Rational Software Corporation®

Rational® ClearQuest®

Administrator's Guide

VERSION: 2003.06.00 AND LATER

PART NUMBER: 800-026168-000

UNIX/WINDOWS EDITION

Rational®
the software development company

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

Legal Notices

Copyright ©1997-2003, Rational Software Corporation. All Rights Reserved.
Part Number: 800-026168-000
Version Number: 2003.06.00

This manual (the "Work") is protected under the copyright laws of the United States and/or other jurisdictions, as well as various international treaties. Any reproduction or distribution of the Work is expressly prohibited without the prior written consent of Rational Software Corporation.

The Work is furnished under a license and may be used or copied only in accordance with the terms of that license. Unless specifically allowed under the license, this manual or copies of it may not be provided or otherwise made available to any other person. No title to or ownership of the manual is transferred. Read the license agreement for complete terms.

Rational Software Corporation, Rational, Rational Suite, Rational Suite ContentStudio, Rational Apex, Rational Process Workbench, Rational Rose, Rational Summit, Rational Unified process, Rational Visual Test, AnalystStudio, ClearCase, ClearCase Attache, ClearCase MultiSite, ClearDDTS, ClearGuide, ClearQuest, PerformanceStudio, PureCoverage, Purify, Quantify, Requisite, RequisitePro, RUP, SiteCheck, SiteLoad, SoDa, TestFactory, TestFoundation, TestMate and TestStudio are registered trademarks of Rational Software Corporation in the United States and are trademarks or registered trademarks in other countries. The Rational logo, Connexis, ObjecTime, Rational Developer Network, RDN, ScriptAssure, and XDE, among others, are 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.

Portions covered by U.S. Patent Nos. 5,193,180 and 5,335,344 and 5,535,329 and 5,574,898 and 5,649,200 and 5,675,802 and 5,754,760 and 5,835,701 and 6,049,666 and 6,126,329 and 6,167,534 and 6,206,584. Additional U.S. Patents and International Patents pending.

U.S. Government Restricted Rights

Licensee agrees that this software and/or documentation is delivered as "commercial computer software," a "commercial item," or as "restricted computer software," as those terms are defined in DFARS 252.227, DFARS 252.211, FAR 2.101, OR FAR 52.227, (or any successor provisions thereto), whichever is applicable. The use, duplication, and disclosure of the software and/or documentation shall be subject to the terms and conditions set forth in the applicable Rational Software Corporation license agreement as provided in DFARS 227.7202, subsection (c) of FAR 52.227-19, or FAR 52.227-14, (or any successor provisions thereto), whichever is applicable.

Warranty Disclaimer

This document and its associated software may be used as stated in the underlying license agreement. Except as explicitly stated otherwise in such license agreement, and except to the extent prohibited or limited by law from jurisdiction to jurisdiction, 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, non-infringement, title or fitness for a particular purpose or arising from a course of dealing, usage or trade practice, and any warranty against interference with Licensee's quiet enjoyment of the product.

Third Party Notices, Code, Licenses, and Acknowledgements

Portions Copyright ©1992-1999, Summit Software Company. All rights reserved.

Microsoft, the Microsoft logo, Active Accessibility, Active Client, Active Desktop, Active Directory, ActiveMovie, Active Platform, ActiveStore, ActiveSync, ActiveX, Ask Maxwell, Authenticode, AutoSum, BackOffice, the BackOffice logo, bCentral, BizTalk, Bookshelf, ClearType, CodeView, DataTips, Developer Studio, Direct3D, DirectAnimation, DirectDraw, DirectInput, DirectX, DirectXJ, DoubleSpace, DriveSpace, FrontPage, Funstone, Genuine Microsoft Products logo, IntelliEye, the IntelliEye logo, IntelliMirror, IntelliSense, J/Direct, JScript, LineShare, Liquid Motion, Mapbase, MapManager, MapPoint, MapVision, Microsoft Agent logo, the Microsoft eMbedded Visual Tools logo, the Microsoft Internet Explorer logo, the Microsoft Office Compatible logo, Microsoft Press, the Microsoft Press logo, Microsoft QuickBasic, MS-DOS, MSDN, NetMeeting, NetShow, the Office logo, Outlook, PhotoDraw, PivotChart, PivotTable, PowerPoint, QuickAssembler, QuickShelf, RelayOne, Rushmore, SharePoint, SourceSafe, TipWizard, V-Chat, VideoFlash, Visual Basic, the Visual Basic logo, Visual C++, Visual C#, Visual FoxPro, Visual InterDev, Visual J++, Visual SourceSafe, Visual Studio, the Visual Studio logo, Vizact, WebBot, WebPIP, Win32, Win32s, Win64, Windows, the Windows CE logo, the Windows logo, Windows NT, the Windows Start logo, and XENIX, are either trademarks or registered trademarks of Microsoft Corporation in the United States and/or in other countries.

