

Push button control

You can use push button controls to initiate specific tasks related to the record. You can associate push buttons with record hooks or with list views.

- When associated with a record hook, clicking the button executes the specified record hook. You are responsible for defining and debugging the record hooks associated with your schema.
- When associated with a list view, you must assign the button to one of three predefined tasks: adding a record, removing a record, or creating a new record. ClearQuest provides the code to perform these tasks. You must also associate the push button with the corresponding list view ID.

Note: Scripted buttons should not be enabled in the ClearQuest Web client if they call dialog boxes or other Windows applications.

General tab

Use the General tab to define the position, size, field used, if applicable, and label of the control.

- X
- Y
- Width
- Height
- Label
- Field Name

Extended tab

Use the Extended tab to associate a push button with a record hook or with a List View control.

- Scripted
- Enabled for web
- Record Hook
- Add
- Remove
- New
- List View ID

Static text box control

A static text box displays an uneditable text string. Although you can assign a field to a static text box control, it is generally better to use a text box control with fields so that the user can modify the contents of the field. A static text box prevents the user from ever editing the contents of the field, which may not be what you want. If you want to prevent the user from editing the field only during specific actions, you should use a text box control and adjust the field's behavior settings appropriately.

General tab

Use the General tab to define the position, size, field used, if applicable, and label of the control.

- X
- Y
- Width
- Height
- Label
- Field Name

Text box control

A text box control displays a field's value as an editable text string. When the field's behavior is optional or mandatory, the user can modify the field by editing the contents of a text box control. If you want to display a simple text string that never changes, use a static text box control.

Text box controls give you several options related to the appearance of the typed text. You can prevent the echoing of typed characters, which is useful for entering passwords or other secure information. You can control the formatting of date and time information. You can allow the user to format the text in the control using multiple lines. You can also add scroll bars to enable automatic or manual scrolling of the text box contents.

The property sheet for a text box control has four tabs: the **General** tab, the **Extended** tab, the **Date/Time** tab, and the **Context Menu Hooks** tab.

General tab

Use the General tab to define the position, size, field used, if applicable, and label of the control.

- X
- Y
- Width
- Height
- Label
- Field Name

Extended tab

The Extended tab contains the following properties:

Property	Description
Password/no echo style	Characters in the text box are replaced with asterisks for security purposes
Multi Line	Text in the text box can span multiple lines. If this option is disabled, all text is placed on the same line
Auto Vert Scroll	Enables automatic vertical scrolling when the user types beyond the bottom of the text box or selects text beyond the top or bottom edge of the text box.

Property	Description
Auto Horz Scroll	Enables automatic horizontal scrolling when the user types beyond the end of the line or selects text beyond the left or right edge of the text box.
Vert Scroll Bar	Displays a vertical scroll bar along the right side of the text box.
Horz Scroll Bar	Displays a horizontal scroll bar along the bottom of the text box.

Date/Time tab

The Date/Time tab contains the following properties:

Property	Description
Date	If checked, this control enables the Date Formats list, allowing you to choose an appropriate date format.
Time	If checked, this control enables the Time Formats list, allowing you to choose an appropriate time format.
Date Formats	Contains the list of formats you can use to display date information. Select the desired format from the list.
Time Formats	Contains the list of formats you can use to display time information. Select the desired format from the list.

Note: If you enable the Date checkbox, ClearQuest automatically adds an additional button to the right of the text box control. When clicked, this button displays a calendar with which the user can choose a specific date to enter into the field.

Context Menu Hooks tab

The Context Menu Hooks tab displays the list of available record hooks and the ones that are currently associated with this control. To associate a hook with the control, select the hook in the Available column and click the Add button. To disassociate a hook, select the hook in the Selected column and click the Remove button. Hooks associated with this control are displayed in the control's shortcut menu. To execute one of these hooks, the user must right-click the control and select the name of the hook.

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.

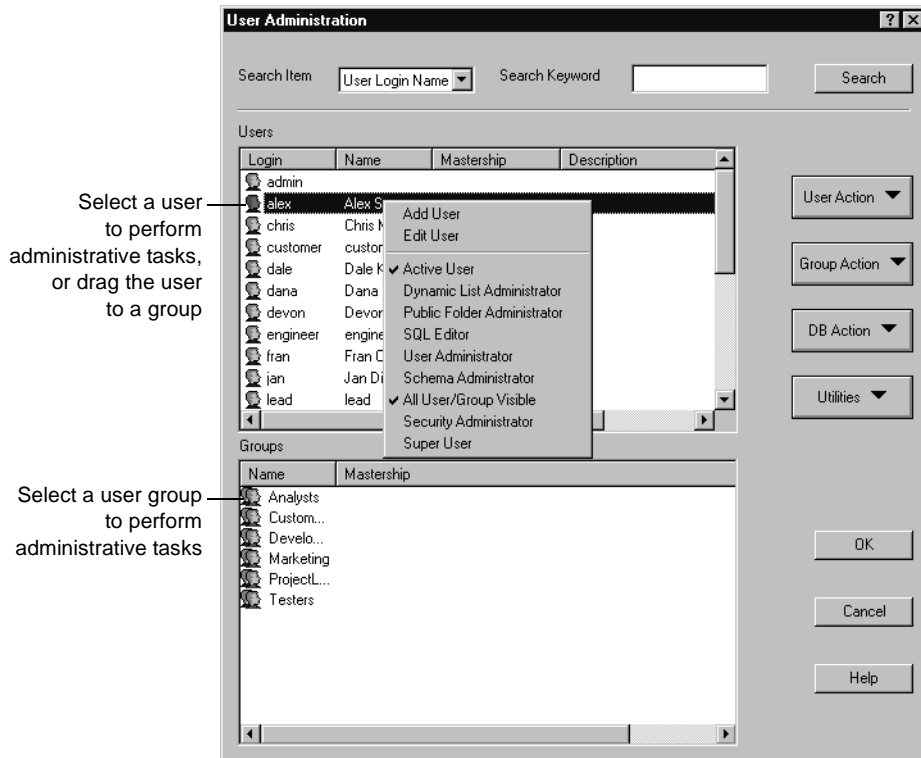
This chapter describes how to set up and administer ClearQuest users and user groups. The topics covered include:

- Overview of user administration
- ClearQuest user privileges
- Working with users
- Working with user groups
- Restricting user access to actions
- Exporting and importing users and user groups
- Administering users in a MultiSite environment

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 138.

Overview of user administration

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



Use the User Administration dialog box 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 138. You can also restrict user and group access to specific ClearQuest actions by adding an access-control hook to the action. See “Restricting user access to actions” on page 151.
- Control the data that users and groups can access by giving them access to specific databases. See “Subscribing users and groups to databases” on page 142.

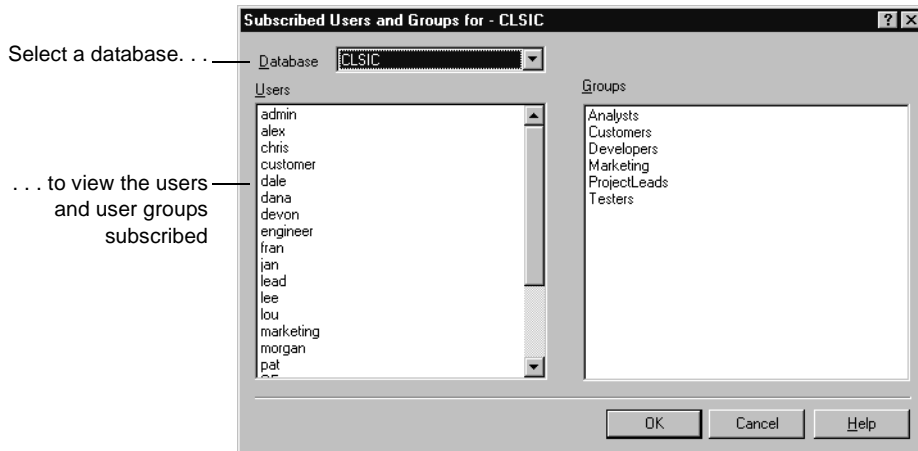
- Export user data and import it into another schema repository. See “Exporting and importing users and user groups” on page 152.

Note: Whenever you add or modify a user or user group, you must upgrade your user database(s) with the new information. See “Subscribing users and groups to databases” on page 142.

Viewing database subscriptions

To view the users or user groups that are subscribed to each database:

- 1 In ClearQuest Designer, select **Tools > User Administration**.
- 2 In the User Administration dialog box, select **DB Action > View Subscription**.
- 3 In the Subscribed Users and Groups dialog box, select a database to view the users and user groups subscribed.



- 4 Click **OK** to close the Subscribed Users and Groups dialog box.

ClearQuest user privileges

ClearQuest supports the following user and user group privileges:

Privilege	Allows user or group to
Active User	<p>This is the default privilege for all new ClearQuest users.</p> <p>Access the basic features of ClearQuest: log into ClearQuest user database and submit new records; modify existing records; and create, modify, and save personal queries, charts, and reports.</p> <p>Log into ClearQuest Designer to view schemas and to view user administration information. Cannot edit schemas or change user information.</p>
All Users/Groups Visible	<p>This is a default privilege for all new ClearQuest users.</p> <p>Users can see information about all other users and groups. See “Administering users in a MultiSite environment” on page 153.</p>
User Administrator	<p>Perform all Active User tasks and:</p> <ul style="list-style-type: none">• Use ClearQuest Designer to create users and user groups and assign and modify their user-access privileges.• Cannot perform any other tasks in ClearQuest Designer.
Dynamic List Administrator	<p>Perform all Active User tasks and:</p> <ul style="list-style-type: none">• Edit dynamic lists in ClearQuest client.
Public Folder Administrator	<p>Perform all Active User tasks and:</p> <ul style="list-style-type: none">• Create, modify, save, and delete public queries, charts, and reports in ClearQuest client.
SQL Editor	<p>Perform all Active User tasks and:</p> <ul style="list-style-type: none">• Edit SQL for queries in ClearQuest client.
Security Administrator	<p>Perform all Active User tasks and:</p> <ul style="list-style-type: none">• Edit SQL for queries in ClearQuest client.• Edit the context group list field for a security context record.• View all records.
Schema Designer	<p>Perform all Active User tasks and:</p> <ul style="list-style-type: none">• Use ClearQuest Designer to create and modify schemas. Add record types, define and modify fields, create and modify states and actions, add hooks to the schema, and update existing databases.• Create, modify, and save public queries, charts, and reports in ClearQuest client.• Cannot perform User Administrator tasks.
Super User	<p>Perform all Active User, Schema Designer, User Administrator, Security Administrator, Public Folder Administrator, Dynamic List Administrator, and SQL Editor tasks and:</p> <ul style="list-style-type: none">• Use ClearQuest Designer to create and delete databases and schemas.• Edit ClearQuest Web settings. <p>The <code>admin</code> user account that comes with ClearQuest Designer has Super User privilege.</p>

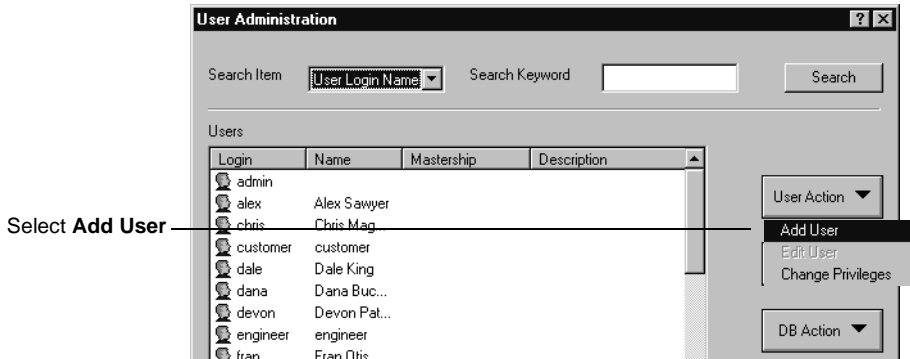
Working with users

To create and modify users or user groups, you must have User Administrator or Super User privileges.

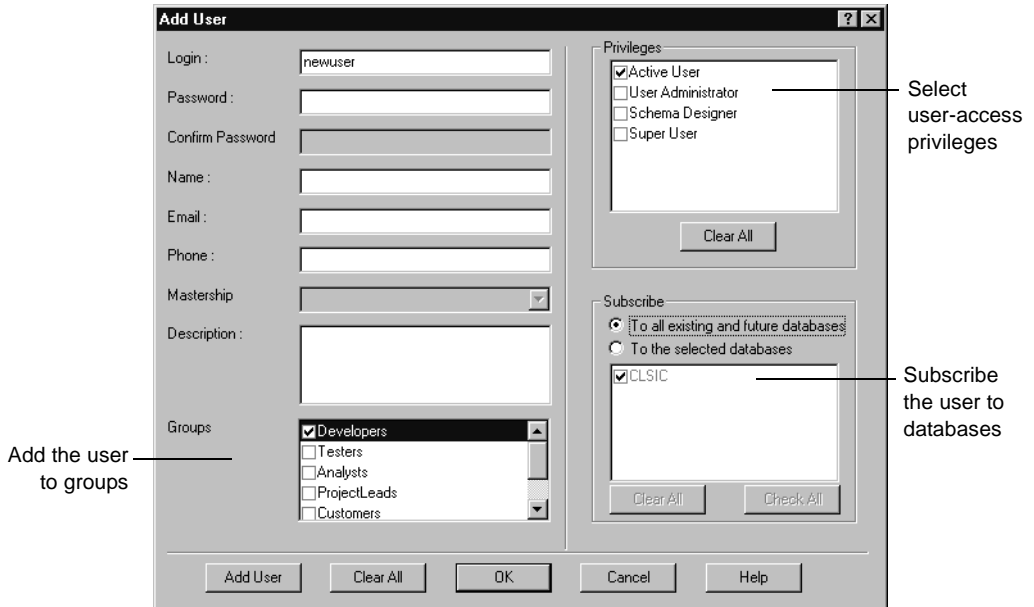
Adding a new user

To add a new user:

- 1 In ClearQuest Designer, select **Tools > User Administration**.
- 2 In the User Administration dialog box, select **User Action > Add User**.



- 3 In the Add User dialog box, fill in the user information.



- 4 Type the user's **Login**, **Password**, **Name**, **Email** address, **Phone**, and if desired, a **Description**.

Warning: Do not use the following characters for a user **Login**. Using these characters can cause problems when logging into ClearQuest client and ClearQuest Web: ! " # \$ % & ' () * + , . / : ; < = > ? @ [\] ^ ` { | } ~

- 5 If you are using ClearQuest MultiSite, select the **Mastership**. For more information, see “Administering users in a MultiSite environment” on page 153.
- 6 Select the **Groups** to add the new user to. For more information, see “Working with user groups” on page 148.
- 7 Select the Privileges for the new user. For more information, see “Assigning user access privileges” on page 141.
- 8 Click **Add User**, then click **OK**.
- 9 In the User Administration dialog box, select **DB Action > Upgrade** to add this information to the user database. For more information, see “Applying schema changes to the user database” on page 144.

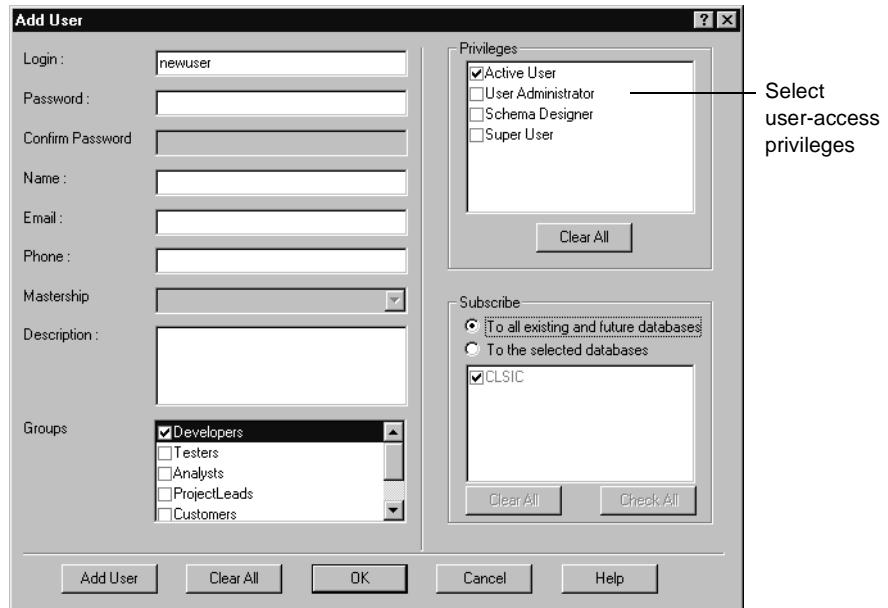
Assigning user access privileges

Note: To grant a privilege to a user, you must have the privilege that you are granting.

By default, all users have the Active User privilege. Check any additional privileges for the user. For a description of access privileges, see “ClearQuest user privileges” on page 138.

To assign user access privileges:

- 1 In the New User dialog box, select the **Privileges** for the user.



- 2 Click **OK**.

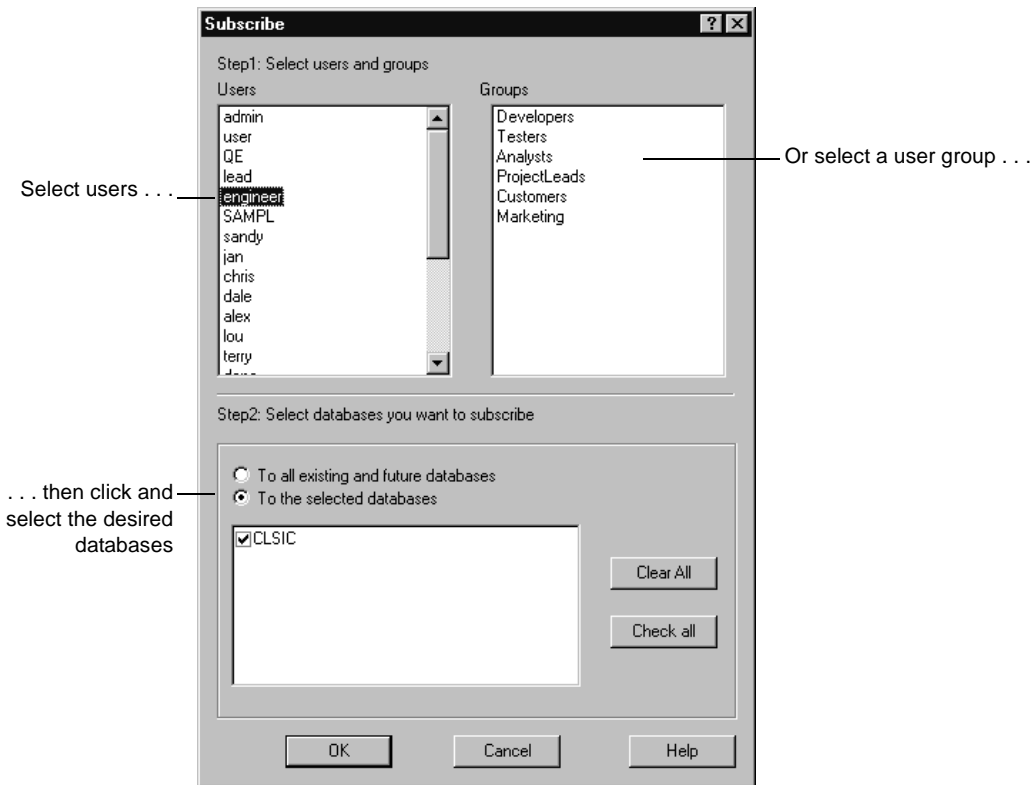
- 3 In the User Administration dialog box, select **DB Action > Upgrade** to add this information to the user database. For more information, see “Applying schema changes to the user database” on page 144.

Subscribing users and groups to databases

Subscribing a user or a user group to a database allows that user or group to access the database. By default, ClearQuest subscribes users and groups to all existing databases and to all databases created in the future. You can restrict user access to the database(s) you specify.

To subscribe users and user groups to selected database(s):

- 1 In ClearQuest Designer, select **Tools > User Administration**.
- 2 In the User Administration dialog box, select **DB Action > Subscribe**.
- 3 In the Subscribe dialog box, select users and groups, then click **To the selected databases** and select the desired databases.



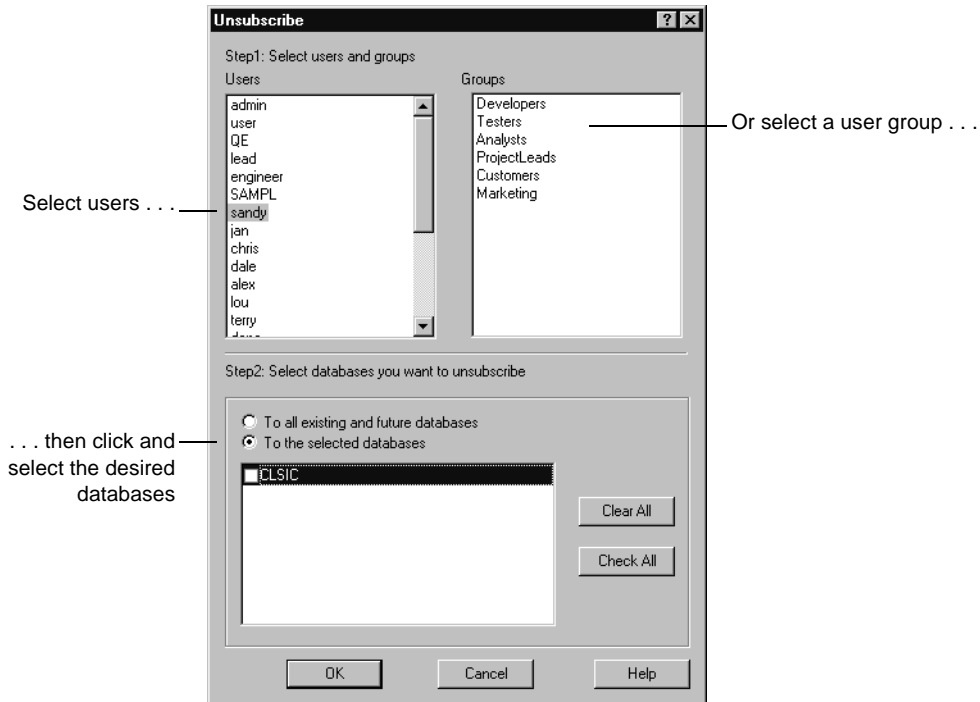
- 4 Click **OK**.
- 5 In the User Administration dialog box, select **DB Action > Upgrade** to add this information to the user database. For more information, see “Applying schema changes to the user database” on page 144.

Unsubscribing users and groups from databases

Unsubscribing a user or a user group from a database removes that user or group access to the database.

To unsubscribe users or user groups to selected database(s):

- 1 In ClearQuest Designer, select **Tools > User Administration**.
- 2 In the User Administration dialog box, select **DB Action > Unsubscribe**.
- 3 In the Unsubscribe dialog box, select users and/or user groups, then click **To the selected databases** and select the databases to remove from access.



- 4 Click **OK**.
- 5 In the User Administration dialog box, select **DB Action > Upgrade** to add this information to the user database. For more information, see “Applying schema changes to the user database” on page 144.

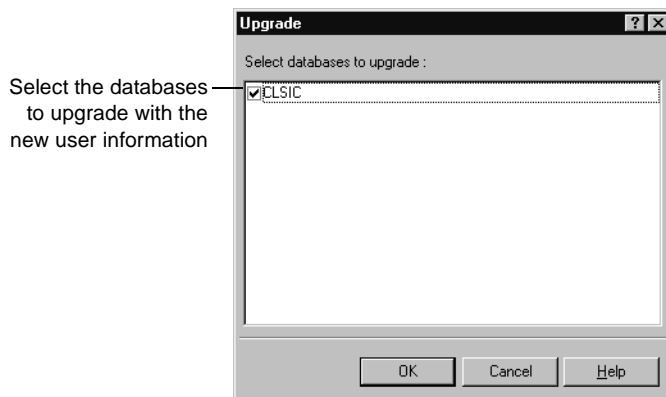
Applying schema changes to the user database

Whenever you add or modify users or user groups, you must apply the schema changes to your user database(s).

Warning: Before applying changes to a user database, make sure that there are no users logged in to the database. If users are connected to the user database, the upgrade will fail. ClearQuest prevents new users from logging in to a database while you are updating it, but it does not disconnect users who are currently logged in.

To apply the schema changes to the user database:

- 1 In ClearQuest Designer, select **Tools > User Administration**.
- 2 In the User Administration dialog box, select **DB Action > Upgrade**.
- 3 In the **Upgrade** dialog box, select the databases that you want the changes applied to and click **OK**.



- 4 Click **OK** to close the Upgrade dialog box, then click **OK** again to close the User Administration dialog box.

Editing users

As an administrator with User Administrator or Super User privileges, you can edit a user profile.

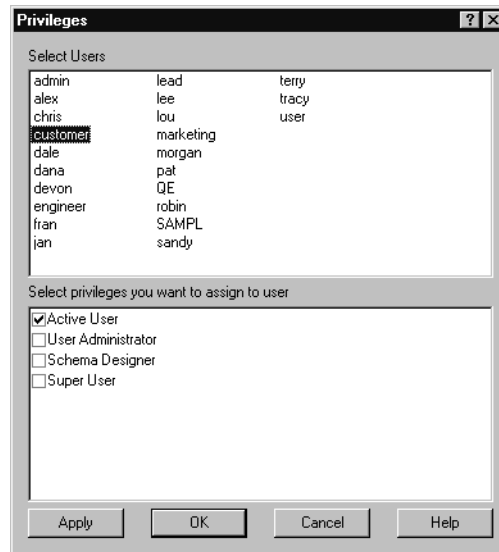
To edit a user profile:

- 1 In ClearQuest Designer, select **Tools > User Administration**.
- 2 In the User Administration dialog box, select **User Action > Edit**.
- 3 Edit the user profile as desired. For more information, see “Adding a new user” on page 139.

Changing user privileges

To change user privileges:

- 1 In the User Administration dialog box, select the user, then select **User Action > Change Privileges**.
- 2 In the Privileges dialog box, select a user and then select or unselect the desired privilege(s).



- 3 When you're done, click **Apply**, then **OK**.
- 4 Upgrade the user database with the new user information. See “Applying schema changes to the user database” on page 144.

Editing a user profile from the ClearQuest client

Note: Any ClearQuest user can edit some of their own user information from the ClearQuest client. For more information, see “Editing a user profile from the ClearQuest client.”

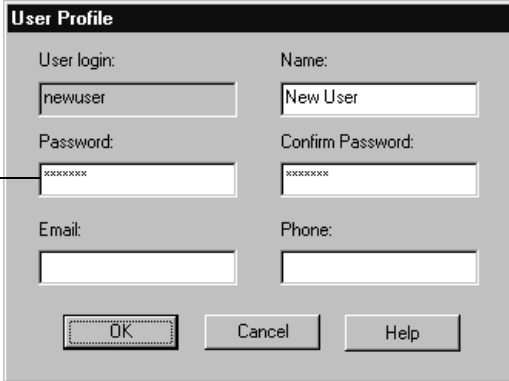
Active users can also change some of their own user information, including their name, e-mail address, password, and telephone number, from the ClearQuest client. However, active users cannot change their own user login or privileges.

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

To modify a user profile from the ClearQuest client:

- 1 In the ClearQuest client, select **View > Change user profile**.

Change the user information



User login:	Name:	
<input type="text" value="newuser"/>	<input type="text" value="New User"/>	
Password:	Confirm Password:	
<input type="password" value="*****"/>	<input type="password" value="*****"/>	
Email:	Phone:	
<input type="text"/>	<input type="text"/>	
<input type="button" value="OK"/>	<input type="button" value="Cancel"/>	<input type="button" value="Help"/>

- 2 Click **OK**.

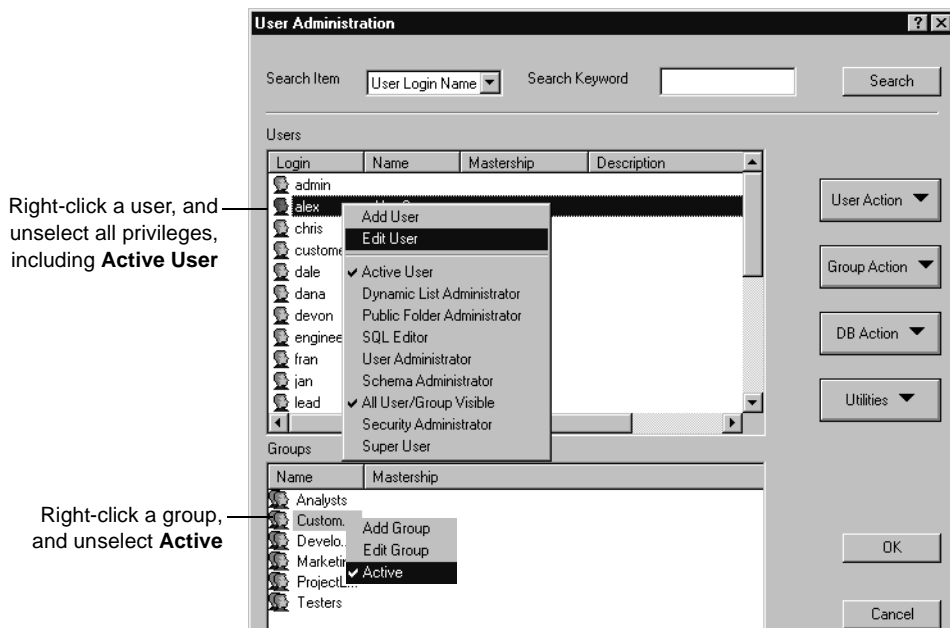
Making users and groups inactive

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

Note: For historical information and data integrity, ClearQuest Designer retains the names of inactive users in the database that the users were subscribed to.

To make a user or user group inactive:

- 1 In ClearQuest Designer, select **Tools > User Administration**.
- 2 In the User Administration dialog box:
 - To make a user inactive, right-click the user and use the shortcut menu to unselect all user privileges, including **Active User**.
 - To make a group inactive, right-click the group and unselect **Active** in the shortcut menu.



- 3 Select **DB Action > Upgrade** to add this information to the user database. For more information, see “Applying schema changes to the user database” on page 144.

To make a user or user group active again, reselect **Active User** for the user or **Active** for the group.

Working with user groups

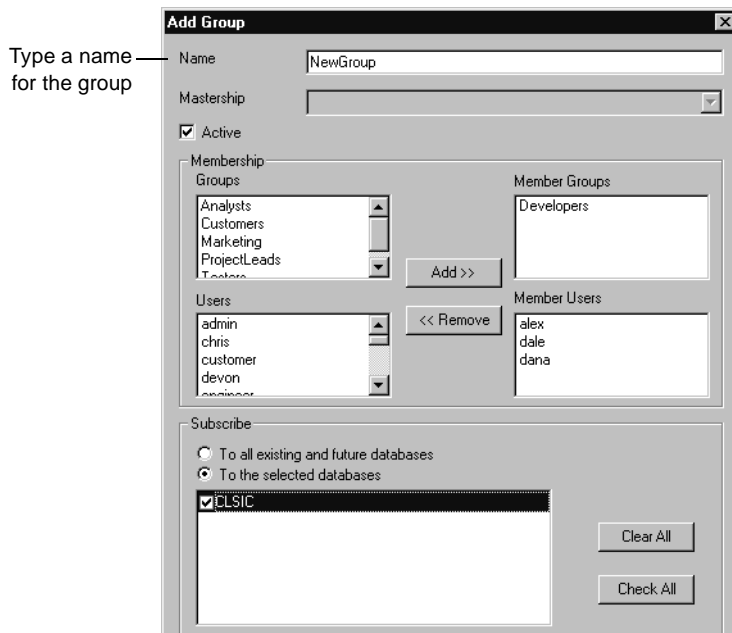
Combining individual users into a group can make your administration tasks easier. For example, you can:

- Create groups such as “Software Engineers” or “Managers,” 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 a new user group

To create a user group, you must have User Administrator or Super User privileges. To create a user group:

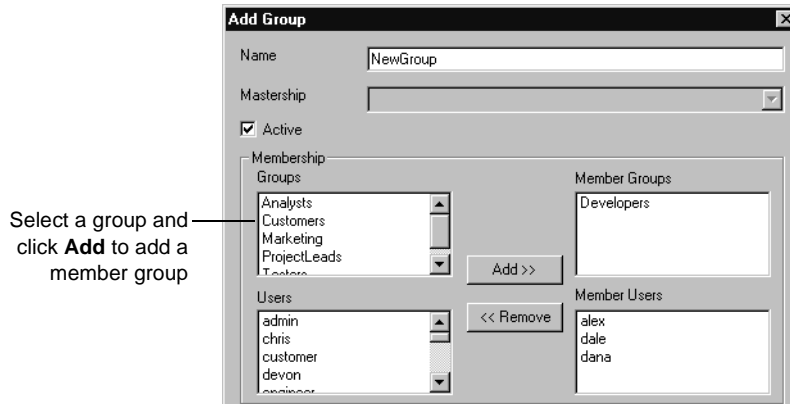
- 1 In ClearQuest Designer, select **Tools > User Administration**.
- 2 In the User Administration dialog box, select **Group Action > Add Group**.
- 3 In the Group dialog box, type a **Name** for the new group.
- 4 If you are using ClearQuest MultiSite, select the **Mastership**.



Creating subgroups

To add subgroups to a group:

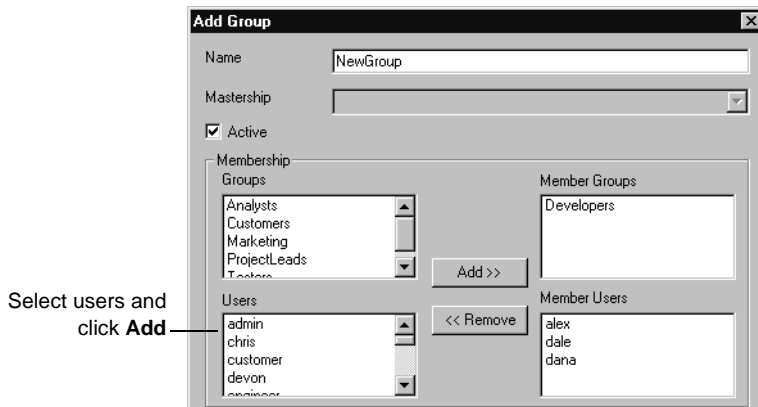
- In the Group dialog box, select an existing group(s) from the **Groups** list and click **Add**.



Adding users to a group

To add users to a group:

- In the Group dialog box, select the user(s) you want to add to the group, and click **Add**.

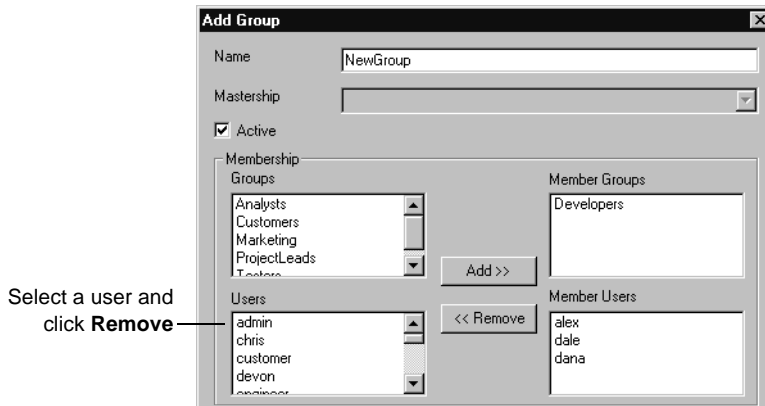


Note: Alternatively, in the User Administration dialog box, you can select users in the **Users** list and drag them to a group in the **Groups** list.

Removing users from a group

To remove users from a group:

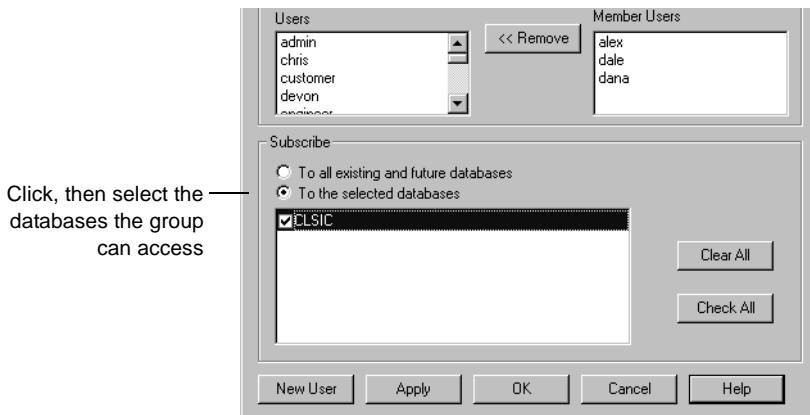
- In the Group dialog box, select a user and click **Remove**.



Subscribing user groups to databases

To subscribe a group only to specific databases:

- 1 In the Group dialog box, click **To the selected databases** and select the desired databases.



- 2 When you're finished, click **Apply** to close the Group dialog box.
- 3 In the User Administration dialog box, select **DB Action > Upgrade** to add the changes to the user database. For more information, see "Applying schema changes to the user database" on page 144.

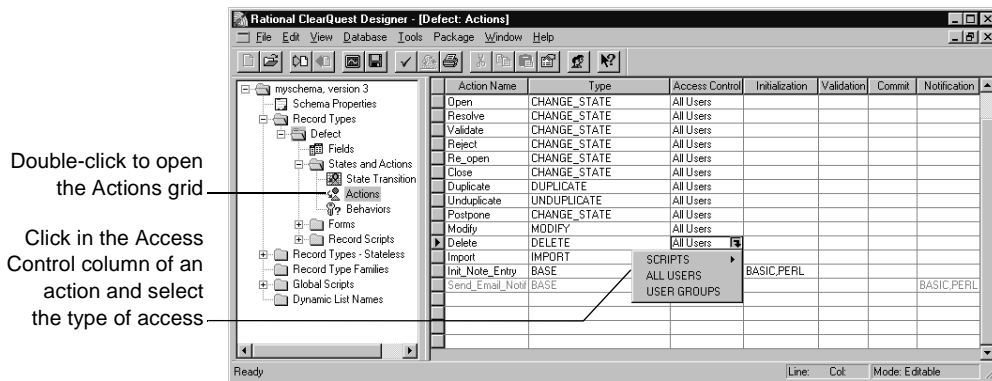
See also, "Subscribing users and groups to databases" on page 142.

Restricting user access to actions

By default, all ClearQuest client users can perform all actions on records. You can focus the responsibility of a user or user group by restricting the actions they can perform in the ClearQuest client. 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, you add an Access Control hook to the action.

To restrict access to an action:

- 1 In ClearQuest Designer, expand **Record Types** > <record type> > **States and Actions**, then double-click **Actions** to display the Actions grid.



- 2 In the Actions grid, click the Access Control cell of the action you want to restrict, then click the down-arrow and select the type of access control to be associated with the action:

- 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 8, “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.

Exporting and importing users and user groups

You can export the profiles of users and user groups from one schema repository and import them into another schema repository.

To export users and user groups:

- 1 In ClearQuest Designer, select **Tools > User Administration** to open the User Administration dialog box.
- 2 In the User Administration dialog box, click **Utilities** and select **Export**.

ClearQuest Designer creates a text file containing the profiles of each user and user group. You can import this text file into another schema repository.

To import users and user groups:

- 1 In ClearQuest Designer, select **Tools > User Administration** to open the User Administration dialog box.
- 2 In the User Administration dialog box, click **Utilities** and select **Import**.
- 3 Select the file containing the user and group information you want to import.

Administering users in a MultiSite environment

In a ClearQuest MultiSite environment, tracking changes and preventing data corruption is accomplished with an exclusive-right-to-modify called *mastership*. Mastership determines when a user of a database replica is allowed to modify data.

With mastership, users and groups (as well as other ClearQuest objects such as records and queries) are assigned a mastering replica. The initial mastering replica of a ClearQuest user or group is the site where the object is created.

This section describes:

- How mastership affects ClearQuest client users
- How mastership affects user administration
- Finding out where users and groups are currently mastered
- Changing the mastership of a user

How mastership affects ClearQuest client users

ClearQuest user access privileges are not affected by mastership; database privileges are propagated from replica to replica. Therefore, if a user has SQL Editor privileges on the local replica, that user has SQL Editor privileges on all replicas, if the replicas are synchronized.

Mastership affects ClearQuest client users in the following ways:

- A user can only modify records or objects (such as Public Queries) that are currently mastered at the replica they are logged into. The user cannot modify records or objects which are not currently mastered at their local replica.
- A user can perform the tasks listed below only at the replica where that user is currently mastered. For example, if a user is currently mastered at Replica A, that user can perform these tasks only when logged into Replica A.
 - Change their user profile
 - Save defaults while submitting queries
 - Assign startup queries
 - Add workspace items to the Query menu as favorites

How mastership affects user administration

Mastership affects user administration in the following ways:

- You must have Super User privilege to change the mastership of a user.
- You can only modify users or groups which are currently mastered at the replica where you are logged in. You cannot change the privileges of a user who is currently mastered at another site.
- You can change the mastership of a user.

It is important that you know where a user is currently mastered.

Finding out where users and groups are currently mastered

When a user is created at one site, that user is visible on all sites with that replica; however, that user can *only* be modified at the site that currently has mastership of the user.

To find out where a user or group is currently mastered, look at the **Mastership** column in the User Administration dialog box.

The screenshot shows the 'User Administration' dialog box. At the top, there is a search section with 'Search Item' set to 'User Login Name' and a 'Search Keyword' field. Below this are two tables. The first table, titled 'Users', has columns for 'Login', 'Name', 'Mastership', and 'Description'. It lists users such as 'admin', 'alex', 'chris', 'customer', 'dale', 'dana', 'devon', 'engineer', 'fran', 'jan', and 'lead'. The 'Mastership' column is highlighted with a red box and a label: 'The Mastership column shows where users are currently mastered'. The second table, titled 'Groups', has columns for 'Name' and 'Mastership'. It lists groups like 'Analysts', 'Custom...', 'Develo...', 'Marketing', 'ProjectL...', and 'Testers'. A red box highlights the 'Mastership' column with a label: 'And where groups are currently mastered'. To the right of the tables are several action buttons: 'User Action', 'Group Action', 'DB Action', 'Utilities', 'OK', and 'Cancel'.

Login	Name	Mastership	Description
admin			
alex	Alex Sawyer		
chris	Chris Mag...		
customer	customer		
dale	Dale King		
dana	Dana Buc...		
devon	Devon Pat...		
engineer	engineer		
fran	Fran Otis		
jan	Jan Dillard		
lead	lead		

Name	Mastership
Analysts	
Custom...	
Develo...	
Marketing	
ProjectL...	
Testers	

Changing the mastership of a user

Note: You must have Super User privilege to change the mastership of a user.

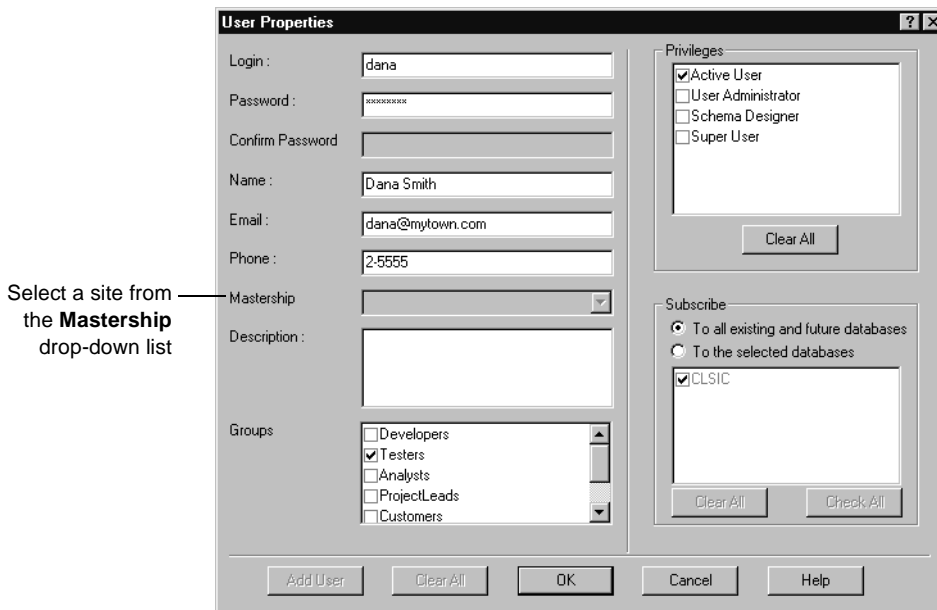
Changing the mastership of a user involves three tasks:

- Changing mastership. Do this at the site that currently has mastership of the user
- Synchronizing the replicas after mastership changes.
- Upgrading the user database with the mastership change. Do this at the new mastering site.

Changing mastership

To change the mastership of a user:

- 1 In the User Administration dialog box, open the User Properties dialog box for the desired user and select a replica from the Mastership drop-down list:



- 2 Notify the user administrator at the new site that mastership of a user has changed. The user administrator at the new site must upgrade that user database with the new user information.

Synchronizing the replicas after mastership changes

See your Administrator's Guide for Rational ClearQuest MultiSite for information on synchronizing replicas.

Upgrading the user database with the mastership change

The user administrator at a site that acquires new mastership of a user must upgrade the user database with the new user information.

- In the User Administration dialog box, select **DB Actions > Upgrade**. For more information, see "Applying schema changes to the user database" on page 144.

ClearQuest security features allow you to control user access to specified records. You can hide specific records from some users while allowing other users to view or change the same records. Using ClearQuest security features, you can protect your data and ensure that only authorized users view or change records.

These security features are particularly valuable when more than one user group has access to the same database; if, for example, you allow more than one group to submit defects to the same database; or, when multiple projects share the same database. You can open your production database so that a variety of customers can submit defects and enhancement requests to your database, while at the same time, preventing customers at the various companies from seeing the records submitted by customers at other companies or records submitted by your internal users.

This chapter covers the following topics:

- Hiding records in ClearQuest
- Security example
- Using ClearQuest's other security features

Note: To implement ClearQuest security features, you need to know how to do the following:

- Work with ClearQuest schemas, including checking schemas in and out of the schema repository, adding fields, and applying schema changes to a user database. See Chapter 3, "Working with ClearQuest schemas" and Chapter 4, "Customizing a schema."
- Administer users and user groups. See "Administering users" on page 135.
- Add fields to record forms. See "Working with forms" on page 89.

Hiding records in ClearQuest

ClearQuest security features work by restricting user access to records in a database based on membership to user groups. Record hiding is accomplished by placing a *security context field* in the record type of the records you want to restrict access to. The security context field references a *security context record* containing data that determines which users can see or change the controlled record.

For example, in order to control which customers are allowed to see defects, you might place a field called “customer_defects” in the Defects record type and reference this field to the “Customer” record type. You would then assign user groups to each customer record, which grants these groups privileges to see defect records that refer to the customer record. Only users who are in the group list of the security context record will be able to see the controlled record.

Hiding records in ClearQuest involves the following procedures:

- Deciding which record types to control
- Creating user groups according to security context
- Deciding which record type to use as the security context
- Creating a security context field
- Submitting security context records
- Editing records to allow user access

Deciding which record types to control

Decide which record type to use as the controlled record type. The controlled record type is the record type you want to hide or restrict access to. For example, to restrict access to defects in your ClearQuest system, you would use the Defect record as the controlled record type.

Creating user groups according to security context

Create new user groups or organize existing user groups according to the security context you want to use. Create user groups that align with your user access privileges. Then, assign users to the groups.

For example, to base security access on specific customers, you might create customer groups, one for each type of access permission. You can use existing groups or create new groups.

Note: If you add more than one security context field, you only need to be a member of one of the users groups to see records of that type.

Deciding which record type to use as the security context

Decide which record type to use as the security context. The security context record type is the record type referenced by the security context field. It contains user group information.

The security context record can be either state-based or stateless. It can be a record you create specifically for this purpose or it can be an existing record. For example, the Project or Customers record types.

Note: You cannot use a security context field to reference any system record types, such as history, users, groups, attachments, and so on.

For example, if you organize user groups by customer, you might use a Customer record type as the security context record. When a defect is associated with or references a particular customer record, ClearQuest allows only those users in the security context group list of that customer to see the record.

Creating a security context field

The security context field is a Reference type field in the controlled record that references the security context record type. You can create a new field to use as the security context field or use an existing field. You can add more than one security context field.

Note: The security context field must be a Reference field type.

Create a security context field of the Reference type in the controlled record type and reference this field to your security context record type. You can use an existing Reference type field or create a new field. Add the security context field to the forms of the controlled record type.

When you reference the security context record type, you also enable security by checking the **Security Context** checkbox for the field. This automatically creates a new tab called “Ratl_Security” in the forms of the referenced record type. The Ratl_Security tab contains a list control for Context Groups.

For example, create a reference type field called `customer_defects` in the Defect record type and reference this field to the Customer record type, ClearQuest adds a `Ratl_Security` tab to the Customer record type forms.

Submitting security context records

Submit records for each security context. If your security context is Customer, then you must submit a record for each customer.

For each record you submit, use the `Ratl_Security` tab to designate the user groups that can access the customer record by adding the group to the Context Groups list.