Sun, Sun Microsystems, the Sun Logo, Ultra, AnswerBook 2, medialib, OpenBoot, Solaris, Java, Java 3D, ShowMe TV, SunForum, SunVTS, SunFDDI, StarOffice, and SunPCi, among others, are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

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

Licensee shall not incorporate any GLOBEtrotter software (FLEXlm libraries and utilities) into any product or application the primary purpose of which is software license management.

BasicScript is a registered trademark of Summit Software, Inc.

Design Patterns: Elements of Reusable Object-Oriented Software, by Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides. Copyright © 1995 by Addison-Wesley Publishing Company, Inc. All rights reserved.

Copyright ©1997 OpenLink Software, Inc. All rights reserved.

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

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

Additional legal notices are described in the legal_information.html file that is included in your Rational software installation.

Contents

Preface xxv
About This Manual xxv
ClearQuest Documentation Roadmap xxvi
ClearQuest Integrations with Other Rational Products xxvii
Typographical Conventions xxviii
Customer Support xxix

1. Understanding ClearQuest Administration 1

Understanding the Concepts of Change Management 1
Why Change Management? 1
The Change Request Lifecycle 2
The Change Management Process Model 3
ClearQuest: A Defect and Change Tracking System for Change Management 5
Defining ClearQuest Schemas, Schema Repositories, Databases, and Connections 5
Schemas 5
Schema Repositories 6
Databases 6
Connections 7
Vendor Database Management Systems 8
Understanding the Architecture and Components of ClearQuest 8
ClearQuest Architecture and Components for Windows Installations 8
Customizations: Hooks and the ClearQuest API 11
ClearQuest Architecture for UNIX Installations 12
Understanding the Functions of the ClearQuest Administrator 14
Designing and Implementing the System 15
Configuring the Database Management System 16
Installing ClearQuest 16
Subscribing to the ClearQuest User Group 16
Creating and Managing Schema Repositories and Connections 16
Performing Code Page Tasks 16
Creating a User Database 17
Creating the Schema (the Process Model) 17
Managing User Databases 19

Enabling E-Mail Notification Through the Rational E-Mail Reader 19
Enabling Web Access 20
Importing and Exporting Data 20
Creating Queries, Charts, and Reports 20
Managing User Accounts 20
Setting Security Controls on Records 21
Delegating Administrative Functions 21
Setting Up ClearQuest MultiSite 22
Getting Started with ClearQuest 22
Setting a Password for Your Administrative Account 22
Toolbars, Palettes, Menus, and Shortcuts 22
Using Toolbars 23
Using Palettes 23
Using Menus 23
Using the Shortcut Menu 23
Using Keyboard Shortcut and Mouse Actions 23

2. ClearQuest and Code Pages 27

Code Page Task Roadmap 27
About the ClearQuest Data Code Page 28
ClearQuest Data Code Page and New Schema Repositories 28
ClearQuest Data Code Page and Existing Schema Repositories 29
Selecting a ClearQuest Data Code Page 29
Guidelines for Selecting a ClearQuest Data Code Page 29
Rational Supported Windows Code Pages 31
Consequences of Your ClearQuest Data Code Page Selection 31
Changing the ClearQuest Data Code Page Value 32
About the Vendor Database Character Set 32
Example 33
Setting the Vendor Database Character Set 34
Setting the ClearQuest Data Code Page Value 35
New Schema Repositories 35
Existing Schema Repositories and User Databases 36
Interoperation with Previous Versions of ClearQuest 38
About the CharacterSetValidation Package 38
Applying the CharacterSetValidation Package 39
Copying the ClearQuest Data Code Page Value to User Databases 40
MultiSite and the ClearQuest Data Code Page Value 40

MultiSite Synchronization Failure Example 41
ClearQuest Integrations and the ClearQuest Data Code Page Value 41
Examples 42

3. Managing Databases 43

Understanding the Process for Working with Databases 44
Establishing Procedures to Back Up Databases 47
Creating Empty Vendor Databases and Database Aliases 47
Setting the Vendor Database Character Set 48
Creating New Schema Repositories and Sample Databases 49
Creating User Databases 52
Managing Connections 55
Creating New Connections 55
Modifying Connections 56
Renaming a Connection 56
Editing a Connection 57
Deleting a Connection 57
Duplicating an Existing Connection 58
Maintaining Connections with a UNIX Client 58
cqreg Options 59
Using Connection Profiles 60
Exporting a Connection Profile 60
Importing a Connection Profile 61
Moving Schema Repositories 62
Preparing to Move Schema Repositories 63
Moving a Schema Repository 63
Moving a Schema Repository Using Vendor Database Tools 64
Updating an Existing Connection 65
Updating Schema Repositories Without Connections 65
Upgrading Databases to New Feature Levels of ClearQuest 66
Upgrading from SQL Anywhere 5.0 to 8.0 66
Upgrading an Existing Connection 66
Upgrading Schema Repositories and User Databases Without Connections 67
Moving and Deleting User Databases 68
Preparing to Move a User Database 68
Moving a User Database 69
Updating the Properties of a User Database 70