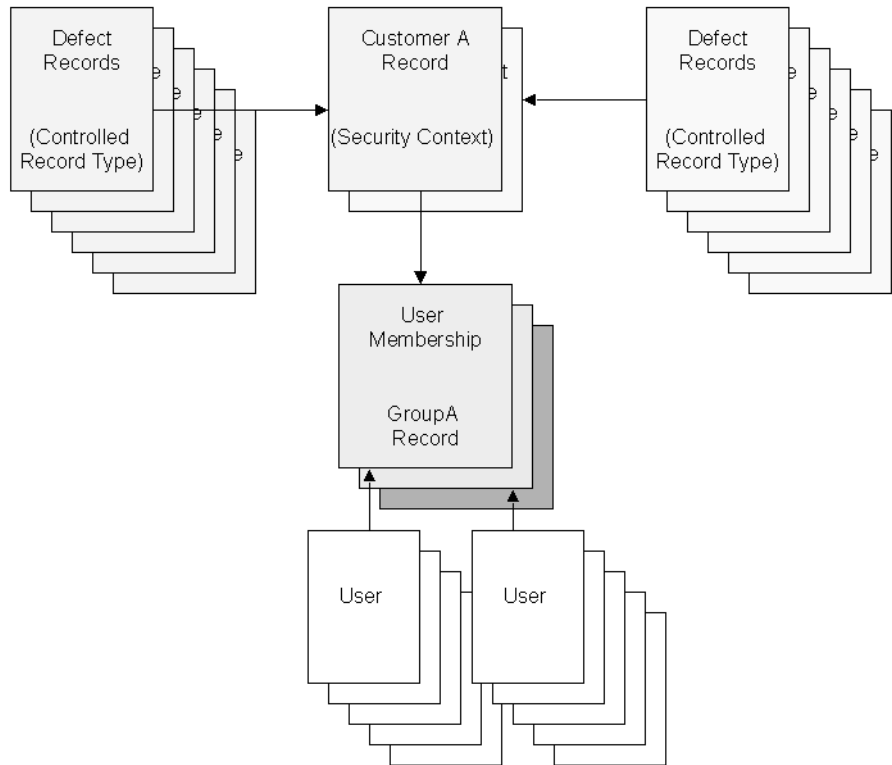
Warning: You must submit security context records and associate groups with each security context record to expose visibility of the controlled record. Any Defect records that have empty security context field values are hidden.

Editing records to allow user access

Edit your existing controlled records (for example, Defects) and select the applicable security context record, in this case the Customer record.

Designing a security system

These are the elements used to design your security system. User membership is defined by user groups. The user groups are associated with specific customers. Each defect record references one or many customer records, as illustrated below.



If you implement the design shown above in ClearQuest, this is what the end user will experience: User A logs into ClearQuest. He is a member of Group A, which is associated with Customer A. When user A runs a query or requests a record, ClearQuest will automatically filter all the defects based the security context "Customer A." Only defects belonging to Customer A will be presented.

Security example

This section provides an example of how to hide records in ClearQuest. In this example, you have three customers: Logic Equipment, Widgets Inc., and Modern Software. You want to control records of the Defect record type so that your customers can access your production database to:

- Submit defects
- Check the status of their defects, either by running a predefined query or by creating a new query
- Edit their existing defects

However, you do not want your customers to see defects submitted by other customers, or by your internal QE team.

When Modern Software logs in to your database, you do not want them to see defects filed by Widgets Inc., Logic Equipment, or by your own QE team. When a Modern Software customer creates a query in the ClearQuest client, the only results that customer will see is the information related to the defects submitted by their own Modern Software customers.

This example describes the following procedures:

- Checking out a schema
- Creating a security context field
- Adding the security context field to the form
- Applying the schema changes
- Creating the user groups
- Submitting the security context records
- Associating groups with each security context record
- Editing records to grant privileges to groups

Note: These procedures require various user access permissions, as described in the example that follows. You must have Super User privileges to complete all of the procedures listed in this example.

Checking out a schema

This example uses a schema based on the predefined DefectTracking schema, which contains Defect and Customer record types. This example assumes that the schema is checked out.

Creating a security context field

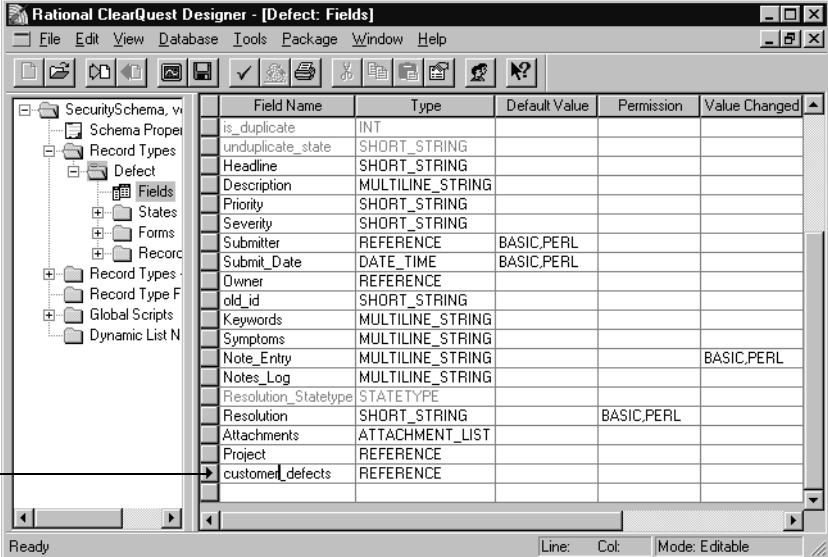
Note: You must have Schema Designer privileges to create a new field and add the field to the record form.

In this example, you want to control access to defect records, so you create a security context field in a Defect record type that references the Customer record. You create the field in the Fields grid, add the field to the record form, then apply the schema changes.

A security context field must be a Reference field type. You can add more than one security context field. If you add more than one security context field, you only need to be a member of one of the groups to see records of that type.

To create a Security Context field in the Defect record type:

- 1 In ClearQuest Designer, expand **Record Types > Defect** and double-click **Fields** to open the Fields grid.
- 2 In the Fields grid, create a new field named “customer_defects” and select **Reference** as the field type. (In your own security system, you can create a new field or use an existing Reference type field.)

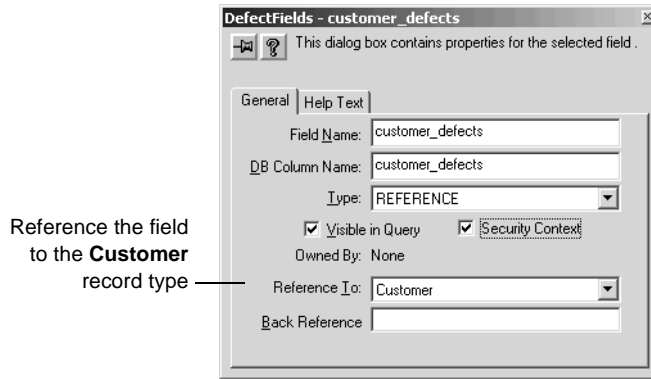


The screenshot shows the Rational ClearQuest Designer interface. The left pane displays a tree view with 'SecuritySchema, v1' expanded to 'Record Types' > 'Defect' > 'Fields'. The main pane shows a table with the following columns: Field Name, Type, Default Value, Permission, and Value Changed. The table contains various fields, and a new field 'customer_defects' is being added at the bottom with the type 'REFERENCE'. A text box on the left points to the 'REFERENCE' type selection.

Field Name	Type	Default Value	Permission	Value Changed
is_duplicate	INT			
unduplicate_state	SHORT_STRING			
Headline	SHORT_STRING			
Description	MULTILINE_STRING			
Priority	SHORT_STRING			
Severity	SHORT_STRING			
Submitter	REFERENCE	BASIC,PERL		
Submit_Date	DATE_TIME	BASIC,PERL		
Owner	REFERENCE			
old_id	SHORT_STRING			
Keywords	MULTILINE_STRING			
Symptoms	MULTILINE_STRING			
Note_Entry	MULTILINE_STRING			BASIC,PERL
Notes_Log	MULTILINE_STRING			
Resolution_StateType	STATETYPE			
Resolution	SHORT_STRING		BASIC,PERL	
Attachments	ATTACHMENT_LIST			
Project	REFERENCE			
customer_defects	REFERENCE			

- 3 Right-click the new customer_defects field and select **Field Properties**.

- 4 In the Field Properties dialog box, select the **Customer** record from the **Reference To** list.



Selecting the Customer record type from the Reference To list, enables the **Security Context** checkbox.

- 5 Check the **Security Context** checkbox.

When you check **Security Context**, ClearQuest Designer adds a tab called “Ratl_Security” to the Submit and default forms of the security context (Customer) record type. You will use this tab in the ClearQuest client to select the groups that can view the record. (In your own system, you can change the name of the Ratl_Security tab, if desired. See “Changing the name of a tab” on page 93.)

Note: You can add more than one security context field to a record type. For example, you might add a security context field that references the Customer record type and another security context field that references the Quality_Assurance record type. If you add customers to the Customer record type, and members of your Quality Assurance group to the Quality_Assurance record, then users in any of the group lists for those record types have access to the records under security control.

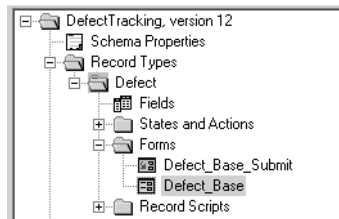
You might want to include a hook to populate the field automatically, based on the user who logs in. This will ensure that the field contains a valid value. You might also consider preventing users from performing certain actions. For example, you might allow only internal users to close a defect, and restrict your customers from deleting records. For more information, see “Using ClearQuest’s other security features” on page 169.

Adding the security context field to the form

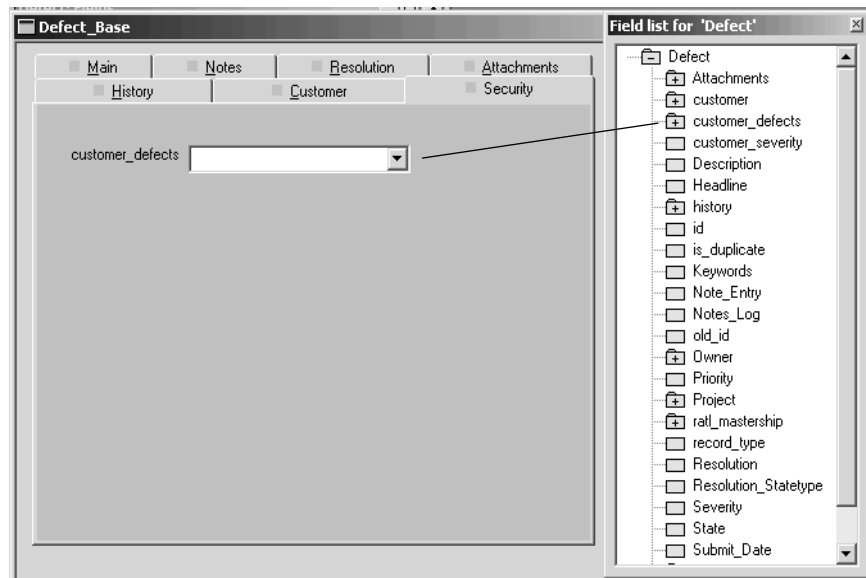
After creating the `customer_defects` field in the Fields grid, you must add it to the Defect record form.

To add the new `customer_defects` field to the Defect record form:

- 1 In ClearQuest Designer, expand **Record Types > Defect > Forms** and double click **Defect_Base** to open the form.



- 2 In the Field List, select the `customer_defects` field and drag it onto your form.



Applying the schema changes

After adding a new field, you must check the schema back into the schema repository and apply the schema changes to the user database. Once you perform these steps, these changes cannot be undone. For more information, see “Applying schema changes to a user database” on page 41.

Creating the user groups

Note: You must have User Administrator privileges to create users and groups.

Next, you create the groups to be associated with the Customer security context record, add users to the groups, and update the user database with the new user information. For this example, you create user groups for Widgets Inc., Modern Software, and Logic Equipment, then add users to these groups.

Note: In your own security system, you can also use existing groups. You may want to create additional groups, such as a group that can view all records submitted by internal users, one that can view all records submitted by all companies, or one that can view all records, regardless of whether they were submitted internally or by customers.

For information on creating groups, see “Creating a new user group” on page 148 and “Adding users to a group” on page 149.

Submitting the security context records

Note: You must have Security Administrator privileges to submit security context records.

Next, you submit a Customer record for each company that you want to have access to your database, Widgets Inc., Modern Software, and Logic Equipment.

In the ClearQuest client:

- 1 Select **Actions > New > Customer** to open the Submit Customer dialog box.
- 2 Submit customer records for Widgets Inc., Modern Software, and Logic Equipment.

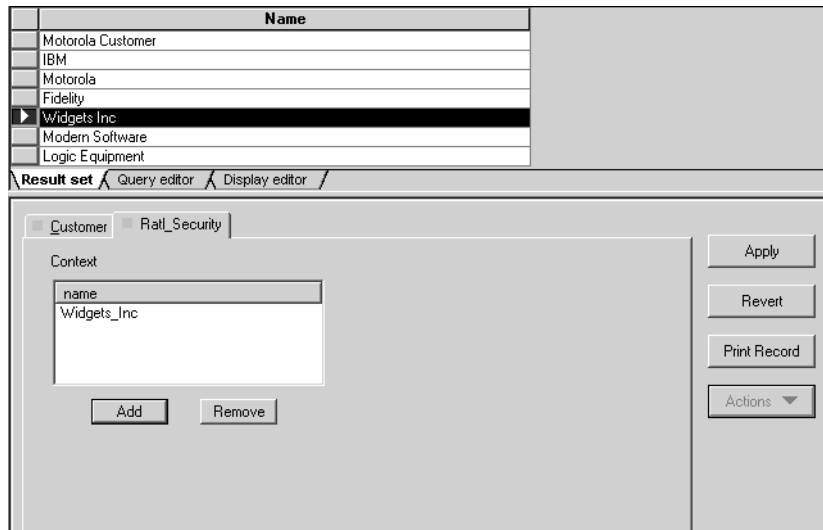
You can also create groups that can view all records. If you create a group that can view all records, add this group to each customer record.

Associating groups with each security context record

Next you must associate specific groups with each security context record. In this example, you select the user groups to associate with the customer records submitted for Widgets Inc., Logic Equipment, and Modern Software. These groups contain the users to whom you wish to grant privileges to view and change records. You must have Security Administrator or Super User privileges to select the groups.

To associate the groups with the Customer records:

- 1 In ClearQuest Client, create and run a query called “All Customers” that displays a list of all Customer security context records.
- 2 Open the Widgets Inc. customer record and click the **Ratl_Security** tab. Select **Action > Modify**.
- 3 On the Ratl_Security tab, select the Widgets Inc. group and click **Add**. Click **Apply**.
- 4 Repeat steps 3 and 4 for the Modern Software and Logic Equipment records.

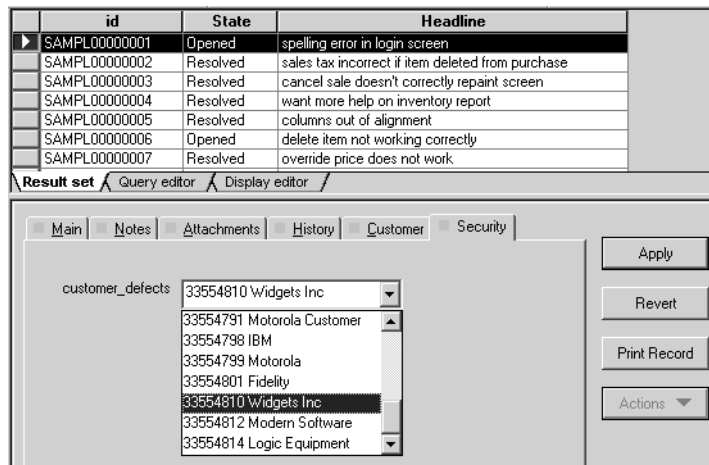


Editing records to grant privileges to groups

Next, you edit each defect record that you want customers to access, assigning a customer to the `customer_defects` field. This will give the Logic, Widgets, Modern groups access to the record. This step assigns the value of the security context record to the security context field.

In the ClearQuest client:

- 1 Log in to the database containing the defect records you want to control.
- 2 Run a query on “All Defects,” in order to see all of the defect records.
- 3 Select and edit the record you want to control. For example, select the defect record, “spelling error in login screen.”



- 4 In the `customer_defects` list, select the customer you want to have access to this record. For example, select Widgets Inc. to give users in the Widgets_Inc group access to this record.
- 5 Edit each record to grant privileges to the Logic Equipment and Modern Software groups as well.

This completes the example of record hiding in ClearQuest. In this example, a Widgets Inc. customer will be able to log in to your database and perform the following tasks:

- Edit existing defect records. The customer choice list will show only Widgets Inc.
- Run a query, chart, or report for the customer record type and see only the Widgets Inc. record.

Using ClearQuest's other security features

You can protect your data and ensure that only authorized users view or change records in ClearQuest by:

- Restricting access to fields
- Restricting access to actions
- Restricting access to dialog tabs
- Adding password protection

Restricting access to fields

You can restrict user access to a field by:

- Defining field behavior as Read Only. Users can view, but not change, the contents of a read-only field.
- Using a field Permission hook to determine the level of user access.

Restricting access to actions

You can add an Access Control hook that restricts access based on criteria you specify, or you can select the user groups that can perform the action.

Restrict access to dialog tabs

You can restrict access to multiple fields by placing them on a dialog tab and restricting access to the tab to selected user groups.

Adding password protection

You can add a password field to a form using a Text Box control to suppress echoing of typed characters in the field. When the user types in the field, each character is replaced by an asterisk (*).

To add a password field to a form:

- 1 Add a Text Box control.
- 2 Double-click the Text Box control to display the property sheet.
- 3 Click the Extended tab and select **Password/no echo style**.

Using hooks to customize your workflow

8

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:

- Understanding hooks
- Working with field hooks
- Working with action hooks
- Execution order of field and action hooks
- Working with record scripts
- Working with global scripts
- Writing external applications
- Using the ClearQuest API
- Common API calls
- Finding text in hook scripts
- Examples of common hooks

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 9, “Administering ClearQuest Web.”

Understanding 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.

Important hook considerations

Consider the following when you plan your hooks:

- 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 38.
- Be aware that hooks run with Super User privileges; and therefore, are not subject to the usual access control or field behavior restrictions. This means that you can use hooks to set and reset values in fields that are normally read-only. (You cannot reset ClearQuest system fields, 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 219.
- 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.

Writing scripts

Field and action hooks support the addition of a script written in VBScript or Perl. If you are creating global scripts or record scripts, you must use VBScript or Perl.

For convenient storage and reference, you can write both VBScript and Perl scripts in the same schema. However, a schema only runs scripts in the language specified (see “Selecting a scripting language” on page 38).

Write or edit scripts in the Script editor. When you define a new script, ClearQuest Designer automatically adds the calling syntax for that hook to the Script editor window. The calling syntax appears in gray and cannot be edited. ClearQuest Designer also adds sample body text that you can edit as necessary. (The initial body text is commented out and will not execute unless you remove the comment markers.)

Note:

- ClearQuest does not support nested Perl scripts.
- If you are using the ClearQuest Web client, you must use VBScript. ClearQuest Web does not support the use of Perl scripts.

- ClearQuest Designer provides a different Script editor for VBScript and Perl, and notes the editor type in the title bar of the ClearQuest window. Be sure you are in the correct editor before writing your code.

Operating context for using scripts

ClearQuest simplifies the process of writing VBScript and Perl hooks by providing you with a consistent operating context. Before a ClearQuest hook even calls a VBScript or Perl script, ClearQuest creates a Session object and logs the user in to the system. Because all hooks (including global scripts) are executed from the context of the current record, ClearQuest automatically provides you with an Entity object corresponding to that record. (Global scripts share the Entity object associated with the hook that called it.)

Within a script, you can call the methods of Entity without specifying a leading identifier. For example, you can call the GetSession method of Entity in the following manner:

```
set curSession = GetSession
```

When you call methods in this manner, ClearQuest automatically assumes you are calling a method of the implicit Entity object passed to the hook. If you want to refer to this Entity object explicitly, you can use the name of the record type as an identifier for the object. Using this identifier can help clarify code that manipulates more than one Entity object at a time, as in the following example, which marks one entity as a duplicate of another:

```
set curSession = GetSession

set duplicateEntity = curSession.GetEntity("defect",
"BUGDB00000037")

curSession.MarkEntityAsDuplicate duplicateEntity, defect,
"duplicate"
```

Because scripts can impact the behavior of a field, you should carefully design and test your hook code. For example, if a hook requires that a field contain some sort of value, the field becomes mandatory even if its behavior is not set to MANDATORY.

Working with 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 types of 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 “Creating a choice list for a field” on page 180.
Default Value	Sets the initial value of the field. This hook is called at the beginning of a Submit action. You can write a default-value hook with a script subroutine. 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).

For instructions on how to specify field hooks, see “Customizing fields by adding hooks” on page 74. To learn about the sequence in which hooks run, see “Execution order of field and action hooks” on page 191.

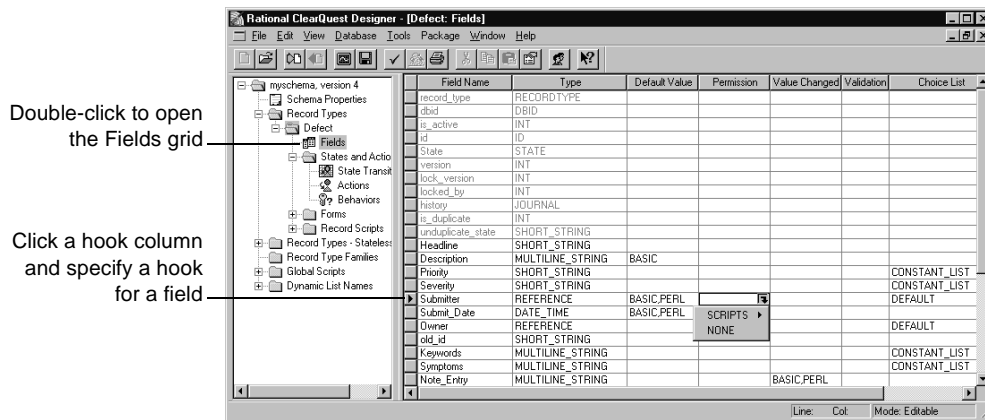
Adding a field hook

You can add field hooks to customize the behavior of fields by providing such features as choice-lists, field validation, default values, and permissions. ClearQuest executes your hooks based on the rules specified in the section “Execution order of field and action hooks” on page 191.

To add a field hook:

- 1 In the Workspace, expand **Record Types**, then expand the desired record type and double-click **Fields** to display the Fields grid.

The fields grid contains a column for each type of predefined field hook.



- 2 Click in a hook column for the desired field, then click the down arrow and select **Scripts > Basic** or **Perl**.

If Instant Editing Mode is enabled, ClearQuest automatically opens the appropriate script editor. If Instant Editing Mode is disabled, you must double-click the cell to open the editor.

Note: The editor that ClearQuest opens depends on the type of hook you select. For example, if you select **SCRIPTS > BASIC**, ClearQuest opens the Basic Script editor. However, if you choose **CONSTANT LIST**, ClearQuest opens the Choice List dialog.

- 3 Edit the hook. ClearQuest automatically adds the calling syntax for that hook to the script editor. The calling syntax appears in gray and cannot be edited.
- 4 Select **Hooks > Compile** to verify that your scripts are error free. ClearQuest displays any errors in the Validation pane at the bottom of the Workspace.

5 Close the script editor. ClearQuest Designer saves your changes.

Note: If you add a permission hook to a field, make sure you adjust the values for that field in the Behaviors grid. The permission hook will not work unless the field's behavior is set to USE_HOOK.

Editing a field hook

To edit a field hook:

- 1 In the Workspace, expand **Record Types**, then expand the desired record type and double-click **Fields** to display the Fields grid.
- 2 Double-click the column of the field whose hook you want to edit. Open the hook editor. (Alternatively, you can right-click the column and select **Edit hook_type** from the shortcut menu. For example, to edit a script, select **Edit Script**.)
- 3 Edit the hook.
- 4 Select **Hooks > Compile** to verify that your scripts are error free. ClearQuest displays any errors in the Validation pane at the bottom of the Workspace.
- 5 Close the editor once your script compiles correctly. ClearQuest Designer saves your changes.

Deleting a field hook

To edit a field hook:

- 1 In the Workspace, expand **Record Types**, then expand the desired record type and double-click **Fields** to display the Fields grid.
- 2 Click the cell in the Fields grid that contains the hook you want to delete and select **NONE** from the drop-down list. (If the field's type is REFERENCE or REFERENCE_LIST and the hook is a choice-list hook, select **DEFAULT**.)
- 3 Click **Yes** to confirm the deletion.

Note: After you delete a hook and then check in the version of your schema you are working on, you cannot restore the deleted hook. However, if the hook was stored in a previous version of the schema, you can open that previous version and copy the hook into any schema.

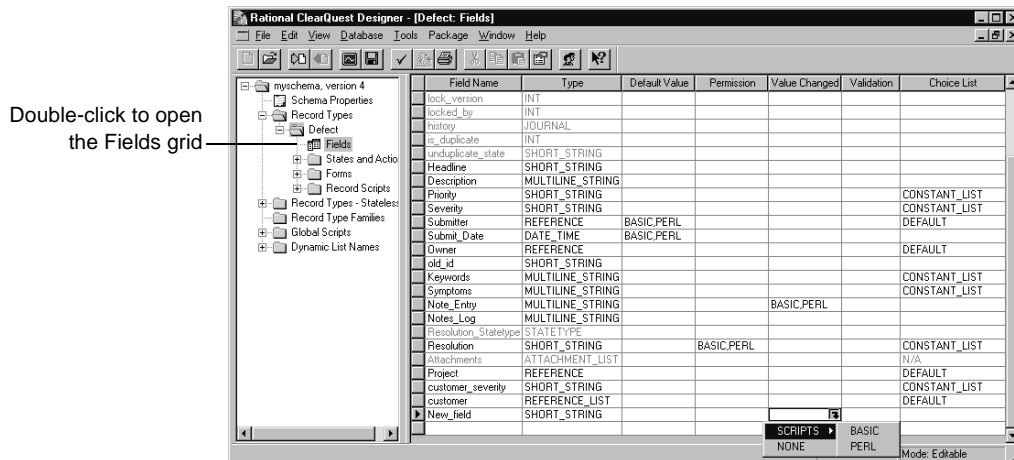
Creating a dependency between fields

You can create a dependency between two fields so that when the value of the parent field changes, the value of the child field (the dependent field) also changes.

Note: Plan field dependencies carefully when designing your schema. Dependent fields require the use of hooks, which can introduce runtime errors if not written correctly. Be sure to test script hooks thoroughly before making them available to your users.

To create a dependency between two fields, add a Value Changed hook to the parent field:

1 Display the Fields grid.



2 Click the **Value Changed** cell of the parent field.

3 Click the down-arrow icon and select **SCRIPTS > BASIC** or **SCRIPTS > Perl**.

If Instant Edit Mode is enabled, ClearQuest Designer automatically opens the Script editor. To enable or disable instant editing, select **Edit > Instant Edit Mode**. Double-click the cell to open the Script editor if it does not automatically appear.

Note: ClearQuest Designer indicates the type of editor you are using in the title bar of the Designer window. Be sure you are using the correct editor before adding or editing your code.

- 4 In the Script editor, write a script that gets the value of the parent field and uses it to set the value of the child field. (For an example of setting a parent value based on child values, see “Action hook for setting the value of a parent record” on page 212.)
- 5 When you finish editing your script, select **Hooks > Compile** to check the syntax of your script code.

Enabling dependent fields for ClearQuest Web

If you want a form with dependent fields to display in ClearQuest Web, you must do the following:

- 1 When you add the field to the record form, use one of the following form controls for the parent field and its dependent fields. You can mix and match.
 - Drop-down list box
 - Combo box,
 - Drop-down combo box

For information on controls, see “Working with form controls” on page 96.

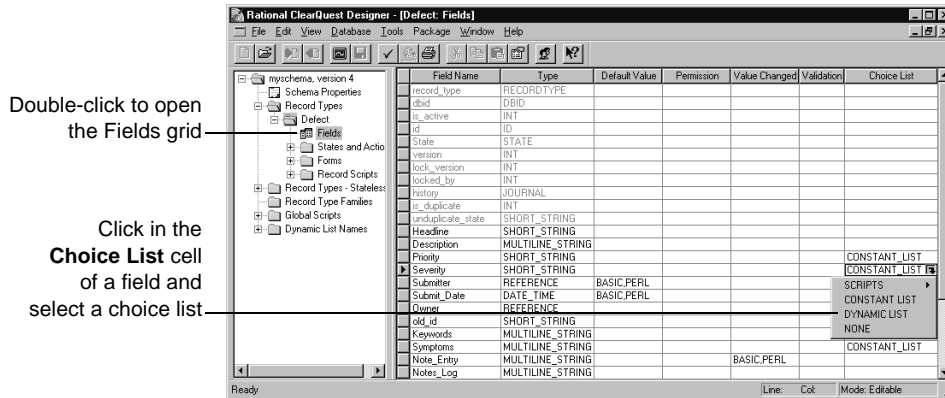
- 2 After adding the controls to the record form, right-click the control for the parent field and select **Properties** from the shortcut menu.
- 3 In the Web Dependent Fields tab, select the appropriate child fields from the **Available** list and add them to the **Selected** list.

Creating a choice list for a field

You can use a choice-list hook to provide a list of options for a field. You can provide a constant list of options, use a script to generate a list of legal values, or set up a dynamic list.

To create a choice list for a field:

- 1 Open the Fields grid.



- 2 Click the **Choice List** cell of the desired field.

- 3 Click the down-arrow icon and select one of the following:

- Select **SCRIPTS** to write a script for a choice list. You can define a dependent list whose values change when the user selects a certain value in another form control. For an example, see “Hook for creating a dependent list” on page 207.
- Select **CONSTANT LIST** to provide a set of values for a choice list.
- Select **DYNAMIC LIST** to create a choice list whose values can be changed from the ClearQuest client by anyone with Schema Designer, Dynamic List Administrator, or Super User privileges. Dynamic-list field hooks are convenient for lists that you want to change frequently, because they enable you to update the list values without changing the schema. See “Creating a dynamic choice list” on page 181.
- Select **NONE** (the default) for Reference and Reference List field types only. The system presents a choice of all the active records of the given type as a default choice list for these field types.

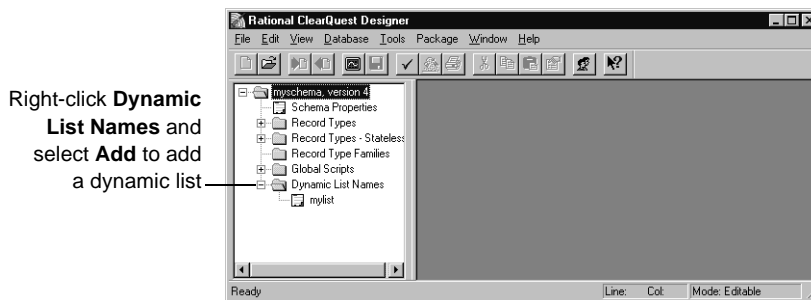
Creating a dynamic choice list

A dynamic list can be a drop-down list such as a list of states. Dynamic lists are helpful when you want to add values to a list without editing the schema. Users with Schema Designer, Dynamic List Administrator, or Super User privileges can add values to a dynamic choice list from any ClearQuest client.

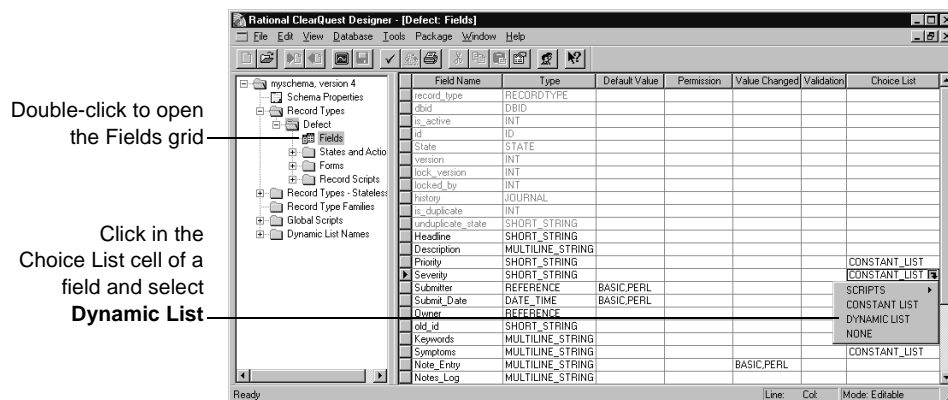
Note: You can use the same dynamic list for multiple fields, regardless of record type. Updating the contents of such a field updates all fields associated with that dynamic list.

To create a dynamic list for a field you first create the list and associate it with a field in ClearQuest Designer, then define the values for the list in ClearQuest client:

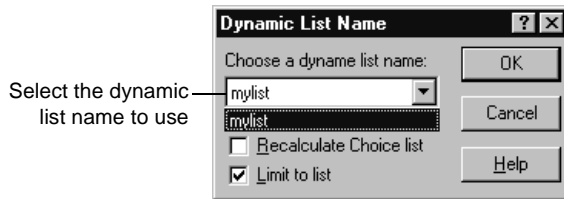
- 1 In the Workspace, right-click **Dynamic List Names** and select **Add** from the shortcut menu.
- 2 Type a name for the dynamic list.



- 3 In the Fields grid, click the Choice List cell for the desired field, click the drop-down arrow icon and select **DYNAMIC LIST**.



- 4 In the Dynamic List Name dialog box, select the dynamic list name you just created.



- 5 You can select **Recalculate Choice list** or **Limit to list** if desired. See “Defining choice list behavior” on page 183.
- 6 Click **OK**.

To save these changes you should first test the schema, check the schema back into the schema repository and then apply the schema changes to the user database. See “Overview of schema procedures” on page 32.

- 7 To define the values for a dynamic list, use ClearQuest client. See “Editing a dynamic list” on page 182.

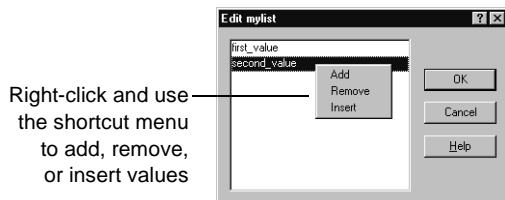
Editing a dynamic list

You first add a dynamic list in ClearQuest Designer (see “Creating a dynamic choice list” on page 181), then you define the values of the dynamic list in ClearQuest client.

You must have Schema Designer, Dynamic List Administrator, or Super User privileges to edit a dynamic list. (See “ClearQuest user privileges” on page 138.)

To edit a dynamic list:

- 1 In the ClearQuest client, select **Edit > Named Lists** and select the list you want to edit.
- 2 Use the Edit Named List dialog box, to Add, Remove, or Insert a value.



- To add a value to the list, right-click in the list box and select **Add**, then type the name of the value you want to add and press **Enter**. Continue adding values as desired.
 - To remove a value from the list, right-click the value and select **Remove**.
 - To insert a value into the list, right-click an existing value and select **Insert**. This inserts a new line above the existing value. Type the name of the value you want to insert and press **Enter**.
- 3 When you are done, click **OK** to close the Edit Named List dialog box.

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 select **Limit to list**, users can enter a value not in the choice list.
- Select **Recalculate Choice list** to have ClearQuest reinitialize the list at runtime, if it depends on changes to another value. For a scripted choice list, this recalculate 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.

Specifying a default value for a field

In some cases, you may want to specify a default value for a field. You can specify a constant for the default value or you can use a script to calculate a default value.

To specify the default value for a field:

- 1 In the Fields grid, click the **Default Value** cell of the field you want to modify.
- 2 Click the down arrow and select the type of hook you want to use.

To specify a constant value, select **CONSTANT** and then enter a value in the **Constant** field.

To write a script, select **SCRIPTS > BASIC** or **SCRIPTS > PERL** and then enter the code to set the default value after the lines that read (in Perl, the lines begin with #):

```
REM Set the initial value of the field here. Example:  
REM SetFieldValue fieldname, 12345
```

To remove a hook, select **None**.

Note:

- For scripts, you should compile the script to expose syntax errors in the code before associating it with the field.
- For an example of a default value script, see Field default value hook example in the ClearQuest Designer online Help.

Validating user input in a field

You can use a validation hook to verify that a user has typed valid information in a field. If the user types invalid information in a field, ClearQuest prompts the user for valid information.

- 1 In the Workspace, expand **Record Types** and the desired record type, then double-click **Fields** to open the Fields grid.
- 2 In the Fields grid, click the **Validation** cell of the field you want to modify, then click the down arrow icon to display a list of available hooks and select **SCRIPTS > BASIC** or **SCRIPTS > PERL**.

If Instant Editing Mode is enabled, ClearQuest Designer automatically opens the script editor. If Instant Editing mode is disabled, double-click the **Validation** cell of the field to open the script editor.

Note: BASIC and PERL each has its own script editor. ClearQuest Designer indicates the type of editor in the title bar of the Designer window. Be sure you are in the correct editor before editing code.

- 3 Enter the code to validate user input after the lines that read (in Perl, the lines begin with #):

```
REM Return a non-empty string explaining why the
REM field's current value is not permitted.
REM Or, if it is valid, return an empty string value.
REM Example:
REM Dim value_info
REM Set value_info = GetFieldValue(fieldname)
REM If Len(value_info.GetValue()) < 10 Then
REM resolution_date_Validation = "Must be at least 10 chars
long"
REM End If
```

For example:

If the field name is “user_number” and its type is INT, the code ensures that users enter a value between 1 and 100:

```
REM Return a non-empty string explaining why the field's
current value is not permitted
REM Or, if it is valid, return an empty string value.

value = GetFieldValue(fieldname).Get Value()
if Not IsNumeric(value)
    user_number_Validation="Field does not contain a number."
Else If (value < 1) or (value > 100) then
    user_number_Validation="User number must be between 1 and
100."
end if
```

- 4 Select **Hooks > Compile**. This compiles the script and checks to make sure that there are no syntax errors.
- 5 Close the Script editor.

Note:

- When you define a new BASIC or PERL hook for a field or action, ClearQuest automatically adds the calling syntax for that hook to the script editor window. When you open the window, the calling syntax appears in gray and cannot be edited.
- Because hooks can impact the behavior of a field, you should carefully design and test hooks before making them available to users. For example, this user input hook example effectively makes the user_number field a mandatory field, regardless of the setting in the Behaviors grid.

Working with 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 types of 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 "Restricting user access to actions" on page 151.	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.
Validation	Validates the field values you specify. If the user types invalid data, ClearQuest prompts the user for valid data. 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. Validation hooks must use a script. See "Validating user input in a field" on page 185.	When the user commits the action.

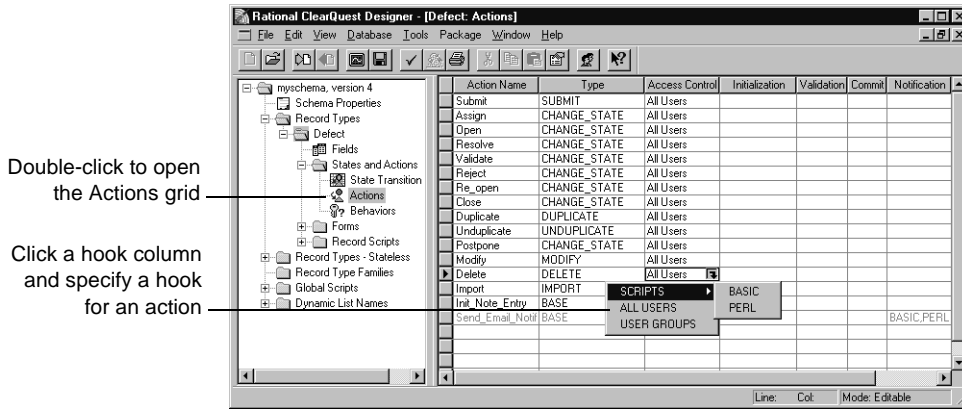
Action hook	Use	Executed
Commit	<p data-bbox="501 210 1011 279">Links an action on multiple records into a single transaction (for example, resolving all the duplicates of a change request when the original is resolved).</p> <p data-bbox="501 314 1011 401">Updates a set of external data sources so that 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 data-bbox="501 435 1011 479">You can write a commit hook as a VBScript or Perl subroutine.</p>	<p data-bbox="1025 210 1225 296">Just before ClearQuest commits the transaction to the database.</p>
Notification	<p data-bbox="501 505 1011 574">Starts a post-commit action that notifies users when an action is performed. See Chapter 10, "Administering ClearQuest E-mail."</p> <p data-bbox="501 609 839 631">Notification hooks must use a script.</p>	<p data-bbox="1025 505 1186 574">After ClearQuest commits the transaction.</p>

To learn about the sequence in which hooks run, see "Execution order of field and action hooks" on page 191.

Adding an action hook

To define an action hook, use the Actions grid.

- 1 In the Workspace, expand **Record Types**, then expand the desired record type.
- 2 Expand **States and Actions** and double-click **Actions** to display the Actions grid.



- 3 Click in the column corresponding to the hook you want to add and select **Scripts > Basic** or **Perl** from the drop-down list.

If you are adding an Access Control hook, you can also select **All Users** to provide access to all users or select **User Groups** to provide access to users of specified user groups.

If Instant Editing Mode is enabled, ClearQuest automatically opens Script editor. If Instant Editing Mode is disabled, you must double-click the cell to open the editor.

- 4 Edit the hook. ClearQuest automatically adds the calling syntax for that hook to the Script editor. The calling syntax appears in gray and cannot be edited.
- 5 When you are finished, close the Script editor.

Editing an action hook

To define an action hook, use the Actions grid.

- 1 In the Workspace, expand **Record Types**, then expand the desired record type.
- 2 Expand **States and Actions** and double-click **Actions** to display the Actions grid.
- 3 Double-click in the cell containing the hook you want to edit. ClearQuest Designer opens the script editor.

Note: ClearQuest Designer provides a different Script editor for VBScript and Perl, and notes the editor type in the title bar of the ClearQuest window. Be sure you are in the correct editor before writing your code. If you double-click a cell that has scripts written in both languages, both Script editors open one on top of the other.

- 4 Edit the hook. ClearQuest automatically adds the calling syntax for that hook to the Script editor. The calling syntax appears in gray and cannot be edited.
- 5 When you are finished, close the Script editor.

Deleting an action hook

Note: Once you delete a hook, you cannot restore the changes you made to the hook unless they are stored in a previous version of your schema.

To delete an action hook:

- 1 In the Workspace, expand **Record Types**, then expand the desired record type.
- 2 Expand **States and Actions** and double-click **Actions** to display the Actions grid.
- 3 Click in the cell containing the hook you want to delete and select **None** from the drop-down menu. For access-control hooks, you must select either **ALL USERS** or **USER GROUPS**.
- 4 Click **Yes** to confirm the deletion.

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 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 checks user access and 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).
- 5 The **Field Validation** hook validates the contents of each field, except for an import action in which no validation occurs. Field validation hooks cannot validate non-existent values, such as a mandatory field in which the value has not been set.
- 6 The **Field Choice List** hook runs for each field that uses the **Recalculate Choice List** option. If there are dependent fields and you selected **Recalculate Choice List** when you created the fields, ClearQuest runs the Field Validation hook and then the Field Choice List hook for each field, until all changed fields have been set and validated.

If a Field Value Changed hook calls the SetFieldValue method of the Entity object, the VALUE_CHANGED hook for that field runs at the time of the SetFieldValue call.

When a field 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, unless the field is marked as having dependent fields. See “Using hooks in ClearQuest Web” on page 219.

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.

Working with 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.

For more information, refer to the online *ClearQuest API Reference*.

Understanding record scripts

Record scripts are a generic form of hook that are called in response to an event on a ClearQuest form or from other hooks. Typically, record scripts are used to implement an action that you want to perform in response to a click event on a push button or on a context menu item associated with a particular field on a ClearQuest form.

Record scripts run in the context of the currently selected record.

All record scripts have the same syntax:

```
Function RecordName_HookName(param)
    ' param As Variant
    ' RecordName_HookName As Variant
    ...
End Function
```

When calling a record script from another hook, the parameter you pass into the hook is a simple Variant containing whatever data is appropriate. If your hook returns information to the calling hook, return that information in a Variant.

When associated with a form control, the parameter passed into the method contains an instance of the EventObject class. This instance contains information about the event that caused the hook to be called. (See “Form control events” on page 196.) ClearQuest does not expect a return value from record scripts when they are associated with form controls. A non-null return value from a record hook is interpreted as an error and can be viewed in a message box by the ClearQuest client.

Record scripts can be associated with push buttons, text fields, and lists. When associated with a button, clicking the button automatically causes the hook to be executed. When associated with text fields and list-related controls, ClearQuest adds the hook to the control's shortcut menu. The user can then select the hook from the shortcut menu to execute it.

You can also associate a record script with an action whose type is RECORD_SCRIPT_ALIAS. This allows you to put a custom button on the Action menu of a ClearQuest form.

See also

- “Writing scripts” on page 173
- “Operating context for using scripts” on page 174

Using record scripts on ClearQuest Web

- ClearQuest web looks at the return value of a record script invoked by a push button. If the return value is a string, it is considered to be an error message and the hook will fail.
- If you don't explicitly set the return value of the record script function, it will return a null or empty value that will indicate to ClearQuest web that the hook was executed successfully.
- In the web, record hooks run on the ClearQuest web server and are active only when you enable them through the Enable for web check box on the Extended Tab of the Control's Property sheet. Because they execute on the server, you shouldn't call Windows routines that cause a window to display, as they would display on the server instead of the Web client. The ClearQuest Web Server sets a Session.NameValue called CQ_WEB when it starts a server session so that your scripts can be made web-aware.

Form control events

When a record script is triggered by a form control, ClearQuest passes the record script an EventObject object as its parameter. This object contains information about the type of event that occurred. Different controls can generate different types of events, including button clicks, item selections, and so on. You must use the information in the EventObject object to determine how to handle the event.

ClearQuest generates the following types of events for form controls:

- **Button-click:** Indicates that the user clicked a push button control.
- **Context menu item-selection:** Indicates that the user invoked the hook from a context menu.

The following table lists the supported type of event for each control and the extra information provided by the EventObject. The constants listed under the supported event type column are part of the EventType enumerated type.

Control type	Supported event type	Additional information
Push button	AD_BUTTON_CLICK	Button name
Combo box	AD_CONTEXTMENU_ITEM_SELECTION	Null string
Drop-down list box	AD_CONTEXTMENU_ITEM_SELECTION	Null string
List box	AD_CONTEXTMENU_ITEM_SELECTION	Current field value selection
List view	AD_CONTEXTMENU_ITEM_SELECTION	Current field value selection
Text box	AD_CONTEXTMENU_ITEM_SELECTION	Current field value selection
Drop-down combo box	AD_CONTEXTMENU_ITEM_SELECTION	Null string

Adding a record script to a record type

Add a record script to a record type through the record type's Record Scripts folder or Actions grid. To add a record script through the Actions grid, see "Adding a new action" on page 84.

To add a record script through the Record Scripts folder:

- 1 In the Workspace, expand **Record Types**, then expand the desired record type.
- 2 Expand the **Record Scripts** folder.
- 3 Right-click the desired language folder and select **Add** from the shortcut menu. ClearQuest Designer adds a script to the folder, highlighted so you can change the name.
- 4 Type a new name for the record script.
- 5 Double-click the new record script name to open the Script editor.
- 6 Edit the record script. Type your code in place of the line that says:

```
REM added your hook code here
```
- 7 When you are finished editing your code, select **Hooks > Compile** to make sure there are no syntactical errors. When you compile, ClearQuest automatically saves your hook code.
- 8 Close the Script Editor window.

For each record script you define, the Script editor creates a stub that includes the record name, hook name, and required syntax. Each hook has the following format:

```
Function RecordName_HookName(param)
    ' param as EventObject
    REM Your code here
End Function
```

The parameter passed to a record script is an instance of the `EventObject` class. This instance contains information about the event that triggered the call to the hook. It contains the type of the event, the control that invoked it, and additional information depending on the type of event. For more information, see "Form control events" on page 196.

Editing a record script

To edit a record script:

- 1 In the Workspace, expand **Record Types**, then expand the desired record type.
- 2 Expand the **Record Scripts** folder until you see the record script you want to modify.
- 3 Double-click the Record script you want to edit. (Alternatively, you can right-click the hook and select **Open** from the shortcut menu.)
- 4 Edit the record script as desired. If this is a new Record script, type your code in place of the line that says:

```
REM added your hook code here
```
- 5 When you are finished editing your code, select **Hooks > Compile** to make sure there are no syntactical errors. When you compile, ClearQuest automatically saves your hook code.
- 6 Close the Script Editor window.

Deleting a record script

To delete a record script:

- 1 In the Workspace, expand **Record Types**, then expand the desired record type.
- 2 Expand the **Record Scripts** folder until you see the record script you want to delete.
- 3 Right-click the record script you want to delete and select **Delete** from the shortcut menu.
- 4 Click **Yes** to confirm that you want to delete the script.

Working with 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.

Understanding global scripts

ClearQuest supports the use of global scripts, which are functions whose use is global to the entire schema. You can call global scripts from any other type of hook, including record and global scripts. Unlike hooks associated with a particular record, global scripts do not have a predefined syntax. You control whether the hook is a function or subroutine and the number and kind of parameters the hook accepts and returns.

The only way to invoke a global script is to call it from another script. Ultimately, however, a call to a global script must be traced back to a call to a record-based hook such as a field hook, action hook, or record script. Because calls to a global script can always be traced to a record-based hook, each global script is provided with an implicit entity object passed to the record-based hook that originated the call.