Deleting User Databases 71
Restoring a Deleted Database 72
Viewing Database Properties 73
Vendor Database Parameters 73
Microsoft Access 74
SQL Anywhere 74
SQL Server 75
Oracle 75
DB2 76

4. Working with ClearQuest Schemas 79

Procedures for Modifying a Schema 80
Creating a Test Database 80
Checking Out a Schema 82
Opening a Schema for Viewing Only 83
Creating a New Schema from the Blank Schema 84
Selecting a Scripting Language 85
Validating Schema Changes 86
Setting the Test Database for the Schema 87
Testing the Schema with a Test Database 88
Checking In a Schema 88
Undoing a Schema Checkout 89
Upgrading a User Database 89
Saving Work in Progress 90
Deleting a Schema or Schema Version 91
Changing a Database to a Different Schema 92

5. Customizing a Schema 93

Overview of Customizing a Schema 93
Working with Record Types 94
State-Based and Stateless Record Types 94
How Many Record Types? 94
Adding a New Record Type 95
Selecting a Unique Key for a Stateless Record Type 96
Selecting a Default Record Type for a Schema 97
Making a Record Type the Default 97
Working with Record Type Families 97
Adding a Record Type Family 97

Adding Members to a Record Type Family 98
Creating Common Fields 99
Removing Members from a Record Type Family 99
Renaming a Record Type or Family 99
Deleting a Record Type or Family 100
Working with Fields 100
Adding a New Field 101
Considerations for Selecting Field Data Types 103
Adding Help Text to a Field 105
Defining Field Behavior 106
Defining Default Field Behavior 107
Modifying a Field 107
Changing the Name of a Field 108
Deleting a Field 108
Using Fields to Link Records 109
Linking Records to Create a Parent/Child Hierarchy 110
Customizing Fields by Adding Hooks 112
Overview of State Models 113
Customizing State Models with the State Transition Matrix 115
Displaying a State Transition Matrix 115
Adding a New State 116
Mapping State Types 116
Changing the Name of a State 117
Deleting a State 118
Working with Actions and Action Types 119
Understanding Actions and Action Types 119
Supported Action Types 120
Adding a New Action 121
Creating a State Transition 122
Modifying Actions 123
Customizing Actions by Adding Hooks 123
Using Default Actions 124
Deleting an Action 125
Considerations in Designing State Models 125
Black Holes 125
How Many States? 125
Duplicates 126
Predefined Schemas 126

Parallel Development 127

6. Applying Packages 129

Overview of Packages and Integrations 129
Packages and Predefined Schemas 131
Cautions About Applying Packages 131
Packages Available 132
Examples: The Notes, Attachments, and E-Mail Packages 133
Checking Which Packages Have Been Applied to a Schema 134
How to Apply a Package 135
Enabling Record Types 138
Mapping State Types 139
Creating Default Actions 140
How to Upgrade Packages 140

7. Working with Forms 143

Overview of Working with Forms 143
Creating and Modifying Forms 144
Creating a New Form 144
Changing the Name of a Form 145
Changing the Size of a Form 145
Changing the Font on Forms 145
Deleting a Form 146
Creating Forms for Multiple Platforms 146
Creating and Modifying Forms for Imported Data 146
Working with Form Controls 146
Adding Controls to a Form 148
Opening the Form 149
Adding a Control Using the Control Palette 149
Adding a Control Using the Field List 150
Adding a Control Using the Form Controls Menu 152
Selecting Controls in a Form 153
Editing Control Properties 153
Deleting a Control from a Form 154
Changing the Size and Location of Controls 154
Moving Controls 155
Aligning Controls 156
Resizing Controls 156

Changing the Tab Order of Controls 156
Working with Tabs on Forms 157
Adding Tabs to a Form 157
Changing the Name of a Tab 157
Restricting Access to a Tab 158
Changing the Order of Tabs 158
Deleting Tabs 159
Copying Tabs 159
Reusing Record Forms 159
Exporting a Form 160
Importing a Form 160

8. Administering Users 161

Overview of User Administration 161
Viewing Database Subscriptions 162
ClearQuest User Privileges 163
Working with Users 164
Adding a New User 164
Assigning User Access Privileges 166
Subscribing Users and Groups to Databases 167
Unsubscribing Users and Groups from Databases 168
Applying Schema Changes to the User Database 169
Editing Users 169
Changing User Privileges 170
Editing a User Profile from the ClearQuest Client 170
Making Users and Groups Inactive 171
Working with User Groups 172
Creating a New User Group 172
Creating Subgroups 173
Adding Users to a Group 174
Removing Users or Subgroups from a Group 174
Subscribing User Groups to Databases 174
Restricting User Access to Actions 174
Exporting and Importing Users and User Groups 175
Administering Users in a MultiSite Environment 176
How Mastership Affects ClearQuest Client Users 176
How Mastership Affects User Administration 177
Establishing Where Users and Groups Are Currently Mastered 177