Note

- You can write your scripts using both VBScript and Perl, but one language type cannot call another language type. When writing a global script, be sure to write it in the same language as the script that calls it.
- See “Writing scripts” on page 173 and “Operating context for using scripts” on page 174.

Creating a global script

To create a global script:

- 1 In the Workspace, expand the **Global Scripts** folder.
- 2 Right-click the desired language folder, **Basic** or **Perl**, and select **Add** from the shortcut menu.
- 3 Type a name for the new global script.
- 4 Double-click the new name to open the Script editor.
- 5 Edit the global script as desired. Type your code in place of the line that reads:

```
REM TODO -- put your script code here
```

When you create a new global script, ClearQuest creates a placeholder for the hook but does not create any code. You must supply all of the code for your function or subroutine, including the Function or Sub declaration and parameter list. The names you give to your global scripts must be unique within the schema.

- 6 When you are finished editing the global script, select **Hooks > Compile** to make sure there are no syntactical errors. When you compile, ClearQuest automatically saves your code.
- 7 When you are finished, close the Script Editor window.

Editing a global script

To edit a global script:

- 1 In the Workspace, expand the **Global Scripts** folder, then expand the desired language folder, **Basic** or **Perl**.
- 2 Double-click the script you want to edit.
- 3 Edit the global script as desired. If this is a new global script, type your code in place of the line that reads:

```
REM TODO -- put your script code here
```

- 4 When you are finished editing, select **Hooks > Compile** to make sure there are no syntactical errors. When you compile, ClearQuest automatically saves your code.
- 5 Close the Script Editor window.

Deleting a global script

To delete a global script:

- 1 In the Workspace, expand the **Global Scripts** folder, then expand the desired language folder, **Basic** or **Perl**.
- 2 Right-click the script you want to delete and select Delete from the shortcut menu.
- 3 Click **Yes** to confirm that you want to delete the script.

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 202.

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.

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.

For 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.

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.

For 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.

For more information, see the online *ClearQuest API Reference*.

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

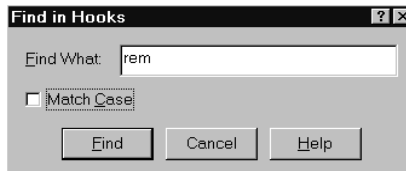
API Call (VBScript/Perl)	Function
QueryDef.BuildFilterOperator QueryFilterNode.BuildFilterOperator \$QueryDef->BuildFilterOperator \$QueryFilterNode->BuildFilterOperator	Builds a filter operator for a query such as "equal to" or "greater than."
QueryFilterNode.BuildFilter \$QueryFilterNode->BuildFilter	Creates support for a complex query.
session.BuildResultSet \$session->BuildResultSet	Creates the ResultSet object necessary to run a query.
ResultSet.Execute \$ResultSet->Execute	Runs the query with the current ResultSet object.
ResultSet.MoveNext \$ResultSet->MoveNext	Moves the cursor to the next record in the data set.
ResultSet.GetColumnValue \$ResultSet->GetColumnValue	Retrieves the value in the column you specify of the current row.
session.GetUserLoginName \$session->GetUserLoginName	Gets the user's login ID.
entity.Revert \$entity->Revert	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 text in hook scripts

You can find specific script text in hooks, searching all hooks in the schema, without having to specify a given hook.

To find text in hook scripts:

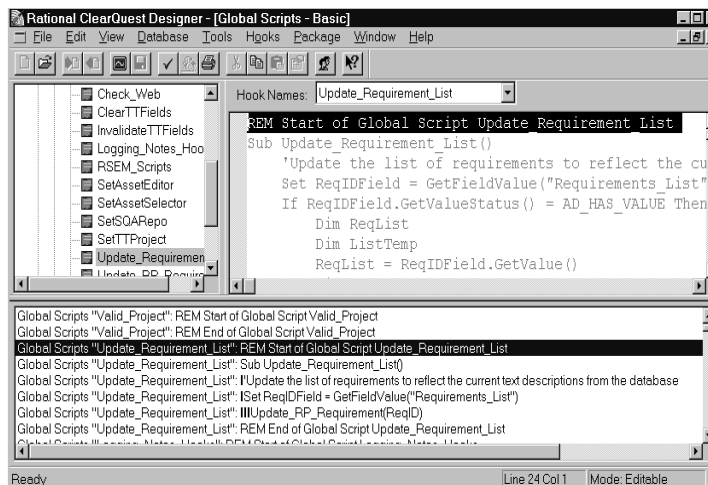
- 1 Open the schema in which you want to search for text.
- 2 Select **Edit > Find in Hooks**.
- 3 In the Find in Hooks dialog box:



- For **Find What**, type the text string to search for.
- Click **Match Case** to search for the text string exactly as you typed it (case-sensitive).
- Click **Find**.

The Validation pane displays the search results:

- 4 In the Validation pane, double-click the specific entry you want to view or edit. This opens an editor window, with the specific script and text displayed.



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** option.
- 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 70).

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, vbCrLf)

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");
    }
}

```

```

} elsif ($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 "")

```

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 *Installation Guide* for Rational ClearQuest.

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 148) 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.

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. See “Enabling dependent fields for ClearQuest Web” on page 219.
- You cannot use message boxes.
- Context-menu hooks are not supported.
- You can use hooks to detect a web session. See “Using hooks to detect a web session” on page 221.

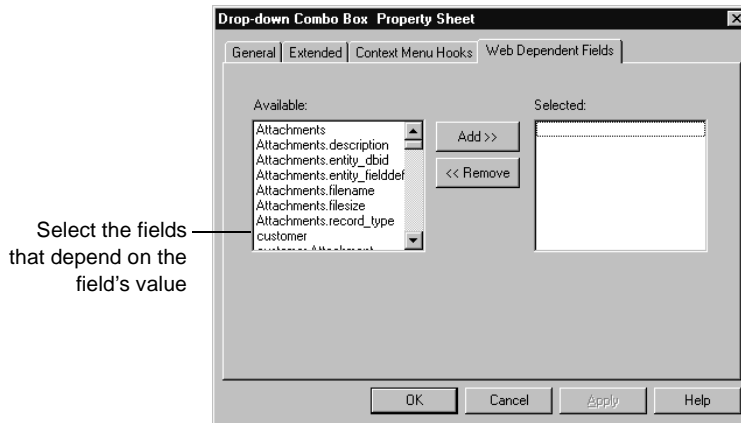
Enabling dependent fields for ClearQuest Web

If you want a form with dependent fields to display in ClearQuest Web, you must do the following:

- 1 When you add the field to the record form, use one of the following form controls for the parent field and its dependent fields. You can mix and match.
 - Drop-down list box
 - Combo box
 - Drop-down combo boxSee “Working with form controls” on page 96.
- 2 After adding the controls to the record form, right-click the control for the parent field and select **Properties** from the shortcut menu.
- 3 In the Web Dependent Fields tab, select the appropriate child fields from the **Available** list and add them to the **Selected** list.

- 4 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, such as a message box, that call other Windows applications 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.

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
}
```


ClearQuest supports the following e-mail capabilities:

- **E-mail notification:** ClearQuest can automatically 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.
- **E-mail submission:** ClearQuest users can submit and modify records by e-mail. For example, users can respond to an automatic e-mail notification with notes that can be appended to a record in the ClearQuest database.
- **Round-trip e-mail:** You can combine e-mail notification with e-mail submission to update records using “round-trip” e-mail.
- **E-mail notification action hook:** See “E-mail notification hook example” on page 232.

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

- Enabling e-mail notification
- Enabling e-mail submission
- Using “round-trip” e-mail
- Creating e-mail actions

Note: For instructions on how to set up e-mail notification for ClearQuest Web, see the *Installation Guide* for Rational ClearQuest.

Enabling e-mail notification

Enabling ClearQuest automatic e-mail notification requires:

- Setting up e-mail rules
- Configuring ClearQuest clients to send e-mail

Setting up e-mail rules

To set up e-mail notification, you create e-mail rules. 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. When you create an e-mail rule, you establish the criteria that triggers e-mail notification. You can send e-mail when:

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

Most ClearQuest predefined schemas include the Email_Rule record type. If the schema you are using does not include the Email_Rule record type, you can add this functionality by adding the Email package (see Appendix A, “ClearQuest schemas and packages”).

Note: To add or modify an e-mail rule, you must have Super User or Schema Designer privilege. For more information, see “ClearQuest user privileges” on page 138.

To set up an e-mail rule:

- 1 In ClearQuest client, select **Actions > New** and select the **Email_Rule** record type.
- 2 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.

The screenshot shows a dialog box titled "Submit Email_Rule". It has a tabbed interface with the following tabs: "To Addressing Info", "CC Addressing Info", "Rule Controls", "Action Controls", and "Display Fields". The "Rule Controls" tab is selected. Inside this tab, there are several fields and controls:

- A "Name:" text input field.
- A "Record Type:" dropdown menu.
- A "Fields to Check for Change:" text area with a small "..." button to its right.
- A "Filter Query:" dropdown menu.
- A checked checkbox labeled "Active Rule".

On the right side of the dialog, there are three buttons: "OK", "Cancel", and "Values" (with a downward arrow).

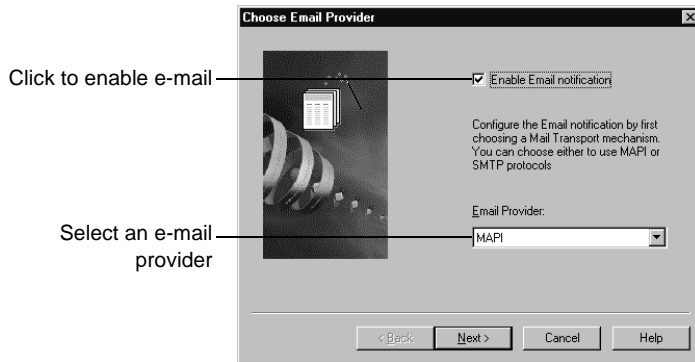
- 3 Submit the record.

Configuring ClearQuest clients to send e-mail

To take advantage of the e-mail rules you set up, each ClearQuest client user must configure their e-mail settings to enable e-mail notification.

To configure e-mail notification:

- 1 In ClearQuest client, select **View > E-mail Options**.



- 2 In the Choose Email Provider dialog box, click **Enable Email notification**.
- 3 Select an **E-mail Provider**, either **MAPI** or **SMTP**, then click **Next** to define the e-mail provider parameters.

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

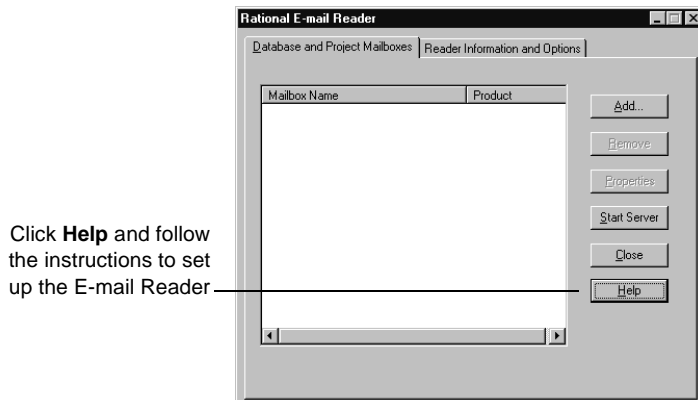
Enabling e-mail submission

To enable ClearQuest 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

By default, the E-mail Reader (`mailreader.exe`) is 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`.



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

- The E-mail Reader requires a dedicated e-mail account to which submissions and modifications can be e-mailed.
- 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 228. As the ClearQuest administrator, you should set up defaults for your ClearQuest users so they will be sure to use the correct 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. For example, you can use the Notes_Log field, which is available in most predefined schemas.

Formatting e-mail for submission

All of your schema rules, including 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.

All e-mail must conform to the format shown below. If it does not conform to this 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 value for a field whose
              data type is a multi-line text field. It requires curly braces.
              }
```

Note: The right curly brace (}) must be on a separate line from the multi-line text it encloses.

Guidelines for using the e-mail format

Use the following guidelines when sending e-mail:

- The <record type> is always required and must be specified first on the Subject line.
- Specify the <action> after the record type on the Subject line. The <action> is the name of the action to be performed on the record. If you have defined default actions in your schema, you can omit the <action> (see the E-mail format example below). Users can override the defaults by specifying a different action.

Note: If you do not provide an <action>, and your schema does not have default actions, the e-mail submission will be rejected.

- List the <record ID> after the <action> on the Subject line. The <record ID> is not necessary when submitting new records.

- Use curly braces for a text field with multiple lines. The right curly brace (}) must be on a separate line from the multiline text it encloses. For example:


```
{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.
- 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.
- E-mail submission does not support adding attachments.
- 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 by e-mail.

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

E-mail format example

In this example, the default action is Modify and the default field is the Notes_Log field. 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.
}

Using “round-trip” e-mail

Your team can use e-mail notification along with e-mail submission to update records. This 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 how you can use round-trip e-mail:

- 1** A user submits a record by e-mail, using the required format. (See “Formatting e-mail for submission” on page 228.) ClearQuest assigns the new defect ID number PROD0000137.
- 2** An e-mail rule that you established automatically sends e-mail that notifies the lead engineer of this defect. (See “Setting up e-mail rules” on page 224.)
- 3** The lead engineer receives 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. (See “Guidelines for using the e-mail format” on page 228)
- 4** The E-mail Reader processes the e-mail message and submits the modified record to the ClearQuest database.

Creating e-mail actions

To have ClearQuest automatically send an e-mail message when a user completes a specific action, you can modify the notification hook of the desired action. In your hook code, you use the ClearQuest API to create the message and send it.

Note: Predefined schemas include the `Send_Email_Notif` base action in all enabled records. This action calls `RSEM_ProcessEmailRules`, and the action type is used for *all* actions.

To create an e-mail action:

- 1 Expand **Record Types** or **Record Types – Stateless** until you see the **Actions** folder.
- 2 Double-click **Actions** to view the actions.
- 3 Click the **Notification** column of the action you want to modify.
- 4 Click the icon to display a list of available hooks.
- 5 Select **SCRIPTS > BASIC** or **SCRIPTS > PERL**.
- 6 If Instant Editing Mode is enabled, ClearQuest opens the script editor automatically. If Instant Editing Mode is disabled, double click the cell to display the script editor.
Note: VBScript and Perl each have their own script editor. ClearQuest Designer indicates the current editor in the title bar of the Designer window. Be sure you are in the correct editor before adding or editing your code.
- 7 Add your hook code, then select **Hooks > Compile** to compile your code and verify that there are no syntax errors.
- 8 Close the script editor window.

You can use the “E-mail notification hook example” on page 232 as a template for generating your own e-mail messages. If your schema includes support for e-mail rules, you can also use those to generate e-mail messages. For more information, see the *Rational ClearQuest API Reference*.

See also:

- “Working with action hooks” on page 187
- “Adding a record script to a record type” on page 197

E-mail notification hook example

The following hook example creates an e-mail message, including some sample header and body text, and sends it to the desired recipients. You should use this example only in conjunction with the notification hook of an action.

Note: These hook examples provide a general idea of how you might add hooks to your schema. For readability, the examples do not include error checking. You should check the return value of the Validate API to ensure there is no error before you commit the record to the database.

VB Script

```
Sub defect_Notification(actionname, actiontype)
    ' actionname As String
    ' actiontype As Long
    ' action = reassign

    Dim mailmsg
    Dim subj
    Dim body
    Dim NL
    Dim session
    Dim submitter
    Dim assign_to
    Dim assignee

    NL = vbCrLf    'newline sequence for imbedding in messages
    ' Set Global to check at end to determine if any valid e-mail
    ' addresses were found.
    sendmsg = False

    ' Create an instance of the mail message class
    Set mailmsg = CreateObject("PAINET.MAILMSG")

    ' Get the current session
    Set session = GetSession

    ' Override the default "from" address (which is the
    ' submitter's id)
    ' and replace it with the administrator's address.
    mailmsg.SetFrom "root@company.com"

    ' Send the message to the submitter (who should always exist)
    ' if that user is active and has a valid e-mail address.
    Set submitter = session.GetEntity("users", _
```

```

GetFieldValue("submitter_id").GetValue)
If submitter.GetFieldValue("is_active").GetValue = 1 _
    And submitter.GetFieldValue("email").GetValue <> "" Then
    mailmsg.AddTo _
        submitter.GetFieldValue("email").GetValue
    sendmsg = TRUE
End If

' If there is an assignee, and that person is not the person
' who submitted the bug, include that person as one of the
' message recipients.
assign_to = GetFieldValue("assign_to").GetValue
If assign_to <> "" Then
Set assignee = session.GetEntity("users",assign_to)
If assignee.GetFieldValue("is_active").GetValue = 1 _
    And assignee.GetFieldValue("email").GetValue <> "" _
    And assignee.GetFieldValue("email").GetValue <> _
        submitter.GetFieldValue("email").GetValue Then
    mailmsg.AddTo assignee.GetFieldValue("email").GetValue
    sendmsg = True
End If
End If

' Send copies of message to Mary and Bob, who are managers.
If InStr(":Submitted:postponed:resolved:",
GetFieldValue("state").GetValue) > 0 Then
mailmsg.AddCc "bob"
mailmsg.AddCc "mary"
End If

' Set the subject of the email message.
subj = GetFieldValue("id").GetValue & _
    ": " & GetFieldValue("state").GetValue & _
    ": " & GetFieldValue("component").GetValue & _
    ": " & GetFieldValue("headline").GetValue
mailmsg.SetSubject subj

' Create the body text of the message. The body text
' contains information from fields of the given record.
body = "id: " & GetFieldValue("id").GetValue & NL & _
    "headline: " & GetFieldValue("headline").GetValue & NL & _
    "state: " & GetFieldValue("state").GetValue & NL & _
    "project: " & GetFieldValue("project").GetValue & NL & _
    "component: " & GetFieldValue("component").GetValue & _
    NL & _
    "assign_to: " & GetFieldValue("assign_to").GetValue & _
    NL & NL & _

```

```

        "submit_date: " & GetFieldValue("submit_date").GetValue &_
        NL & _
        "submitter_id: " & GetFieldValue("submitter_id").GetValue_
        & NL & NL & _
        "impacts_doc: " & GetFieldValue("impacts_doc").GetValue _
        & NL & _
        "problem_description: " & _
        GetFieldValue("problem_description").GetValue & NL & _
        NL & _
        "analysis: " & GetFieldValue("analysis").GetValue & NL & _
        NL

    ' For certain conditions, add extra information to the
    ' message.
    If GetFieldValue("state").GetValue = "resolved" Then
    body = body & _
        "resolution_type: " & _
        GetFieldValue("resolution_type").GetValue & NL & _
        "resolution_date: " & _
        GetFieldValue("resolution_date").GetValue & NL & _
        "resolution_description: " & _
        GetFieldValue("resolution_description").GetValue & NL

    ' Add some extra fields for verified state
    ElseIf GetFieldValue("state").GetValue="verified" Then
    body = body & _"verification_description: " & _
        GetFieldValue("verification_description").GetValue & NL
    End If

    ' Munge mail message to replace LF & "." CR which causes SMTP
    ' to truncate the message
    body = Replace(body, vbNl & "." & vbCr, vbNl & ".." & vbCr)
    mailmsg.SetBody body

    If sendmsg Then
        If mailmsg.Deliver <> 1 Then
            ' Do any necessary failure processing (logging or user
            ' notification). Note that interactive dialogs (i.e.
            ' MsgBox)should not be used since these scripts run in a
            ' service/background context at times(which causes a hang)
            End If
        End If
    End Sub
End Sub

```

Perl

You can write your own hook code for e-mail notification in a manner similar to the above VBScript example. If you choose to do so, you can choose a Perl third party Email package to send out e-mail notification.

Importing and exporting with ClearQuest

11

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

Topics covered include:

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

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 the *Administrator's Guide* for Rational ClearQuest.
- Create an import schema to support the data you want to import. See “Creating an import schema” on page 238.
- 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 240.
- 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 246.

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 240.

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 state names and action names must match the values used in the input file you are importing.
- 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 performing updates to existing records, your import schema must have an action of the Modify type.

- If you will be importing duplicates, your schema must have an action of the Duplicate type. It must also include a DUPLICATE state.
- The schema must contain a record form that users can use to view and modify the imported data.
- If importing duplicates, your form must have controls that support viewing and modifying duplicate records. Use the Duplicate Base control to view the ID of the parent record. Use the Duplicate Dependents control to view the ID(s) of the duplicate records. For an example, look at the Duplicates tab of the Defect record type in any ClearQuest predefined schema.
- 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 243.

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.

For more information, see the following chapters in the *Administrator’s Guide* for Rational ClearQuest: Chapter 3, “Working with ClearQuest schemas” and Chapter 4, “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, attachments, and when making updates to existing records. It is also useful for finding records based on the original record ID.

The original record ID is also used when re-creating parent-child links. In many cases, such as duplicates or parent-child linking, ClearQuest uses the original ID to maintain data integrity after importing new records.

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 is already associated with the schema you have 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.

For more information, see “Applying schema changes to a user database” on page 41.

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 the 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 a separate import file for each record type.

Note: If a record type includes history and attachments, you'll need to create at least three import files, one for records, one for history, and one for 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 values can contain carriage returns. For example:

```
"To reproduce the error, follow these steps:<CR>1. Launch Explorer.<CR>2. Choose File > Open."
```

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:

Data Type	Description	Behavior
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: 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	ClearQuest validates the 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

Each history import file should contain history information for only one record type. If you are importing multiple record types, you will need to have history import files for each record type that uses history information.

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

Each attachment import file should contain attachment information for only one record type. If you are importing multiple record types, you will need to have history import files for each record type that uses attachments.

Keep in mind the following when formatting an 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.
- Each attachment needs to be in its own separate file.

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.

Determining what records to import

The ClearQuest Import Tool allows you to import one record type at a time. You can import history and attachment information for that record type at the same time. If you have duplicate records, you must import them separately. See “Importing duplicate records” on page 250.

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.

Using the ClearQuest Import Tool

You use the ClearQuest Import Tool to import records into ClearQuest.

To begin, 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 ClearQuest Import Tool.

- 1** In the **ClearQuest Import Tool - Step 1 of 5** dialog box:
 - a** Under **Step 1**, select the name of the record type to which you will be importing records.
 - b** Under **Step 2**, select the type of data you will be importing. Select all that apply. You can import history and attachment information for the selected record type, at the same time you import the records of that type.
 - c** Under **Step 3**, if applicable, indicate whether or not you are updating existing records. If this is your first import, this step is not necessary.
Note: You must select **Yes**, if you are importing history information, attachments or duplicate information.
 - d** Click **Next**.
- 2** In the **ClearQuest Import Tool - Step 2 of 5** dialog box:

a Under Record Data,

- Type the path to the import file that contains the data to be imported. You can also browse to the location of the file.
- Type the location and name to use for the discarded data log. You can also browse to the location where you want to store the file. This file must not already exist.
- Select the field delimiter used in the import file.

Note: If a record does not convert successfully, it is saved to the discarded data file you specify. You can use this file to fix problems and to reimport the record. See “Importing records from the discarded data log” on page 251.

b Under History Data, if you are importing history information, do the following:

- Type the path to the import file that contains the data to be imported. You can also browse to the location of the file.
- Type the location and name to use for the discarded data log. You can also browse to the location where you want to store the file. This file must not already exist. If during the import, any data or record cannot be imported, it is stored in the discarded data log.
- Select the field delimiter used in the import file.

Note: If a record does not convert successfully, it is saved to the discarded data file you specify. You can use this file to fix problems and to reimport the record. See “Importing records from the discarded data log” on page 251.

c Under Attachment Data, if you are importing attachment information, do the following:

- Type the path to the import file that contains the data to be imported. You can also browse to the location of the file.
- Type the location and name to use for the discarded data log. You can also browse to the location where you want to store the file. This file must not already exist. If during the import, any data or record cannot be imported, it is stored in the discarded data log.

- Select the field delimiter used in the import file.

Note: If a record does not convert successfully, it is saved to the discarded data file you specify. You can use this file to fix problems and to reimport the record. See “Importing records from the discarded data log” on page 251.

d Click **Next**.

3 In the **ClearQuest Import Tool - Step 3 of 5** dialog box:

a Under **Importing State Values**, select 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.

b Under **Importing Duplicates**,

- Mark whether your import file contains information about duplicate records.
- If your import file contains information about duplicates, select the exported field name that contains the duplicate information. For more information on importing duplicates, see “Importing duplicate records” on page 250

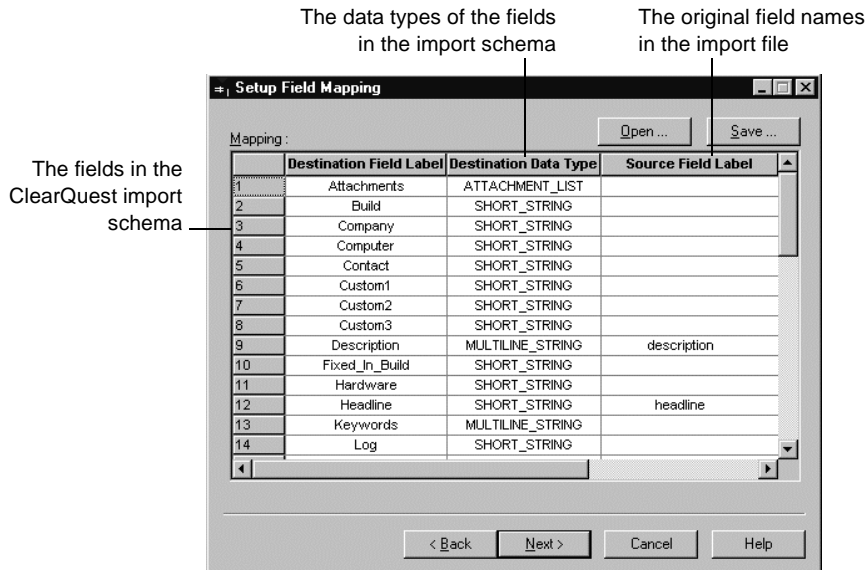
c Under **Importing Record Updates, Duplicates, History, or Attachments**,

- When importing history, attachments, and duplicate records, ClearQuest must track the past unique identifier (typically the old ID number of the imported record) of the imported record. ClearQuest must know the record to which the attachment, history information or duplicate record belongs.

Select the field in the import schema that will contain the original record ID (or other unique identifier) of the exported data.

d Click **Next**.

- 4 In the **ClearQuest Import Tool - Step 4 of 5** dialog box, 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.
- Be sure to map the import file's original ID to the Destination field that will store it.

To save the mapping you create:

- a Click **Save**.
- b Browse to a location where you want to save the mapping profile, and type a name. The file must have a *.txt extension.
- c Click **Save**.

To re-use an existing mapping:

- a Click **Load**.
- b Browse to where the mapping profile exists and select it.

- c Click **OK**.
- 5 In the **ClearQuest Import Tool - Step 5 of 5** dialog box,
- a Under **Summary**, review the import summary to verify that the settings you chose are correct.
 - b Specify how many errors can occur before the import process is terminated
 - c Click **Import**.

Importing duplicate records

You can import duplicate records when importing all of your records. To import duplicate records, consider 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 246.
- 2 In Step 3, click **Import Duplicates** and enter the name of the field that contains the ID of the parent record (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 discarded data log.
- 4 When the import is completed, use the discarded data log (after fixing the problems) 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 discarded data log

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

- `discarded data log file`: ClearQuest saves the unimported records to the error file whose name and location you defined in Step 1 above.
- `errlog.txt`: If ClearQuest encounters problems with the discarded data log, it saves error messages to a text file in the same directory as the discarded data log.

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 241.
- 4 Perform the import process again, this time specifying the error file as the import file.

Updating 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.

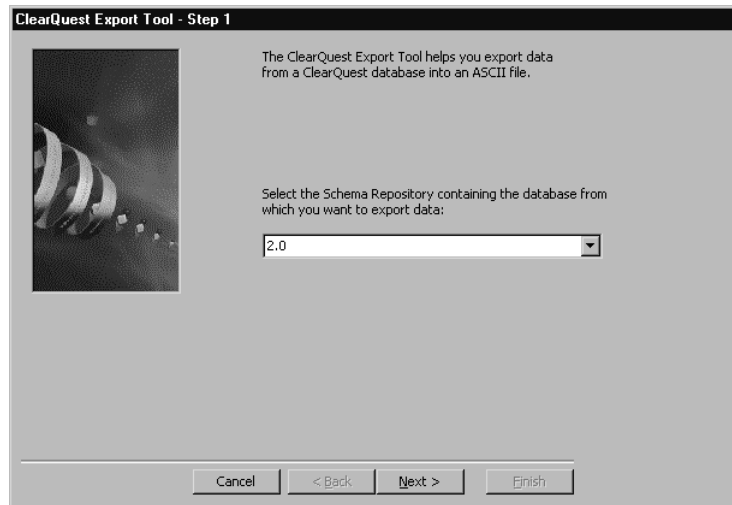
To upgrade existing records, follow the procedure for “Performing the data import” on page 246. In Step 1, select **Update existing records**. All other steps remain the same.

Using the ClearQuest Export Tool

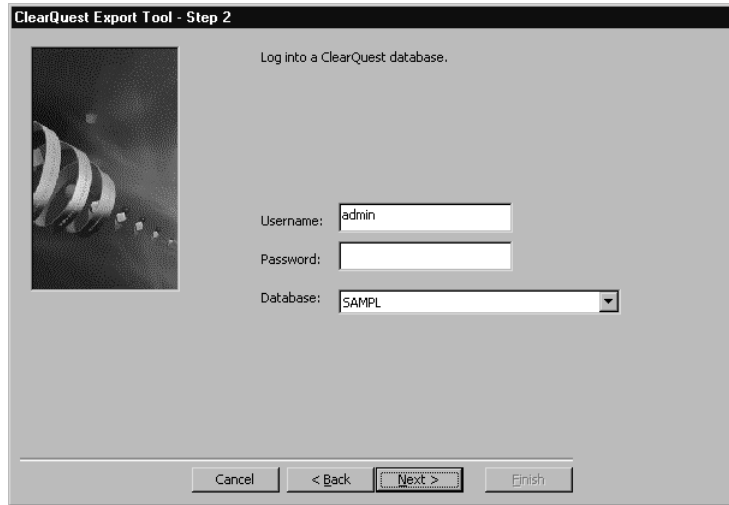
The ClearQuest Export Tool exports data from a ClearQuest database into an ASCII format optimized for the ClearQuest Import Tool.

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.



The screenshot shows the 'ClearQuest Export Tool - Step 2' dialog box. It features a decorative image on the left and the instruction 'Log into a ClearQuest database.' on the right. Below the instruction are three input fields: 'Username:' with the text 'admin', 'Password:', and 'Database:' with a dropdown menu showing 'SAMPL'. At the bottom, there are four buttons: 'Cancel', '< Back', 'Next >', and 'Finish'.

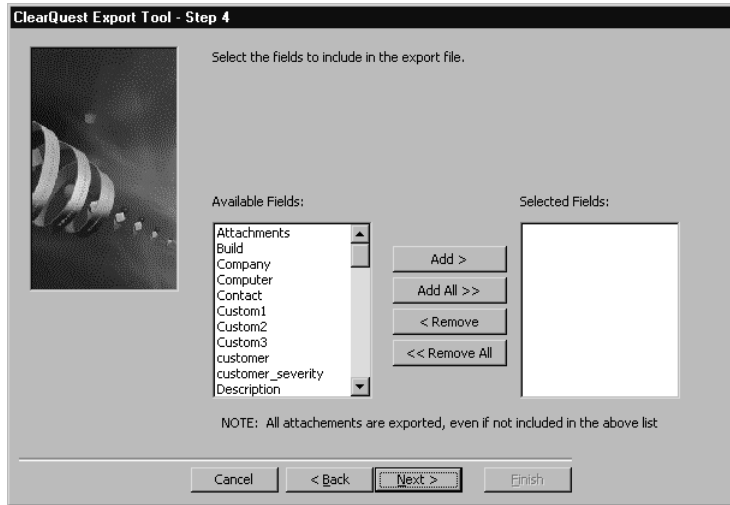
4 Select a record type and indicate which records you want to export. You can export all records or select a ClearQuest query to export specific records.



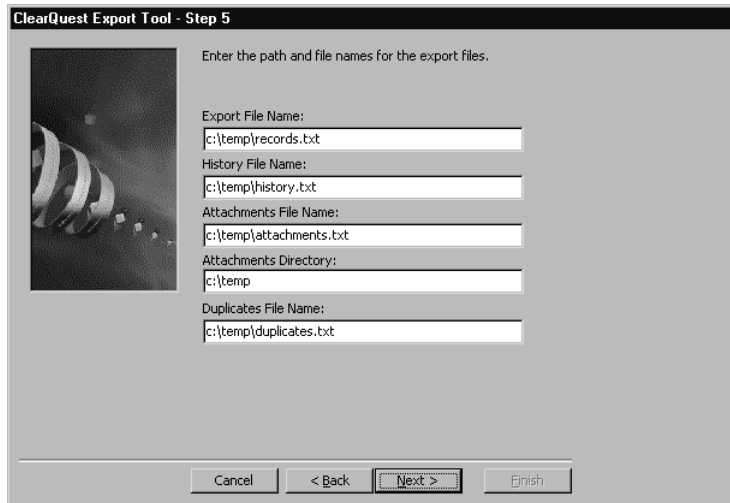
The screenshot shows the 'ClearQuest Export Tool - Step 3' dialog box. It features a decorative image on the left and the instruction 'Select a record type and indicate which records you want to export.' on the right. Below the instruction are three input fields: 'Select a record type' with a dropdown menu showing 'Defect', a radio button for 'Export all records' (which is selected), and a radio button for 'Export records based on an existing query' with an empty dropdown menu below it. At the bottom, there are four buttons: 'Cancel', '< Back', 'Next >', and '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.



ClearQuest Export Tool - Finished!

Enter the export file parameters, path and file name for the error log, then click Finish to begin exporting data.

Enter number of records to include per export file. Enter 0 to include all.

Select field delimiter:

▼

Error log file name:

Cancel < Back Next > Finish

- Specify **0** to include all records in one file or enter the number of records you want per file.
 - 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.

ClearQuest schemas and packages



ClearQuest includes several predefined schemas: There is a schema for each Rational Windows suite, and a “Blank” schema, containing only required record types, which 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.

Warning: Be sure to plan carefully before adding packages. Once you add a package to a schema, you cannot remove the package. You must delete all schema versions in which the package exists. You can delete schema versions only if you have not applied them to a user database.

For instructions on how to add packages to a schema, see “Working with packages” on page 46. For information on how to customize a schema, see Chapter 4, “Customizing a schema.”

ClearQuest predefined schemas

ClearQuest includes the following predefined schemas:

Schema	Description	Packages included
Blank	Contains only record types. Use Blank to create a schema from scratch.	None
Common	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, TeamTest
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, TeamTest
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
Enterprise	For use with Rational EnterpriseStudio. Contains fields and hooks that work with most Rational Suite products. This schema is UCM-enabled.	AMStateTypes, Attachments, BaseCMActivity, Email, EnhancementRequest, History, Notes, PQC, Resolution, Repository, RequisitePro, TeamTest, UCMPolicyScripts, UnifiedChangeManagement, TeamTest
UnifiedChange Management (UCM)	Provides support for integration with UCM and sets up ClearQuest to use the predefined ClearCase policies.	AMStateTypes, Attachments, BaseCMActivity, Email, History, Notes, Resolution, UCMPolicyScripts, UnifiedChangeManagement

ClearQuest packages

ClearQuest includes the following packages. Some packages are read-only; their functionality cannot be changed. This section describes the latest version of each package. Earlier versions may create different record types and fields.

Package	Description	Record type(s) added/modified	Field(s)
AMBaseActivity	Provides additional support for Rational ClearQuest Project Tracker.	Adds the Main tab to the forms of the enabled record type.	Fields modified in or added to the enabled record type: <ul style="list-style-type: none"> • Headline • Owner • Description
AMStateTypes	Provides additional support for Rational Unified Change Management and its state types. Requires you to map schema states to the following state types: Waiting, Ready, Active, and Complete.	Does not add any record types.	Fields added to the enabled record type: <ul style="list-style-type: none"> • am_statetype
AMWorkActivitySchedule	Provides scheduling attributes needed to integrate Rational ClearQuest and Microsoft Project 2000 using Rational ClearQuest Project Tracker. With the AMWorkActivitySchedule record type family, you can query records being created and updated with Rational ClearQuest Project Tracker.	Defines and adds the AMSchedule record type family to the enabled schema. Record types being enabled with this package will be added to this record type family. Adds the Schedule tab to the enabled record type.	Fields added to the enabled AMSchedule record type: <ul style="list-style-type: none"> • am_planned_start_date • am_planned_end_date • am_planned_work • am_planned_rem_work • am_planned_duration • am_planned_rem_duration • am_actual_start_date • am_actual_end_date • am_actual_work
Attachments (read-only)	Allows you to add and remove attachments related to a record.	Adds an Attachments tab to the enabled record type.	Fields added to enabled record type: <ul style="list-style-type: none"> • Attachments

Package	Description	Record type(s) added/modified	Field(s)
BaseCMActivity	<p>Provides support for the BaseCMActivity 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.</p> <p>For more information, see <i>Managing Software Projects with ClearCase</i>.</p>	<p>Adds the BaseCMActivity record type.</p>	<p>Fields included in BaseCMActivity record type:</p> <ul style="list-style-type: none"> • Owner • Description • Headline
ClearCase (read-only)	<p>Provides basic support for the Rational Base ClearCase integration. This package does not set up ClearQuest to use predefined ClearCase policies; they must be set up by the ClearCase administrator.</p>	<p>Adds the cc_change_set and cc_vob_object record types.</p> <p>Adds the ClearCase tab to the enabled record type.</p>	<p>Fields included in cc_change_set</p> <ul style="list-style-type: none"> • record type: • objects <p>Fields included in cc_vob_object</p> <ul style="list-style-type: none"> • record type: • name • object_oid • vob_family_uuid <p>Fields added to enabled record type:</p> <ul style="list-style-type: none"> • cc_change_set
ContentStudio	<p>Provides support for integration with Rational Suite ContentStudio.</p>	<p>Adds the ContentChangeRequest record type.</p>	<p>Fields included in ContentChangeRequest</p> <ul style="list-style-type: none"> • record type: • Description • Headline • Note_Entry • Notes_Log • Owner • vgnAssignee • vgnCMS • vgnDueDate • vgnManagementID • vgnProjectPath • vgnTaskName

Package	Description	Record type(s) added/modified	Field(s)
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 select.</p>	<p>Fields included in the Customer record type:</p> <ul style="list-style-type: none"> • Attachment • CallTrackingID • Company • Description • Email • Fax • Name • Phone <p>Fields added to enabled record type:</p> <ul style="list-style-type: none"> • Customer • Customer_Severity
Email (read-only)	Supports automatic e-mail notification when records are modified.	<p>Creates an Email_Rule stateless record type.</p> <p>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.</p>	<p>Fields included in Email_rule record type:</p> <ul style="list-style-type: none"> • Actions • Action_Types • CC_Actioner • CC_Addr_fields • CC_Additional • CC_Groups • CC_Users • Change_Fields • Display_Fields • Entity_Def • From_Addr • Include_Defect • Is_Active_Rule • Filter_Query • Name • Operator_Value • Show_Previous • Source_States • Subject_Fields • Target_States • To_Additional • To_Addr_Fields • To_Groups • To_Users

Package	Description	Record type(s) added/modified	Field(s)
Enhancement Request	Supports an additional record type for product enhancement requests.	Adds the EnhancementRequest record type.	Fields included in EnhancementRequest record type: <ul style="list-style-type: none"> • 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
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.	No fields added.
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, which deletes the Note_Entry value.	Fields added to enabled record type: <ul style="list-style-type: none"> • 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.	Fields added to enabled record type: <ul style="list-style-type: none"> • PQC_DiagnosticTool • PQC_Executable • PQC_TestCommand • PQC_TestTool • PQC_Stack • PQC_StackID
Project	Allows you to track records according to project. (Note: No relation to the UCM package "Project" concept.)	Creates a Project stateless record type.	Fields included in Project record type: <ul style="list-style-type: none"> • Name • Description Fields added to enabled record type: <ul style="list-style-type: none"> • Project

Package	Description	Record type(s) added/modified	Field(s)
Repository (read-only)	Provides support needed for both Rational RequisitePro, Rational Administrator, and Rational TeamTest.	Creates an RASProject stateless record type.	<p>Fields included in RASProject record type:</p> <ul style="list-style-type: none"> Name TT_Repo (Refers to the TeamTest Repository) RA_Project_Path <p>Fields added to enabled record type:</p> <ul style="list-style-type: none"> RASProject
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>Fields included in Requirement record type:</p> <ul style="list-style-type: none"> Name Project_Name Req_GUID Req_ID Requirement Tag <p>Fields included in RequirementMap record type:</p> <ul style="list-style-type: none"> CQBackReqListAttName CQDatabase CQDatabasePath CQDialogTitle CQEntityDefName CQHelpContextID CQModifyAction CQReqListAttName CQRepoProjectAttName HelpFileName RPAtrrGUID RPHelpContextID RPProjectName RPProjectPath RPreqTypeGUID <p>Fields added to enabled record type:</p> <ul style="list-style-type: none"> Requirements_List
Resolution	<p>Adds support for tracking 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>Fields added to enabled record type:</p> <ul style="list-style-type: none"> Resolution Resolution_StateType (read-only)

Package	Description	Record type(s) added/modified	Field(s)
TeamTest (read-only)	Provides support for integration with Rational TeamTest.	<p>Adds Test Data and Environment tabs to the record type you specify.</p> <p>Also adds a TestInput stateless record type.</p>	<p>Fields added to enabled record type:</p> <ul style="list-style-type: none"> • 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_Case • Test_Case_UID • Test_Script • Test_Script_ID • Test_Source_UID • Test_Input_List • Verification_Point <p>Fields added to the TestInput record type:</p> <ul style="list-style-type: none"> • Test_Input_Name • Test_Input_ID • Source_UID
UCMPolicyScripts	Provides support for the UnifiedChangeManagement (UCM) package by adding three global scripts.	Does not add any record types.	

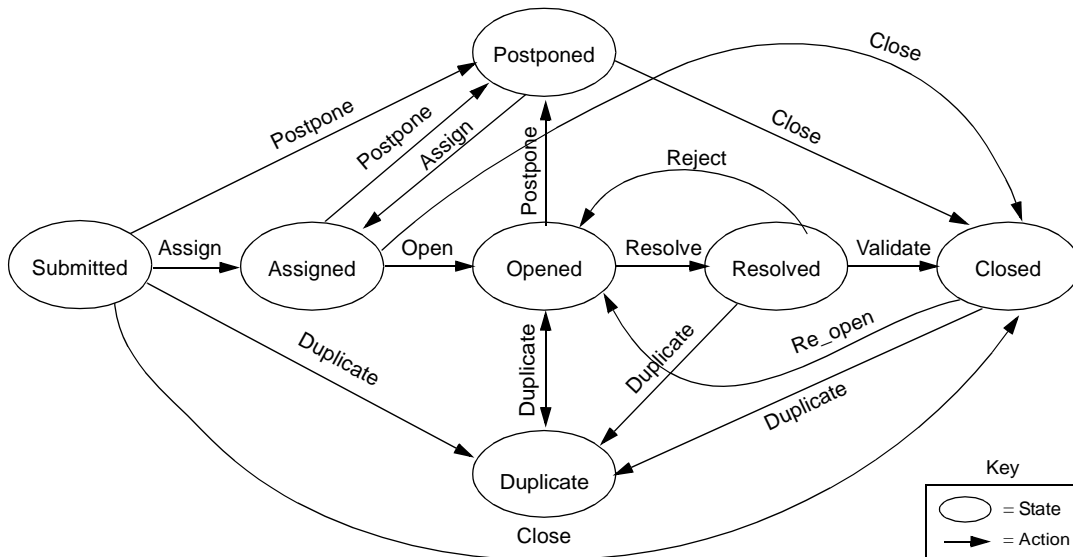
Package	Description	Record type(s) added/modified	Field(s)
UnifiedChange Management (UCM) (read-only)	<p>Provides supports for the UCM process by enabling integration with Rational ClearCase 4.0 and above; links a ClearCase Project VOB with a ClearQuest user database. Requires the UCMPolicyScripts package. 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 the UCMUtilityActivity record type.</p> <p>Adds UCM_Project stateless record type.</p> <p>Adds UCM queries to the client workspace in the Public Folders.</p> <p>Adds the ucm_base_synchronize action to the enabled record type.</p>	<p>Fields included in UCMUtilityActivity record type:</p> <ul style="list-style-type: none"> am_statetype Description Owner Headline ucm_vob_object ucm_stream_object ucm_stream ucm_view ucm_project <p>Fields included in UCM_Project record type:</p> <ul style="list-style-type: none"> name ucm_vob_object ucm_chk_before_deliver ucm_chk_before_work_on ucm_chk_mstr_before_dlv ucm_cq_act_after_deliver <p>Fields added to enabled record type:</p> <ul style="list-style-type: none"> am_statetype ucm_vob_object ucm_stream_object ucm_stream ucm_view ucm_project
Visual SourceSafe (read-only)	<p>Provides support for integration with Microsoft Visual SourceSafe.</p>	<p>Adds SSOBJect and SCSnapObject stateless record types.</p> <p>Adds the SourceSafe tab to the form of the enabled record type.</p>	<p>Fields added to enabled record type:</p> <ul style="list-style-type: none"> VSSChangeSet <p>Fields added in SSOBJect record type:</p> <ul style="list-style-type: none"> CQDefects VSSCheckOutState VSSFileName VSSSpec VSSUser VSSVersion <p>Fields added in SCSnapObject record type:</p> <ul style="list-style-type: none"> CreatedBy CreatedOn Label SnapElements

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.



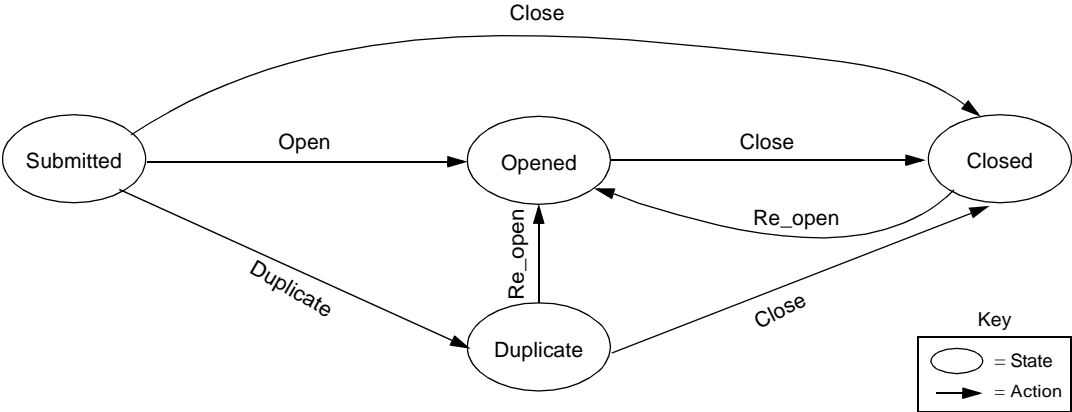
All Duplicate actions can be unduplicated by using the Unduplicate action.

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

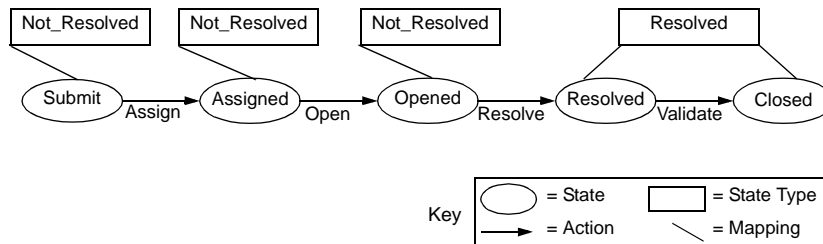
A state type is a label that defines a state's role in your state model. Some ClearQuest schemas have packages that require that each state in your state model be assigned or mapped to a specific state type. For example, the UnifiedChangeManagement schema and package use state types to invoke certain ClearCase actions. 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 78.

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.

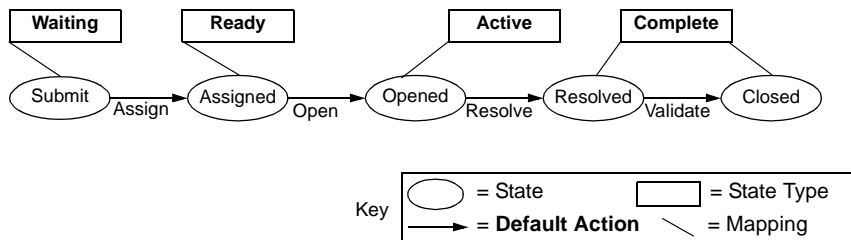
State type	Description
Not_Resolved	The record is not resolved.
Resolved	The record is resolved.



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 has been assigned. It appears in the assigned user's To Do List query and is ready to be worked on. It may or may not have its ucm_project field set.	Assigned
Active	User has started working on the record, which is now associated with a ClearCase project and can contain ClearCase element information.	Opened
Complete	The record has been either: 1) Worked on and completed, which is associated with a ClearCase project and can contain ClearCase element information, or not worked on and abandoned. OR 2) Postponed or closed, which may or may not be associated with a ClearCase project.	Resolved Closed Duplicate



See Appendix B, “Adding ClearQuest integrations” for more information.

Adding ClearQuest integrations

B

You can integrate ClearQuest with software configuration management tools to create a complete change management system. This appendix describes the integrations available with ClearQuest. The topics covered include:

- Overview of ClearQuest integrations
- Viewing the packages in your schema
- Adding independent integrations
- Adding dependent integrations

Note: The instructions described in this appendix assume that you know how to work with ClearQuest schemas. See Chapter 3, “Working with ClearQuest schemas.”