Changing the Mastership of a User 177
Changing Mastership 178
Synchronizing the Replicas After Mastership Changes 178
Upgrading the User Database with the Mastership Change 178

9. Using Security in Rational ClearQuest 179

Hiding Records in ClearQuest 179
Deciding Which Record Types to Control 180
Creating User Groups According to Security Context 180
Deciding Which Record Type to Use as the Security Context 181
Creating a Security Context Field 181
Submitting Security Context Records 181
Editing Records to Allow User Access 182
Designing a Security System 182
Security Example 184
Checking Out a Schema 185
Creating a Security Context Field 185
Adding the Security Context Field to the Form 186
Applying the Schema Changes 186
Creating the User Groups 186
Submitting the Security Context Records 187
Associating Groups with Each Security Context Record 187
Editing Records to Grant Privileges to Groups 188
Using Other ClearQuest Security Features 188
Restricting Access to Fields 188
Restricting Access to Actions 189
Restrict Access to Dialog Tabs 189
Adding Password Protection 189

10. Using Hooks to Customize Your Workflow 191

Understanding Hooks 192
Important Hook Considerations 192
Writing Scripts 193
Operating Context for Using Scripts 194
Working with Field Hooks 194
Adding a Field Hook 195
Editing a Field Hook 196
Deleting a Field Hook 197
Creating a Dependency Between Fields 197

Enabling Dependent Fields for ClearQuest Web 198
Creating a Choice List for a Field 199
Creating a Dynamic Choice List 199
Editing a Dynamic List 201
Defining Choice List Behavior 202
Specifying a Default Value for a Field 202
Validating User Input in a Field 203
Working with Action Hooks 204
Adding an Action Hook 206
Editing an Action Hook 207
Deleting an Action Hook 207
Execution Order of Field and Action Hooks 208
When an Action Begins 208
When a Field Value Is Set 209
When the Record Is Validated 209
When the Record Is Committed 209
Working with Record Scripts 210
Understanding Record Scripts 210
Using Record Scripts on ClearQuest Web 211
Form Control Events 212
Adding a Record Script to a Record Type 213
Editing a Record Script 214
Deleting a Record Script 214
Working with Global Scripts 214
Understanding Global Scripts 215
Creating a Global Script 215
Editing a Global Script 216
Deleting a Global Script 216
Writing External Applications 217
Using the ClearQuest API 217
Working with Sessions 217
Working with Queries 218
Working with Records 218
Common API Calls 218
Finding Text in Hook Scripts 220

11. Administering ClearQuest Web 223

ClearQuest Web Considerations 223

ClearQuest Data Code Page and ClearQuest Web 224
Customizing ClearQuest Web 224
ClearQuest Web Settings 224
Running Field Scripts 226
Enable E-mail Notification Settings 227
Presentation Scheme Settings 227
Working with Predefined Presentation Schemes 229
Using a Predefined Presentation Scheme 229
Editing a Presentation Scheme 229
Applying a Different Presentation Scheme 230
Deleting a Presentation Scheme 230
Renaming a Presentation Scheme 230
Creating a New Presentation Scheme 230
Limiting Access to ClearQuest Web 230
Using Hooks in ClearQuest Web 231
Enabling Dependent Fields for ClearQuest Web 231
Displaying Messages on ClearQuest Web 232
Using Hooks to Detect a Web Session 232
Web Session Detection in VBScript 232
Web Session Detection in Perl 233

12. Administering ClearQuest E-Mail 235

Overview of ClearQuest E-Mail Features 235
Enabling Automatic E-Mail Notification 236
Creating E-Mail Rules 236
Specifying Rule Controls for an E-Mail Rule 237
Specifying Action Controls for an E-Mail Rule 238
Specifying Display Fields for an E-Mail Rule 239
Specifying E-Mail Addresses for an E-Mail Rule 240
Specifying cc Addresses for an E-Mail Rule 241
Sample E-Mail Rules 242
Finding Existing E-Mail Rules 243
Modifying an Existing E-Mail Rule 243
Administering E-Mail Addresses 244
Hidden Users 244
Enabling ClearQuest Clients to Send E-Mail 244
Enabling E-Mail Notification for ClearQuest Web 244
Using Round-Trip E-Mail 245
Using an Action Hook to Send E-Mail 246

Enabling E-Mail Submission Through the Rational E-Mail Reader 247
Creating Mailboxes for the Rational E-Mail Reader 248
Configuring the Rational E-Mail Reader. 248
Configuring Additional Options for the Rational E-Mail Reader 253
Configuring the Rational ClearQuest Mail Service Properties 254
Formatting E-Mail for Submission 254
Guidelines for Formatting E-Mail 255
E-Mail Format Example 256
Format Example for Submitting a New Record 256
Format Example for Modifying a Record Using the Default Action 257

13. Importing and Exporting Data 259

Overview of Importing and Exporting Data 259
Starting the Import Process 259
Testing the Import Process 260
Creating an Import Schema 260
Including the Original Record ID 261
Creating a Database for Imported Data 262
Creating a ClearQuest Import File 262
Formatting Record Type Import Files. 263
Important Import-File Format Considerations 264
Data Types Supported in ClearQuest 265
Formatting History Import Files 266
Formatting Attachment Import Files 266
Exporting Data from ClearQuest 267
Importing Data into ClearQuest 271
Determining the Order of Record Importing 271
Using the ClearQuest Import Tool 271
Importing Duplicate Records 275
Importing Duplicate Records Separately 275
Recovering from Import Errors. 275
Updating Existing Records. 276

A. ClearQuest Schemas and Packages 277

ClearQuest Predefined Schemas 278
ClearQuest Packages. 279
State Model for the Defect Record Type 288
State Model for the EnhancementRequest Record Type 289

State-Type Models for Packages 291
Resolution Package State Type Model 291
UnifiedChangeManagement Package State Type Model 292

B. Adding ClearQuest Integrations 295

Overview of ClearQuest Integrations 295
Independent Integrations 296
Dependent Integrations 296
ClearQuest Integrations and Code Pages 297
Example 297
Enabling Record Types for Integrations 298
Viewing the Packages in Your Schema 299
Testing Integrations 300
Adding Independent Integrations 300
Adding Dependent Integrations 301
Adding a Rational Administrator Integration 302
Adding the Repository Package 302
Saving the Schema Changes 303
Configuring Rational Administrator 303
Adding a Rational ClearQuest Project Tracker Integration 303
Adding the AMBaseActivity Package 303
Adding the AMWorkActivitySchedule Package 304
Saving the Schema Changes 304
Configuring Rational ClearQuest Project Tracker 304
Adding a Rational TeamTest Integration 304
Adding the Repository Package 305
Adding the TeamTest Package 305
Saving the Schema Changes 305
Configuring Rational TeamTest 305
Adding a Rational UCM Integration 305
Adding the AMStateTypes Package 306
Setting the Default Actions for UCM 308
Adding the UCMPolicyScripts Package 309
Adding the UnifiedChangeManagement Package 309
Adding the BaseCMActivity Package (Optional) 309
Saving the Schema Changes 309
Configuring Rational UCM 310
Adding a Microsoft Visual SourceSafe Integration 310
Adding the Visual SourceSafe Package 310
Saving the Schema Changes 310

Creating a Query in ClearQuest Client 311
Setting Up Each Client Machine 311
Applying Package Upgrades 312
Viewing the Package Upgrade Status 313

C. Hook Examples 315

Field Hook Examples 316
Field Choice List Hook Example 316
Hook for Creating a Dependent List 317
Field Choice List Hook to Display User Information 318
Field Default Value Hook Examples 320
Example 1 320
Example 2 321
Field Permission Hook Example 322
Field Validation Hook Example 323
Field Value Changed Hook Example 324
Action Hook Examples 325
Action Initialization Hook Example 325
Action Initialization Hook for a Field Value 326
Action Hook for Setting the Value of a Parent Record 327
Action Access Control Hook Example 331
Action Commit Hook Example 332
Action Notification Hook Example 334
Action Validation Hook Example 337
Record Script Example 340
Global Script Example 341
Using CAL Methods in ClearQuest Hook Scripts Example 342
Advanced Reporting and Automation with cqperl 345

D. Form Controls 349

ActiveX Control 349
Attachment Control 350
Check-Box Control 351
Combo-Box Control 352
Drop-Down List Box Control 353
Drop-Down Combo Box Control 355
Duplicate Base Control 356
Duplicate Dependents Control 357

Group Box Control 358
History Control 358
List Box Control 359
List View Control 360
Option Button Control 361
Parent/Child Control 362
Picture Control 364
Push Button Control 364
Static Text Box Control 366
Text Box Control 366

E. Command Line Utilities 369

The dbset Parameter in Command Line Utilities 369
Using installutil 369
Creating Database Copies 371
installutil convertschemarepo 371
installutil convertuserdb 372
Unlocking Databases 373
Examples 374
Oracle 374
SQL Server 375
Copying a Schema with cqload 376
Preparing to Use cqload 376
Copying a Schema into a Schema Repository 376
Copying a Partial Schema into an Existing Schema 376
exportschema 376
Syntax 376
Example 377
importschema 377
Syntax 377
Example 378
exportintegration 378
Syntax 378
Example 379
importintegration 379
Syntax 380
Example 380
Importing and Exporting Dynamic Lists with importutil 381

Exporting the Dynamic List 381
Syntax 381
Example 381
Importing the Dynamic List 382
Syntax 382
Example 382
Moving Workspace Items Between User Databases with bkt_tool 382
bkt_tool Command Line Options 383
Export 384
Syntax 384
Example 384
Import 384
Syntax 384
Example 385
Update 385
Syntax 385
Example 385
Executing Nightly Reports Using cqtool on UNIX 385