Overview of ClearQuest integrations

To integrate ClearQuest with other software, you add ClearQuest packages to existing schemas. Some ClearQuest integrations are *independent integrations*, which only require adding the appropriate package. Other integrations are *dependent integrations*, which not only require you to add one or more packages in a specific order, but may also require additional configurations to ClearQuest.

Note: The instructions in this appendix assume that you are adding a new integration and that the necessary packages do not already exist in your schema. If you need to upgrade an integration or package, see the *Installation Guide* for Rational ClearQuest.

Warning: Be sure to plan carefully before adding packages. Once you add a package to a schema, you cannot remove the package. You must delete all schema versions in which the package exists. You can delete schema versions only if you have not applied them to a user database.

Independent integrations

The integrations listed below are independent integrations. You can use the same installation process for all independent integrations. See “Adding independent integrations” on page 276.

- Rational Base ClearCase/ClearQuest
Associates one or more ClearQuest change requests with one or more ClearCase versions.
- Rational Suite ContentStudio/ClearQuest
Allows you to submit and track content changes with ClearQuest.
- Rational PureCoverage/ClearQuest
Allows you to submit code coverage data to a ClearQuest database and track it.
- Rational Purify/ClearQuest
Allows you to submit data to a ClearQuest database and track it.
- Rational Quantify/ClearQuest
Allows you to submit performance data to a ClearQuest database and track it.

- **Your e-mail system/ClearQuest**
Enables ClearQuest to communicate with users through their e-mail systems.

An e-mail system integration involves configuring Rational E-mail Reader and adding the e-mail notification package. See Chapter 10, “Administering ClearQuest E-mail.”

Dependent integrations

The integrations listed below are dependent integrations. For more information about dependent integrations, see “Adding independent integrations” on page 276.

- **Rational Administrator/ClearQuest**
Associates Rational projects with ClearQuest databases. See “Adding a Rational Administrator integration” on page 279.
- **Rational ClearQuest Project Tracker/ClearQuest**
Allows you to exchange project data between the two systems. See “Adding a Rational ClearQuest Project Tracker integration” on page 280.
- **Rational RequisitePro/ClearQuest**
Associates RequisitePro requirements with ClearQuest records.

For information on how to add a RequisitePro/ClearQuest integration, see Chapter 4, “Configuring the Integration for ClearQuest and RequisitePro” in the *Administrator’s Guide for Rational Suite*.

- **Rational TeamTest/ClearQuest**
Allows you to submit defects found through TeamTest to ClearQuest databases, and to track them. See “Adding a Rational TeamTest integration” on page 282.
- **Rational Unified Change Management (UCM)/ClearQuest**
Links ClearCase UCM projects and activities to ClearQuest records. See “Adding a Rational UCM integration” on page 284.
- **Microsoft Visual SourceSafe/ClearQuest**
Associates Visual SourceSafe information with ClearQuest records. See “Adding a Microsoft Visual SourceSafe integration” on page 290.

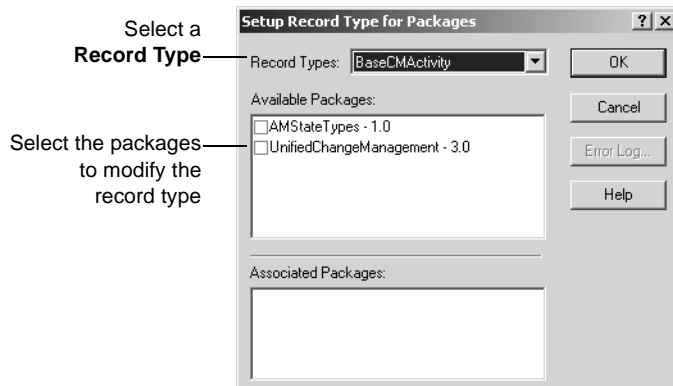
Enabling record types for integrations

Some packages enable existing record types in a schema. At the time you install such packages, you have the option of selecting which record types you want to be enabled by the package. If you add a new record type after adding the package, you can also enable the new record type with the package functionality.

For a list of packages and the record types they modify, see “ClearQuest packages” on page 259.

To enable package functionality for a new record type:

- 1 In ClearQuest Designer, select **File > Open Schema** and select a schema to open.
- 2 With the schema open, select **Package > Setup Record Types for Package**.



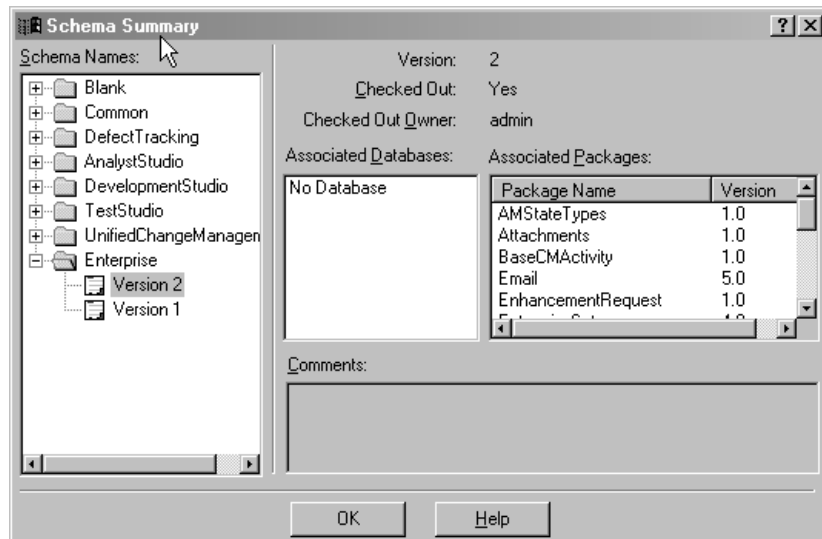
- 3 In the Setup Record Types for Package dialog box, select a record type from the **Record Types** list.
- 4 In the **Available Packages** list, check the package(s) that you want to enable the record type.

Viewing the packages in your schema

Many packages are used by multiple integrations. Before you add any integrations to your schema, you should find out what packages are already in the schema.

To view a list of packages already in your schema:

- 1 In ClearQuest Designer, select **View > Schema Summary**.
- 2 In the Schema Summary dialog box, open the desired schema folder and select the latest version of the schema.



- The **Associated Databases** list shows the databases associated with the schema.
- The **Associated Packages** list displays every package version that is currently installed in your schema.

Testing integrations

After you add packages to a schema, check in the schema, and apply the schema to your user database, you cannot undo the changes. To avoid permanent damage to your production database, it is highly recommended that you add integration packages in a test environment before modifying your production environment.

When performing a test integration, replicate both production environments: the ClearQuest environment, and the environment of the application being integrated with ClearQuest. For example, in a ClearQuest/ClearCase integration, you need:

- A test ClearQuest database
- A test ClearQuest schema
- A test ClearCase VOB

For more information, see “Working with a test database” on page 33.

Adding independent integrations

Independent integrations require you to simply add the appropriate package.

Integration	Package to add	Additional documentation
Rational ClearCase (Base ClearCase)	ClearCase	ClearCase online help. In the table of contents, select ClearCase User Interface > Integrations with other products > Base ClearCase integration with ClearQuest. Note: The ClearCase integration discussed in this section is a Base ClearCase integration; there are no predefined policies. Policies must be set up by the ClearCase administrator. The UCM integration described in “Adding a Rational UCM integration” on page 284, automatically sets up predefined ClearCase policies.
Rational Purify	PQC	Rational Purify online help. Look up <i>ClearQuest</i> .
Rational Quantify	PQC	Rational Quantify online help. Look up <i>ClearQuest</i> .
Rational PureCoverage	PQC	Rational PureCoverage online help. Look up <i>ClearQuest</i> .
Rational Suite ContentStudio	ContentStudio	<i>Installing Rational Suite ContentStudio</i>

You must have Super User or Schema Designer user privileges to add packages.

To find out if a package is already installed in your schema, see “Viewing the packages in your schema” on page 275.

If possible, add all integrations first in a test environment before adding the integration to your production database. See “Testing integrations” on page 276.

Warning: Be sure to plan carefully before adding packages. Once you add a package to a schema, you cannot remove the package. You must delete all schema versions in which the package exists. You can delete schema versions only if you have not applied them to a user database.

To add an independent integration:

- 1** In ClearQuest Designer, make sure the schema you want to add the package to is checked in. To check in a schema, select **File > Check In**.
- 2** Select **Package > Package Wizard** and install the desired package. See “Working with packages” on page 46 for detailed instructions.
- 3** Validate the schema changes. See “Validating schema changes” on page 39 for detailed instructions.
- 4** Select **File > Check In** to save the schema changes. See “Checking in a schema” on page 40 for detailed instructions.
- 5** Apply the schema changes to the user database by selecting **Database > Upgrade Database**. See “Applying schema changes to a user database” on page 41 for detailed instructions.
- 6** Configure the integrated application as needed. See the application’s documentation for additional configuration needs.

Adding dependent integrations

Dependent integrations may require that you add multiple packages in a specific order, and may also require additional configurations. Integration packages must be installed in the specified order.

You must have Super User or Schema Designer user privileges to add packages.

If possible, add the integration to a test database and familiarize yourself with the integration features before adding the integration to your production database. See “Testing integrations” on page 276.

Warning: Be sure to plan carefully before adding packages. Once you add a package to a schema, you cannot remove the package. You must delete all schema versions in which the package exists. You can delete schema versions only if you have not applied them to a user database.

The dependent integrations include:

- Rational Administrator, on page 279
- Rational ClearQuest Project Tracker, on page 280
- Rational RequisitePro, see Chapter 4, “Configuring the Integration for ClearQuest and RequisitePro” in the *Administrator’s Guide for Rational Suite* guide.
- Rational TeamTest, on page 282
- Rational UCM, on page 284
- Microsoft Visual SourceSafe, on page 290

Adding a Rational Administrator integration

Integrating with Rational Administrator associates Rational projects with ClearQuest databases. The Rational Administrator integration requires:

- Adding the Repository package
- Saving the schema changes
- Configuring Rational Administrator

Note: To find out if the Repository package already exists in your schema, see “Viewing the packages in your schema” on page 275. If you already have a Repository package in your schema, and you just want to apply it to a new record type, see “Enabling record types for integrations” on page 274 for instructions.

Adding the Repository package

- 1 In ClearQuest Designer, make sure the schema you want to add the package to is checked in. To check in a schema, select **File > Check In**.
- 2 Select **Package > Package Wizard** and add the latest **Repository** package. See “Working with packages” on page 46 for detailed instructions.

Saving the schema changes

After you add the Repository package:

- 1 Validate the schema changes. See “Validating schema changes” on page 39 for detailed instructions.
- 2 Select **File > Check In** to save the schema changes. See “Checking in a schema” on page 40 for detailed instructions.
- 3 Apply schema changes to the user database by selecting **Database > Upgrade Database**. See “Applying schema changes to a user database” on page 41 for detailed instructions.

Configuring Rational Administrator

Configure the Rational Administrator application as needed. See Rational Administrator online help for additional configuration information.

Adding a Rational ClearQuest Project Tracker integration

Integrating with Rational ClearQuest Project Tracker allows you to exchange project data between Project Tracker and ClearQuest. The Rational ClearQuest Project Tracker integration requires:

- Adding the AMBaseActivity package
- Adding the AMWorkActivitySchedule package
- Saving the schema changes
- Configuring Rational ClearQuest Project Tracker

Warning: To avoid errors, you must install packages in the order listed above.

Note: To find out if the AMBaseActivity and AMWorkActivitySchedule packages are already installed in your schema, see “Viewing the packages in your schema” on page 275. If you already have these packages in your schema, and you just want to apply them to a new record type, see “Enabling record types for integrations” on page 274 for instructions.

Adding the AMBaseActivity package

- 1 In ClearQuest Designer, make sure the schema you want to add the package to is checked in. To check in a schema, select **File > Check In**.
- 2 Select **Package > Package Wizard** and select the latest **AMBaseActivity** package to add. See “Working with packages” on page 46 for detailed instructions.
- 3 Check the schema back into the schema repository by selecting **File > Check In**.

Adding the AMWorkActivitySchedule package

With the schema checked out, select **Package > Package Wizard**, and add the latest AMWorkActivitySchedule package.

Saving the schema changes

After you add the AMWorkActivitySchedule package:

- 1 Validate the schema changes. See “Validating schema changes” on page 39 for detailed instructions.
- 2 Select **File > Check In** to save the schema changes. See “Checking in a schema” on page 40 for detailed instructions.
- 3 Apply schema changes to the user database by selecting **Database > Upgrade Database**. See “Applying schema changes to a user database” on page 41 for detailed instructions.

Configuring Rational ClearQuest Project Tracker

Configure Project Tracker as needed. See the *User’s Guide for Rational ClearQuest Project Tracker* for instructions on linking your project plans with ClearQuest databases and other integrated tasks.

Adding a Rational TeamTest integration

Integrating with Rational TeamTest allows you to submit defects found through TeamTest and to keep track of changes more easily. The Rational TeamTest integration requires:

- Adding the Repository package
- Adding the TeamTest package
- Saving the schema changes
- Configuring Rational TeamTest

Warning: To avoid errors, you must install packages in the order listed above.

Note: To find out if the Repository and TeamTest packages are already installed in your schema, see “Viewing the packages in your schema” on page 275. If you already have these packages in your schema, and you just want to apply them to a new record type, see “Enabling record types for integrations” on page 274 for instructions.

Adding the Repository package

- 1 In ClearQuest Designer, make sure the schema you want to add the package to is checked in. To check in a schema, select **File > Check In**.
- 2 Select **Package > Package Wizard** and add the latest **Repository** package. See “Working with packages” on page 46 for detailed instructions.
- 3 Check the schema back into the schema repository by selecting **File > Check In**.

Adding the TeamTest package

With the schema checked out, select **Package > Package Wizard**, and add the latest TeamTest package.

Saving the schema changes

After you add the TeamTest package:

- 1 Validate the schema changes. See “Validating schema changes” on page 39 for detailed instructions.
- 2 Select **File > Check In** to save the schema changes. See “Checking in a schema” on page 40 for detailed instructions.

- 3 Apply schema changes to the user database by selecting **Database > Upgrade Database**. See “Applying schema changes to a user database” on page 41 for detailed instructions.

Configuring Rational TeamTest

Configure the TeamTest application as needed. See Rational TeamTest online help for additional configuration information.

Adding a Rational UCM integration

The Unified Change Management (UCM) integration links ClearCase UCM projects and activities to ClearQuest records. This is also known as a UCM-ClearCase integration.

The UCM integration requires:

- 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. For instructions on using schemas, see “Working with ClearQuest schemas” on page 31.

You can also add UCM support to an existing schema by adding the correct packages to it. This section describes integrating ClearQuest and UCM by adding packages. These packages must be added in the order described in each step.

Note: Although the UCM integration allows you to work with ClearCase, you must not add the ClearCase package for this integration. The ClearCase package is only used for a Base ClearCase integration, which does not set up predefined ClearCase policies. See “Adding independent integrations” on page 276 for instructions on adding a Base ClearCase integration.

Integrating Rational UCM with packages requires:

- Adding the AMStateTypes package
- Setting the default actions for UCM
- Adding the UCMPolicyScripts package
- Adding the UnifiedChangeManagement package
- Adding the BaseCMAActivity package (optional)
- Saving the schema changes
- Configuring Rational UCM

Warning: To avoid errors, you must install packages in the order listed above.

Note: To find out if the AMStateType, UCMPolicyScripts, UnifiedChangeManagement, and BaseCMAActivity packages are already installed in your schema, see “Viewing the packages in your schema” on

page 275. If you already have these packages in your schema, and you just want to apply them to a new record type, see “Enabling record types for integrations” on page 274 for instructions.

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

Adding the AMStateTypes package

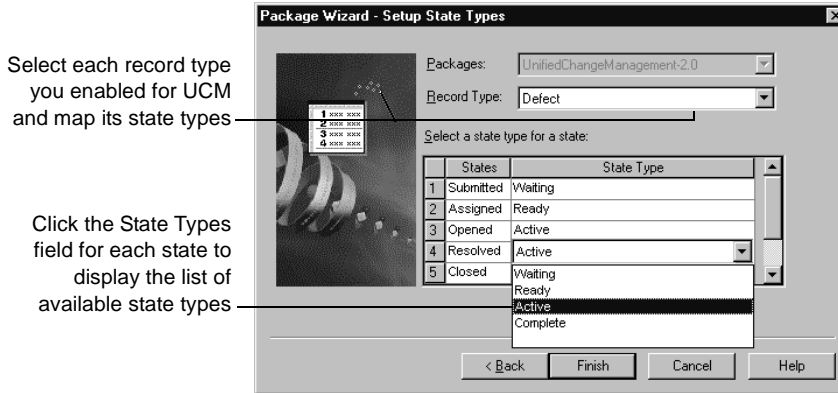
- 1 In ClearQuest Designer, make sure the schema you want to add the package to is checked in. To check in a schema, select **File > Check In**.
- 2 Select **Package > Package Wizard** and add the latest **AMStateType** package. See “Working with packages” on page 46 for detailed instructions.

The AMStateType package requires you to map state types and define default actions, if they have not already been defined.

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



- 4 In the Setup State Types dialog box, map the states in your schema to the UCM state types:
 - Select a record type.
 - For each state in the record type, click in the **State Type** field and select the appropriate UCM state type. For more details regarding mapping state types, see “Mapping state types” on page 78.



- 5 Repeat the state type mapping for each record type you enabled. When you are done, click **Finish**.

ClearQuest automatically validates your schema. The validation window will indicate that you need to set default actions. Continue to “Setting the default actions for UCM,” next.

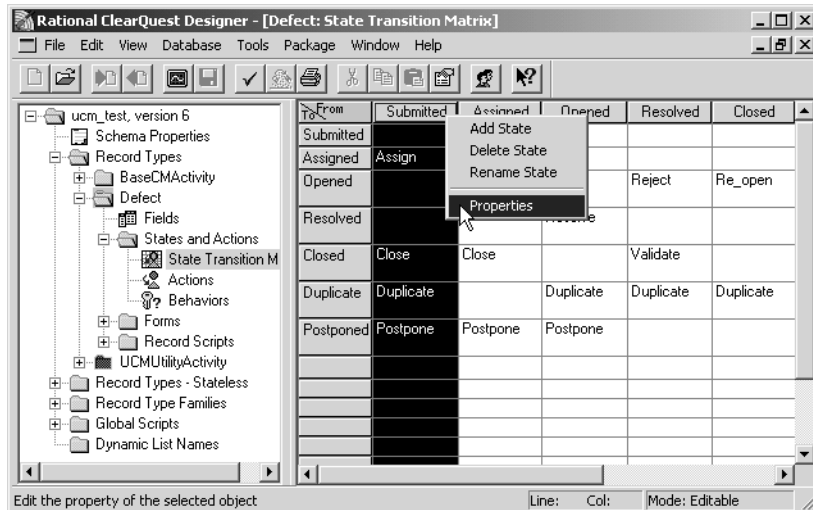
Setting the default actions for UCM

The State Transition Matrix in your schema must provide at least one path through the state type model for the UnifiedChangeManagement package, from the Waiting state type, to Ready, to Active, to Complete. See “Using the State Transition Matrix” on page 76, and “UnifiedChangeManagement package state type model” on page 269 for more information.

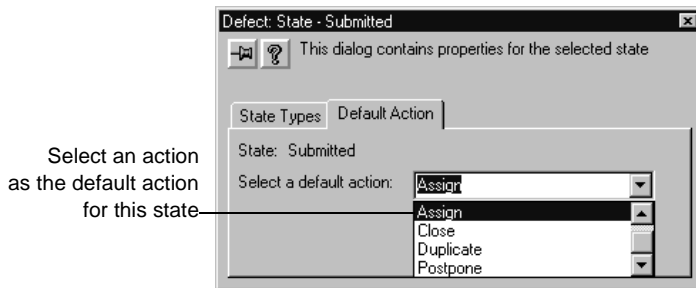
For each state in your schema, except the state mapped to the UCM Complete state, you must assign a default action that moves the record from that state to the next state type in the UCM state type model. See “Using default actions” on page 87 for more information.

To assign default actions:

- 1 In ClearQuest Designer, expand **Record Types**, then expand the record type you enabled for UCM, and double-click its **State Transition Matrix**.
- 2 In the State Transition Matrix, right-click a state and select **Properties** to open the Properties dialog box for that state.



- 3 In the **Default Action** tab of the Properties dialog box, 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.

- 4 Select **File > Check In** to check in the schema. See “Checking in a schema” on page 40 for detailed instructions.

For more information, see “Using default actions” on page 87 and “UnifiedChangeManagement package state type model” on page 269.

Adding the UCMPolicyScripts package

- 1 With the schema checked in, select package (**Package > Package Wizard** add the latest **UCMPolicyScripts**. See “Working with packages” on page 46 for detailed instructions.
- 2 Select **File > Check In** to check in the schema.

Adding the UnifiedChangeManagement package

- 1 With the schema checked in, select package (**Package > Package Wizard** add the latest **UnifiedChangeManagement** package.
- 2 Select **File > Check In** to check in the schema.

Adding the BaseCMAActivity package (optional)

The BaseCMAActivity package adds a lightweight activity record type to your schema. You can use this alternative to the Defect record type as is, enable it for UCM, or develop it into a new record type. This package is optional. For a more intense activity tracker, see *Using Rational ClearQuest Project Tracker*.

- 1 With the schema checked in, select package (**Package > Package Wizard** add the latest **BaseCMAActivity** package.
- 2 Select **File > Check In** to check in the schema.

Saving the schema changes

After installing the last UCM package:

- 1 Validate the schema changes. See “Validating schema changes” on page 39 for detailed instructions.
- 2 Select **File > Check In** to save the schema changes. See “Checking in a schema” on page 40 for detailed instructions.
- 3 Apply schema changes to the user database by selecting **Database > Upgrade Database**. See “Applying schema changes to a user database” on page 41 for detailed instructions.

Configuring Rational UCM

Configure the UCM application as needed. See *Managing Software Projects with ClearCase* for additional configuration information.

Adding a Microsoft Visual SourceSafe integration

The Visual SourceSafe integration adds Visual SourceSafe fields to a record type you select, and the Visual SourceSafe tab to the record type form.

The Microsoft Visual SourceSafe integration requires:

- Adding the Visual SourceSafe package
- Saving the schema changes
- Creating a query in ClearQuest client
- Setting up each client machine

For instructions on using the integration after completing these steps, see the online Help for the ClearQuest Visual SourceSafe tool.

Note: To find out if the Visual SourceSafe package is already installed in your schema, see “Viewing the packages in your schema” on page 275. If you already have these packages in your schema, and you just want to apply it to a new record type, see “Enabling record types for integrations” on page 274 for instructions.

Adding the Visual SourceSafe package

Note: You can only apply a Visual SourceSafe package to one record type in a schema.

- 1 In ClearQuest Designer, make sure the schema you want to add the package to is checked in. To check in a schema, select **File > Check In**.
- 2 Select **Package > Package Wizard** and add the latest **Visual SourceSafe** package. See “Working with packages” on page 46 for detailed instructions.

Saving the schema changes

After installing the Visual SourceSafe package:

- 1 Validate the schema changes. See “Validating schema changes” on page 39 for detailed instructions.
- 2 Select **File > Check In** to save the schema changes. See “Checking in a schema” on page 40 for detailed instructions.
- 3 Apply schema changes to the user database by selecting **Database > Upgrade Database**. See “Applying schema changes to a user database” on page 41 for detailed instructions.

Creating a query in ClearQuest client

You need to create a query that users can use to find the records you want to associate with Visual SourceSafe projects.

To create a query:

- 1 Log in to the ClearQuest client as a user with Public Folder privileges.
- 2 Define the query. See the ClearQuest client online Help for detailed instructions.