F. Database and Troubleshooting Guide 389

Oracle Connection Issues on UNIX 389
Database Connection Options 389
Unknown Host Errors 390
Program Unavailable Errors 391
Oracle Not Available Errors 391
Diagnostic Procedures 392
Pinging a Host 392
Verifying ClearQuest Database Settings 393
Verifying That the OpenLink Request Broker Is Running 394
Verifying Oracle Connectivity 395
Other Oracle Issues 395
Searches with the Contains Operator Are Always Case-Sensitive 395
Debugging e-mail Notification Issues with dbwin32 395

Index 397

Figures

Figure 1 A Change Management Process Model 4
Figure 2 A Schema Repository with Multiple User Databases 7
Figure 3 Sharing Common Data with REFERENCE_LIST Fields 109
Figure 4 Example of Parent/Child Hierarchy 111
Figure 5 State Model for the Enhancement Record Type 114
Figure 6 Elements of a Security System 183
Figure 7 State Model for Defect Record Type 289
Figure 8 State Model for Enhancement Record Type 290
Figure 9 State Type Model for Resolution Package. 292
Figure 10 State Type Model for Enhancement Record Type 294

Tables

Table 1 Roles in the Change Request Lifecycle 2
Table 2 ClearQuest Architecture for Windows Installations 9
Table 3 ClearQuest Architecture for UNIX Installations 12
Table 4 ClearQuest Administrator's Tasks 14
Table 5 Predefined Schemas 18
Table 6 Keyboard Short Cuts 24
Table 7 Mouse Actions 25
Table 8 Code Page Related Administrator Tasks 28
Table 9 Guidelines for Selecting a ClearQuest Data Code Page Value 30
Table 10 Rational Supported Windows Code Pages 31
Table 11 Database Vendor Character Sets and Windows Code Pages 34
Table 12 ClearQuest Database Activities 44
Table 13 Oracle Character Sets and Windows Code Pages 49
Table 14 Overview of ClearQuest Packages 132
Table 15 Sample Data by Vendor Type 370

Preface

Rational ClearQuest is a customizable defect and change tracking system. It helps development teams organize and automate the processes involved in submitting, assigning, tracking, testing and releasing change requests. These change requests can include the correction of defects, the enhancement of existing functions, the addition of new features, and the modification of documentation and packaging.

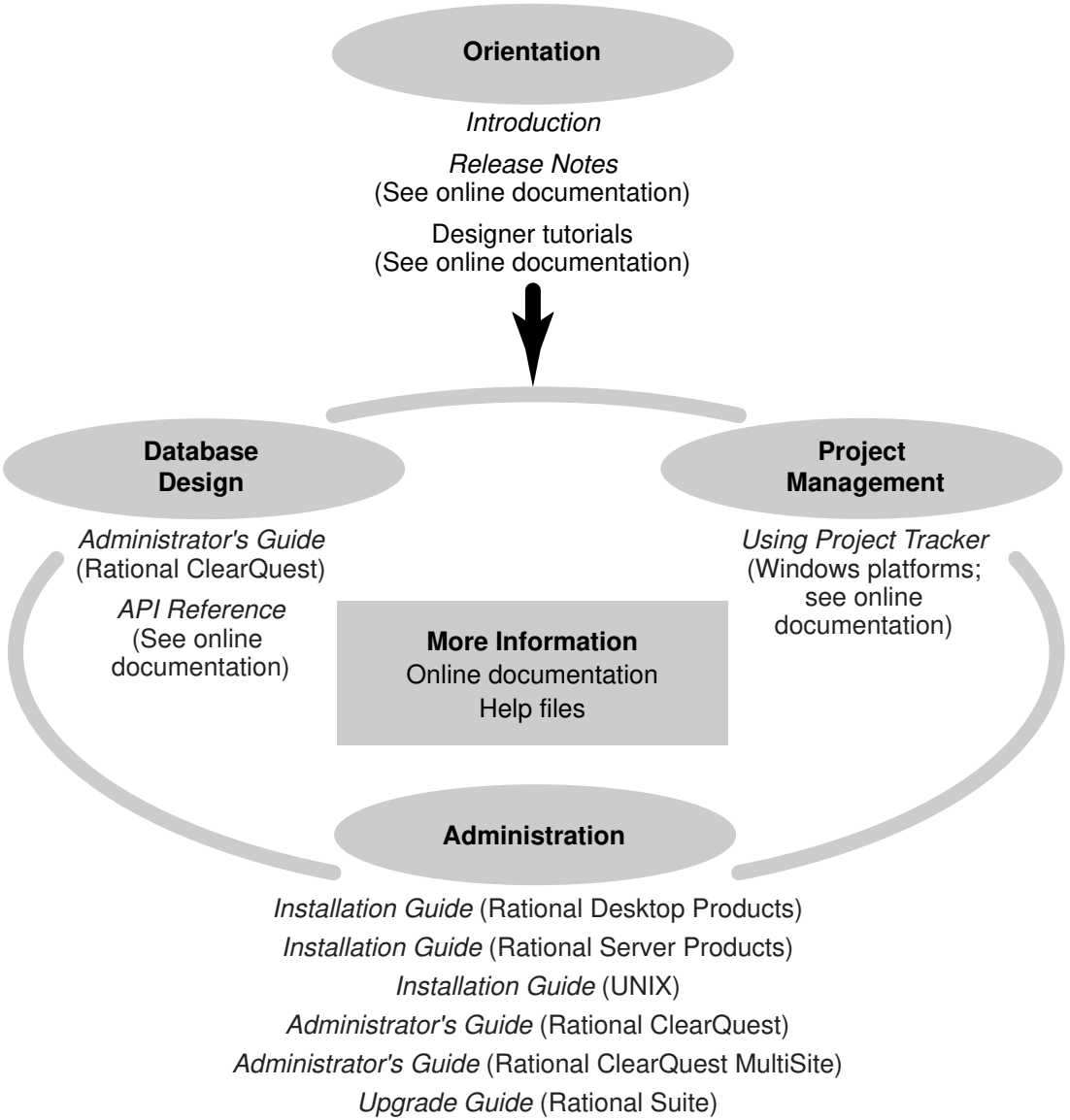
About This Manual

This guide is for ClearQuest administrators; that is, people responsible for setting up the ClearQuest environment, creating the process models for managing change requests, adding and managing users, and administering the ClearQuest environment on an ongoing basis.

The topics covered in this guide include the concepts of change management, the functions of the ClearQuest administrator, managing ClearQuest databases, creating and modifying ClearQuest schemas, administering users on security, utilitizing ClearQuest tools and utilities and implementing integrations between ClearQuest and other Rational software applications.

This guide assumes that you are familar with the ClearQuest Client, have experience administering relational databases, and know how to write scripts in VBScript or Perl.

ClearQuest Documentation Roadmap



ClearQuest Integrations with Other Rational Products

Integration	Description	Where it is documented
ClearQuest-Base ClearCase	Associates change requests with versions of ClearCase elements.	ClearCase: *Developing Software* ClearCase: *Managing Software Projects* ClearQuest: *Administrator's Guide*
ClearQuest-PurifyPlus	Allows developers to invoke ClearQuest from PurifyPlus.	PurifyPlus Help ClearQuest: *Administrator's Guide*
ClearQuest-RequisitePro	Allows developers to invoke ClearQuest from RequisitePro and to associate requirements with ClearQuest change requests.	Rational Suite: *Administrator's Guide* ClearQuest: *Administrator's Guide*
ClearQuest-SoDA	Collects information from ClearQuest and presents it in various report formats.	*Using Rational SoDA for Word* *Using Rational SoDA for Frame* SoDA Help
ClearQuest-Test Manager	Allows developers to invoke ClearQuest from TestManager.	*Using Rational TestManager* ClearQuest: *Administrator's Guide*
ClearQuest-Robot	Allows developers to invoke ClearQuest from Robot.	Rational Robot User's Guide Rational Robot Help
ClearQuest-UCM	Links UCM activities to ClearQuest records.	ClearCase: *Developing Software* ClearCase: *Managing Software Projects* ClearQuest: *Administrator's Guide*

Typographical Conventions

This manual uses the following typographical conventions:

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

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

  lsc· heckout

Customer Support

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

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

Understanding ClearQuest Administration

1

Rational ClearQuest is a customizable defect and change tracking system that allows you to implement a comprehensive change management system which will help your development teams deliver better quality products, faster, and with less wasted effort.

Before you start deploying ClearQuest, it is helpful to have an overview of the concepts of change management and the functions of the ClearQuest administrator. The topics covered include:

- *Understanding the Concepts of Change Management* on page 1.
- *Defining ClearQuest Schemas, Schema Repositories, Databases, and Connections* on page 5.
- *Understanding the Architecture and Components of ClearQuest* on page 8.
- *Getting Started with ClearQuest* on page 22.

If you are familiar with the concepts of change management, you can skip the Understanding the Concepts of Change Management section.

Understanding the Concepts of Change Management

Why Change Management?

Change management systems organize and automate the processes involved in submitting, assigning, tracking, testing, and releasing change requests. The changes tracked by these systems can include the correction of bugs and defects, the enhancement of existing functions, the addition of new features, and the modification of documentation or packaging.

A change management system helps development teams:

- Avoid duplication of effort.
- Track the progress of individual change requests.
- Prioritize change requests and address the most important first.
- Balance the workload among the members of the development team.
- Improve teamwork and resolve differences among team members.
- Monitor the progress and status of a project as a whole.
- Understand and improve the product development process.

The Change Request Lifecycle

The concept of the change request lifecycle is fundamental to change management systems. A change request is submitted by a member of the development team or by an outside party (such as a customer, a product manager, or a customer support representative). A member of the development team reviews the request, which is then assigned to a person who can perform the work to implement the change. The result is then passed on to additional people who review and test the work, then integrate it into the rest of the project. When the change is complete, tested, and integrated, it is considered closed.