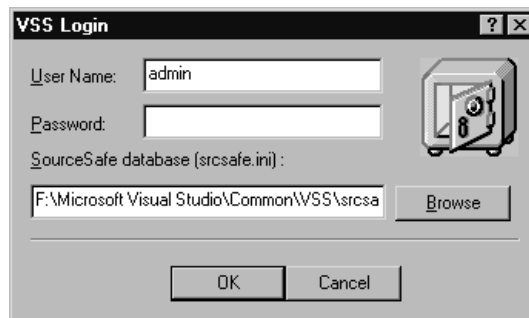
Setting up each client machine

Note: When you run the ClearQuest Visual Source Safe tool, `cqvss.exe`, the information you enter is stored in a `.ini` file, in the following location:

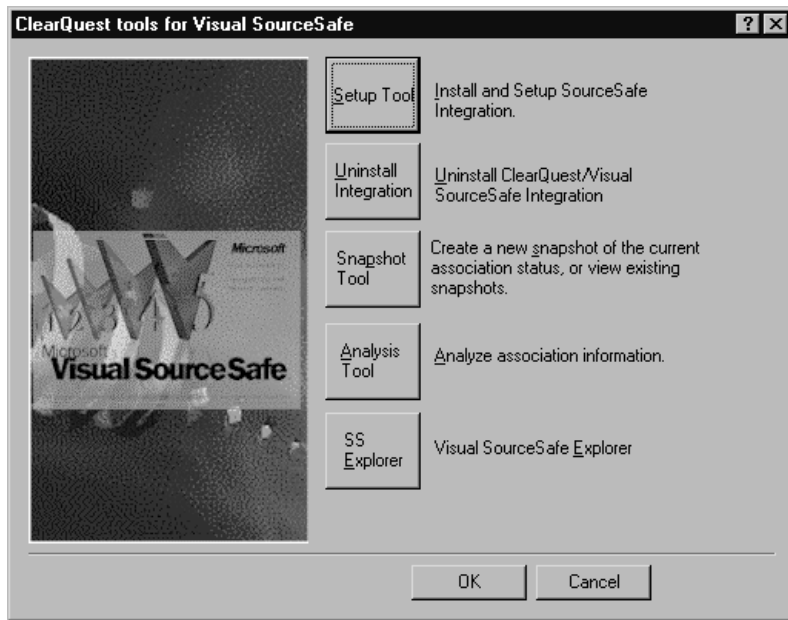
`vss\users\<user_name directory>\cqsspref.ini`

On each client machine where the Visual Source Safe integration will be used:

- 1 Start the ClearQuest Visual Source Safe tool (`cqvss.exe`) located in the ClearQuest home directory. You can drag a copy of `cqvss.exe` to the desktop to create a shortcut to it.
- 2 Log in and select the Visual SourceSafe database you want to use.



- 3 In the ClearQuest tools for Visual SourceSafe dialog box, select **Setup Tool**.



- 4 In the Visual SourceSafe Integration dialog box, select the ClearQuest query you created in “Creating a query in ClearQuest client” on page 291 and click **OK**.

Index

A

access control

- hook for 151, 187, 191
- user privileges 138

action hooks

- Access Control 187, 191
- adding 86
- adding new 189
- Commit 188, 193
- deleting 190
- editing 190
- execution order of 191
- for setting value of parent record 212
- Initialization 187, 191
- Initialization example 211
- Notification 188, 193
- reusing with Base action type 83
- validating fields 187
- Validation 187, 192
- see also* hook examples

action types

- Base 83
- Change_state 83, 84
- Delete 83
- Duplicate 83
- Import 83
- Modify 83
- Record_script_alias 83
- Submit 83
- Unduplicate 83

actions

- adding 84, 85
- and state model 75
- and State Transition Matrix 76
- creating state transitions 84
- default 87
- default for UCM 287
- deleting 88
- modifying 85
- overview 8
- properties 82, 85

- restricting access to 151

see also action hooks, action types

Actions grid 82

Active User privileges 138

admin user ID 1, 135

administrator, ClearQuest

- defined 1

- overview of tasks 10

AMBaseActivity package

- ClearQuest Project Tracker 285

- ClearQuest Project Tracker, for 280, 282

AnalystStudio schema 44, 258

API, ClearQuest

- common calls 204

- overview 5, 202

- records (entities) 203

- sessions 202

- using to build queries 202

- writing external applications 201

architecture, ClearQuest 5

assigning records to inactive users 147

ATTACHMENT_LIST data type 65, 243

attachments

- import file format 245

- importing 246

Attachments package 259

B

back reference fields 73

backups, database 13

Base action type 83

Base ClearCase

- integrating 272, 276

- predefined ClearCase policies (note) 284

BaseCMAActivity package 260

- UCM 289

behavior, *see* field behavior

Behaviors grid 66

Blank schema 37, 44, 258

BuildEntity method 203

C

- Change_state action type 83, 84
- characters, restricted for user login 140
- charts, using in ClearQuest Web 217
- checking in schemas 40
- checking out schemas 35
 - undoing check out 41
- child/parent, *see* parent/child
- choice lists
 - and inactive users 147
 - field hook 175
 - hook example 208
 - limiting value options 180, 183
 - reinitializing 183, 220
- ClearCase package 260
 - Base ClearCase, for 276
- ClearCase, Base
 - integrating 272, 276
- ClearCase, UCM
 - integrating 273
- ClearQuest
 - architecture 5
 - client 6
 - documentation xiii
 - Export Tool, *see* Export Tool, ClearQuest
 - getting users started 11
 - Import Tool, *see* Import Tool, ClearQuest
 - integrations 272
 - Maintenance Tool, *see* Maintenance Tool, ClearQuest
 - Web, *see* Web, ClearQuest
- ClearQuest Designer 6
 - starting 1
 - tutorial xiii
- ClearQuest Project Tracker
 - AMBaseActivity package 285
 - AMBaseActivity package for 280, 282
 - configuring 281
 - integrating 273, 280
- closing schemas 36
- code
 - reusing with Base action type 83
 - see also* hooks, scripts
- COM objects 173
- Commit action hook 188
- common fields 57
- Common schema 44, 258
- connecting to ClearQuest 17
- constant list field hook 180
 - selecting behavior for 183
- constant value field hook
 - selecting behavior for 183
- ContentStudio
 - integrating 272, 276
- ContentStudio package
 - ContentStudio, for 276
- Context Groups control 159
- Control palette 2
- controlled record type 158
- controls, list of
 - Attachment 111, 112
 - Checkbox 113
 - Combo box 114
 - Drop-down combo box 118
 - Drop-down list box 116
 - Duplicate base 120
 - Duplicate dependents 121
 - Group box 122
 - History 122
 - List box 123
 - List view 125
 - Option button 127
 - Parent/Child 128
 - Picture 129
 - Push button 130
 - Static text box 131
 - Text box 132
- copying
 - tabs 95
- CQLOAD 46
- Customer package 261

D

- data
 - backing up 13
 - defining 52
 - exporting from ClearQuest database 6, 252

- exporting from other systems 241
 - how stored 13
 - linking records to share 70
- data import 237, 244
 - attachments 246
 - attachments file format 245
 - delimiter 241, 247, 248
 - discarded data file 247, 248
 - duplicate records 250
 - empty fields 242
 - existing records 251
 - exporting from other systems 241
 - file format 241
 - from discarded data log 251
 - history file format 244
 - history information 246
 - original (old) record ID 240
 - overview of process 238
 - performing imports 246
 - reference fields 242
 - supported data types 243
 - unsupported data types 243
- data types
 - field 65
 - fields in import file 243
 - mapping unsupported 243
- database properties, viewing 29
- databases
 - accessing with API 202
 - backing up 13
 - changing vendor 20
 - changing vendors 20
 - creating new 15
 - creating user 16
 - deleting 27
 - maintenance overview 14
 - master, schema repository 7
 - moving 20
 - required 7, 13
 - test database 33
 - undeleting 28
 - see also* user databases, schema repository
- date format in import file 244
- DATE_TIME data type 65, 243
- DB Column Name 63, 68, 69
- DB Name 55, 58
- DBID data type 65
- default
 - actions for states 87
 - actions for UCM 78, 287
 - field behavior 67
 - record type 57
 - user ID 1, 135
- Default Value field hook 175
- Defect record type state model 266
- DefectTracking schema 44, 258
- Delete action type 83
- deleting
 - actions 88
 - fields 69
 - forms 93
 - record type families 60
 - record types 60
 - schemas 43
 - states 80
 - system fields 69
 - tabs from forms 95
 - user databases 27
- delimiters for import files 241, 247, 248
- dependent fields
 - creating 178
 - enabling for web client 179, 219
- dependent list hook example 207
- Designer toolbar 2
- destination state 85
- DevelopmentStudio schema 44, 258
- diagram, *see* state model and state type model
- discarded data file 247, 248
 - importing from 251
- discarded data log file 251
- display_name field 244, 245
- documentation, ClearQuest xiii
- double quotes in imported records 242
- Duplicate action type 83
- duplicate records
 - importing 250
- duplicate records, importing 250
- dynamic list field hook 180, 181
 - selecting behavior for 183

E

- e-mail
 - configuring ClearQuest users 226
 - Email_Rule record type 225
 - fields to include 229
 - format example 229
 - format for submitting 228
 - format guidelines 228
 - integrating 273
 - multi-line fields 229
 - notification 224
 - round-trip 230
 - triggering notification 224
- e-mail format 228
- Email package 261
- E-mail Reader, Rational 5, 6
 - configuring 227
- e-mail rules
 - access privileges 224
 - basing on a user group 148
 - creating 224
 - triggering notification 224
- Email_Rule record type 225
- empty fields, importing 242
- Enhancement Request package 262
- EnhancementRequest record type
 - state model 267
- Enterprise schema 44, 258
- entity 203
- Entity object 174, 199, 202, 204
- error log 278
- EventObject 196
- execution order of hooks 191
- Export Tool, ClearQuest 5, 6
 - using 252
- exporting
 - data from ClearQuest 237
 - data from ClearQuest database 6
 - data from other systems 241
 - forms 109
 - users and groups 152
- external applications
 - and user groups 148
 - writing 201

F

- family, *see* record type families
- field behavior
 - default 67
 - for choice list fields 183
 - reset by hooks 172
 - supported 66
- field hooks
 - adding 61, 74
 - Choice List 175, 191, 192
 - creating new 176
 - Default Value 175, 191
 - deleting 177
 - Dynamic-List 180
 - editing 177
 - execution order of 191
 - Permission 175, 191
 - Validation 175, 191, 192
 - Value Changed 175, 192
 - see also* hook examples
- Field List palette 2
- field names, changing 68
- fields
 - back reference 73
 - changing name of 68
 - common 57
 - creating 61
 - data type, selecting 62
 - data types for import file 243
 - data types, changing 65
 - data types, supported 65
 - DB Column Name 63, 68, 69
 - default behavior 67
 - defining behavior 52, 66
 - deleting 69
 - dependent 178
 - dependent for web client 179
 - e-mail format for multi-line 229
 - Help text for 64
 - initializing values 211
 - linking records in 70, 72
 - maximum length for SHORT_STRING 63
 - modified by packages 259
 - modifying 61

- reserved keywords in names 63
- system fields 61, 69
- Visible in Query 62
- see also* field behavior, field hooks, and controls, form
- file format
 - for importing 241
 - for importing attachments 245
 - for importing history 244
- Finding hook script text 206
- fonts, changing in forms 92
- form control events 196
- Form Layout toolbar 2
- forms
 - adding tabs to 93
 - changing font of 92
 - changing name of 92
 - changing tab order in 94
 - copying tabs in 95
 - creating 53, 91
 - creating for multiple platforms 108
 - deleting 93
 - deleting tabs from 95
 - exporting 109
 - importing 110
 - renaming tabs on 93
 - resizing 92
 - restricting access to tabs 94
 - reusing 109
- Forms window 90

G

- GetSession method 202
- global scripts 172, 199
 - creating new 200
 - deleting 201
 - editing 200
 - Entity object 174
 - understanding 199
- groups, user
 - adding 139
 - controlling access to actions 151
 - creating sgroups 149

- importing and exporting 152
- making inactive 147
- subscribing to databases 136, 150
- uses for 148

H

- Help text, adding to fields 64
- history
 - import file format 244
 - importing 246
- History package 262
- hook examples 207
 - choice list 208
 - dependent list 207
 - hook for initializing field value 211
 - hook to set value of parent record 212
- hooks
 - and global scripts 199
 - and Super User privileges 172
 - and user groups 148
 - COM objects 173
 - editing script text 206
 - execution order of 191
 - fields 175
 - finding script text 206
 - for actions 86, 187
 - for detecting web session 221
 - for fields 74
 - important considerations 172
 - in web client 219
 - overview 172
 - planning 53
 - resetting read-only field value 172
 - reusing action hooks 83
 - scripting language for 38
 - see also* hook examples, action hooks, field hooks

I

- ID data type 65
- ID, user
 - admin default 1, 135

- Windows user profiles 135
- import
 - discarded data file 247, 248
- Import action type 83
- import file
 - creating 241
 - field data types supported 243
 - format 241
 - format considerations 242
- Import Tool, ClearQuest 5, 6
- importing
 - attachments 246
 - attachments file format 245
 - creating import file 241
 - data from another system 246
 - data into ClearQuest 237
 - date formats 244
 - delimiter 241, 247, 248
 - discarded data file 247, 248
 - duplicate records 250
 - embedded double quotes 242
 - empty fields 242
 - file format 241
 - forms 110
 - from discarded datafile 251
 - history 246
 - history file format 244
 - original (old) record ID 240
 - overview of process 238
 - performing imports 246
 - reference fields 242
 - upgrading existing records 251
 - users and groups 152
- inactive users and user groups 147
- Initialization action hook 187, 211
- INT data type 65, 243
- integrating
 - errors when 278
- integrations
 - add package failures 278
 - analyzing current 275
 - Base ClearCase 276
 - ClearCase, Base 276
 - ClearQuest 272
 - ClearQuest Project Tracker 280
 - ContentStudio 276
 - dependent 273, 278
 - e-mail system/ClearQuest 273
 - e-mail system/ClearQuest (note) 273
 - errors 278
 - independent 272, 276
 - Microsoft Visual SourceSafe 290
 - Microsoft Visual SourceSafe/ClearQuest 273
 - PureCoverage 276
 - Purify 276
 - Quantify 276
 - Rational Administrator 279
 - Rational Administrator/ClearQuest 273
 - Rational Base ClearCase/ClearQuest 272
 - Rational ClearQuest Project
 - Tracker/ClearQuest 273
 - Rational PureCoverage/ClearQuest 272
 - Rational Purify/ClearQuest 272
 - Rational Quantify/ClearQuest 272
 - Rational RequisitePro/ClearQuest 273
 - Rational Suite
 - ContentStudio/ClearQuest 272
 - Rational TeamTest/ClearQuest 273
 - TeamTest 282
 - test database 276
 - testing 276
 - UCM 284
 - Unified Change Management (UCM) 273

J

- JOURNAL data type 65

K

- key, unique 55

L

- languages, scripting 38
- length, maximum for SHORT_STRING data
 - type 63
- Limit to list 183

- lists
 - and inactive users 147
 - choice 180, 183
 - dynamic 175
 - hook for dependent 207
- logging in
 - and inactive users 147
 - default admin ID 1
 - without checking out a schema 36
- login, restricted characters for 140

M

- mailreader.exe 227
- Maintenance Tool, ClearQuest 5, 6
- maintenance, database 14
- mandatory field behavior 66
- mapping
 - state types 78
 - state types for UCM 286
- master database, *see* schema repository
- message boxes, in ClearQuest Web 220
- metadata 7, 31
- method
 - BuildEntity 203
 - GetSession 202
- Microsoft Visual SourceSafe
 - configuring client 291
 - integrating 273, 290
 - query for 291
- model, *see* state model, state type model
- Modify action type 83
- moving
 - ClearQuest database 20
 - databases 20
 - schema repository 20, 21
 - user databases 20, 23, 45
- MULTILINE_STRING data type 65, 243

N

- Notes package 262
- notification
 - action hook 188

- hook 172
- setting up e-mail rules 224
- triggering 224

O

- object
 - Entity 202, 204
 - QueryDef 202
 - ResultSet 202
 - session 202, 204
- optional field behavior 66
- original (old) record ID 240

P

- Package Wizard 47
- packages
 - adding to schemas 47
 - Attachments 259
 - BaseCMAActivity 260
 - ClearCase 260, 276
 - ContentStudio 276
 - Customer 261
 - Email 261
 - enabling record types 48, 274
 - Enhancement Request 262
 - History 262
 - in schema 275
 - list of predefined 259
 - mapping state types for 78
 - Notes 262
 - Package Wizard 47
 - PQC 262, 276
 - Project 262
 - Repository 263
 - requirements for 46
 - RequisitePro 263
 - Resolution 263
 - TeamTest 264
 - UCM 265
 - Visual SourceSafe 265
- palettes 2
- parent/child

- hierarchy 72
- hook to set value of parent record 212
- record linking 72
- Perl scripting language 38
 - COM objects 173
- Permission field hook 175
- permissions, *see* privileges
- PQC package 262
 - PureCoverage, for 276
 - Purify, for 276
 - Quantify, for 276
- privileges
 - Active User 138
 - for schema design 31, 51
 - Schema Designer 138
 - Super User 138
 - system 145
 - types of 138
 - User Administrator 138
- privileges, user 145
- Project package 262
- properties
 - action 82, 85
 - schema 38
 - setting for schemas 38
 - state types and packages 78
- PureCoverage
 - integrating 272, 276
- Purify
 - integrating 272, 276

Q

- Quantify
 - integrating 272, 276
- query
 - field Visible in Query 62
 - multiple record types 57
 - using API to build 202
- QueryDef object 202

R

- Rational Administrator

- configuring 279
- integrating 273, 279
- Repository package for 279
 - user database 279, 281, 283, 289
- Rational E-mail Reader 5, 6
 - configuring 227
- Rational Software web site xv
- Rational suites
 - schemas for 257
- ratl_replicas system record type 60
- Ratl_Security tab 159
- read-only field behavior 66
 - values reset by hooks 172
- read-only fields
 - values for e-mail submission 229
- Recalculate Choice list 183, 220
- record scripts 194
 - adding to record type 197
 - deleting 198
 - editing 198
 - form control events 196
 - overview 194
 - using on ClearQuest Web 195
- record type families 54
 - adding members 58
 - common fields 59
 - common fields for 59
 - creating 57
 - deleting 60
 - naming conventions 57
 - removing members 59
 - renaming 60
- record types 8, 54
 - adding with packages 55
 - applying packages to new 274
 - applying packages to new (note) 48
 - creating 55, 57
 - DB Name 55, 58
 - default 57
 - Defect 266
 - deleting 60
 - Email_rule 225
 - enabling for UCM 285
 - EnhancementRequest 267
 - fields 61

- querying across multiple 54
- re-applying packages to (note) 48
- renaming 60
- selecting default 57
- state-based 54
- stateless 54
- system 55
 - see also* record type families
- Record_script_alias action type 83
- records
 - assigning to inactive users 147
 - e-mail format 228
 - entity 203
 - hiding 158
 - importing duplicate 250
 - linking 70, 72
 - parent/child linking 72
 - upgrading existing 251
- REFERENCE data type 65, 243
- reference fields, importing 242
- REFERENCE_LIST data type 65, 243
- remote access, *see* Web, ClearQuest
- renaming
 - fields 68
 - forms 92
 - record type families 60
 - record types 60
 - states 79
 - system fields 69
 - tabs on forms 93
- reports on ClearQuest Web 217
- Repository package 263
 - Rational Administrator, for 279
- RequisitePro
 - integrating 273
- RequisitePro package 263
- reserved keywords in field names 63
- resizing forms 92
- Resolution package
 - description 263
 - state type model 268
- ResultSet object 202
- reusing forms 109
- round-trip e-mail 230

S

- SAMPL database, connecting to 14
- saving
 - schemas 32
 - status messages 25
- schema
 - viewing packages in 275
- Schema Designer privileges 138
- schema properties 38
- schema repository
 - connecting to 6, 14
 - creating new 14, 15
 - defined 7
 - importing schemas into with CQLOAD 46
 - metadata 7
 - moving 20, 21
 - updateschema repository
 - changing database vendor 22
- schema repository
 - moving 20
- schemas
 - adding packages to 47
 - AnalystStudio 44, 258
 - Blank 37, 44, 258
 - changing 45
 - checking in 40
 - checking out 35
 - closing 36
 - Common 44, 258
 - creating 37
 - customizing 51
 - customizing overview 52
 - default record type for 57
 - DefectTracking 44, 258
 - deleting 43
 - DevelopmentStudio 44, 258
 - editing over multiple sessions 42
 - Enterprise 44, 258
 - for Rational suites 257
 - importing/exporting with CQLOAD 46
 - opening 36
 - overview 7, 32
 - packages 44, 46
 - packages in 259

- predefined 44, 258
- privileges required to customize 31, 51
- properties of 38
- saving 32
- saving work in progress 42
- scripting language for 38
- selecting 44
- testing 33
- TestStudio 44, 258
- undo check out 41
- UnifiedChangeManagement 44, 258
- upgrading user databases with 41
- validating 39, 40
- version number 40
- viewing 36
- scripting languages, selecting 38
- scripts
 - access control 151
 - global 172, 199
 - record 194
 - selecting language for 38
- Security 157
- security
 - context 158
 - context field 159
 - controlled record types 158
 - designing 161
 - example 162
 - overview 158
 - security context record type 159
 - user groups 158
- session object 202, 204
- Set Test Database 33
- SHORT_STRING data type 65, 243
- source state 85
- STATE data type 65, 243
- state model
 - Defect record type 266
 - defining 8, 52, 75
 - EnhancementRequest record type 267
- State Transition Matrix 76
- state transitions
 - creating 84
- state type model
 - packages 268
 - Resolution package 268
 - UCM package 269
- state types
 - editing (note) 50
 - mapping 78
 - mapping for UCM 286
 - Resolution package 268
 - see also* state type model
- state-based records 54
- stateless records 54
 - unique key 55, 56
- states 8, 75
 - action 76
 - adding 77
 - creating transitions 84
 - default actions for 87
 - deleting 80
 - destination 76, 85
 - renaming 79
 - source 76, 85
 - see also* state types, state transition
- status messages, saving 25
- subgroup for user groups 149
- subgroups, creating user 149
- Submit action type 83
- subscriptions, viewing database 137
- Super User privileges 138
 - and hooks 172
- support, technical xv
- system
 - fields 61, 69
 - record types 55
- system fields 69

T

- tabs 93
 - adding to forms 93
 - changing order of 94
 - changing the name of 93
 - copying 95
 - deleting 95
 - deleting from forms 95
 - restricting access to 94

- TeamTest
 - configuring 283
 - integrating 273, 282
 - TeamTest package 282
- TeamTest package 264
 - TeamTest integration 282
- technical support xv
- test database 33
- testing
 - integrations 276
- TestStudio schema 44, 258
- text, Help 64
- toolbars 2
- tutorial, ClearQuest Designer xiii

U

- UCM
 - BaseCMAActivity package 289
 - ClearCase version for 284
 - configuring 289
 - default actions 287
 - integrating 273, 284
 - predefined ClearCase policies (note) 284
 - user database 289
- undeleting user databases 28
- undoing schema check out 41
- Unduplicate action type 83
- Unified Change Management
 - and BaseCMAActivity 260
 - enabling record types 285
 - mapping state types 286
 - package 265
 - required packages 284
 - UCM-enabled schemas 284
 - UnifiedChangeManagement schema 44, 258
- UnifiedChangeManagement schema 44, 258
- unique key, stateless record type 55, 56
- UNIX
 - ClearQuest support for 6
 - scripting language for 38
- updating
 - schema repository 22
- Upgrade user DB 137
- Use_hook field behavior 66
- User Administration dialog 136, 139
- User Administrator privileges 138
- user databases 7
 - adding user information to 41, 137
 - applying schema changes to 41
 - associating with schema 41
 - changing database vendor 20, 23
 - creating 15, 16
 - defined 13
 - deleting 27
 - moving 20, 23, 45
 - subscribing users to 137, 142, 143
 - switching schemas 45
 - undeleting 28
 - unsubscribing users 143
 - upgrading with user information 144
 - viewing subscriptions 137
 - Visible to designer only 33
 - working with test database 33
- user groups
 - adding users 149
 - and external applications 148
 - and hooks 148
 - creating 148
 - displaying members 149, 150
 - importing and exporting 152
 - making iactive 147
 - making inactive 147
 - privileges for administering 135
 - removing users 149, 150
 - restricting access to actions 151
 - subgroups 149
 - subscribing to databases 136, 150
 - upgrading database with user group information 144
- user ID
 - admin default 1, 135
- user privileges 138
- users
 - adding new 139
 - adding to groups 149
 - administering 135
 - changing privileges 145
 - exporting and importing 152

- making active 147
- making inactive 147
- modifying profile 145
- privileges 138
- privileges required for administering 135
- removing from a group 150
- restricting access to actions 151
- restricting access to ClearQuest Web 218
- subscribing to databases 136, 142
- unsubscribing to databases 143
- upgrading database with user information 137, 144
- Windows accounts 135

V

- validating schemas 39, 40
- validation
 - action hook 172, 187
 - field hook 172, 175
- Value Changed field hook 175
- VBScript
 - COM objects 173
 - for Web hooks 219
 - selecting as scripting language 38
- version number of schemas 40
- viewing database properties 29

- Visible in Query 62
- Visible to designer only, test database 33
- Visual SourceSafe package 265

W

- web session
 - detecting 221
 - Perl example 221
 - VBScript example 221
- web site, Rational Software xv
- Web, ClearQuest 6
 - customizing 217, 218
 - dependent fields 179
 - dependent fields for 219
 - displaying messages on 220
 - hook for detecting web session 221
 - limiting access (web entry) 218
 - navigating back and forward 217
 - server 5
 - using charts in 217
 - using hooks on 219
 - using reports on 217
- Windows
 - ClearQuest for 6
 - user profiles 135
- workflow, defining in ClearQuest 8