Table 1 shows how this lifecycle might work in a typical product development organization.

Table 1 Roles in the Change Request Lifecycle

Team Member	Role in the Change Request Lifecycle
User	Submits a change request.
Change Control Manager	Reviews and accepts the change request.
Project Manager	Assigns and schedules work.
Engineer	Performs the work to resolve the change request.
Tester	Verifies the solution in a test build.

Of course, the lifecycle of a change request can be more complicated than the one shown in Table 1. There may be more or fewer parties involved, and the change request might be rejected, revised, reassigned, postponed, duplicated, and otherwise modified or rerouted.

The Change Management Process Model

A change management process model (also called a state transition model) is a systematic representation of the possible steps in one change request lifecycle. This model defines the lifecycle in terms of the states that the change request passes through, the actions that are taken to move between the states, and the rules that define when and how the actions can be taken.

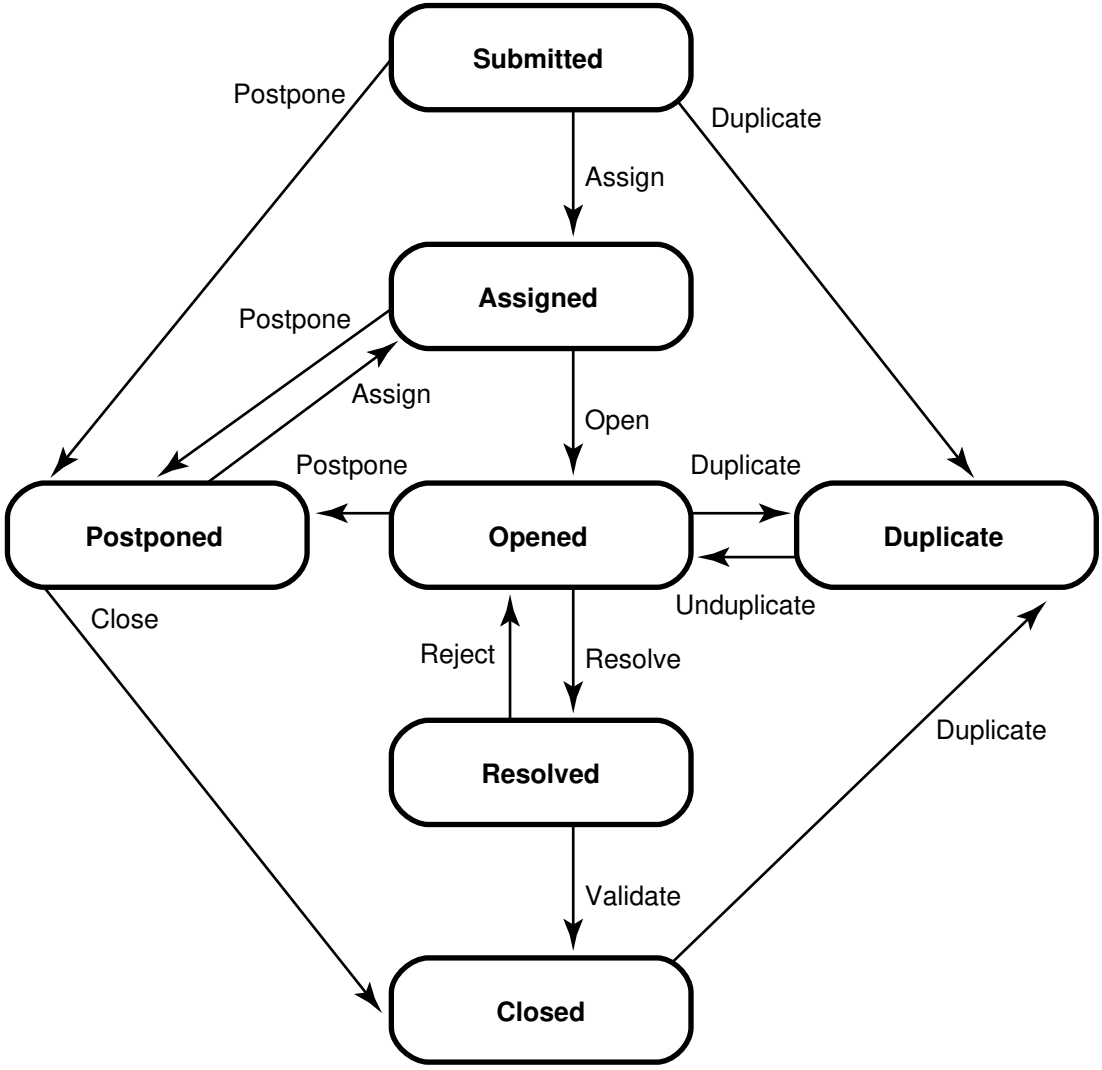
The state of a change request is its current status. Typical states include submitted, assigned, opened, postponed, duplicated, resolved, and closed.

An action is an activity that moves a change request from one state to another (a state transition). Typical actions include assign, reject, open, postpone, duplicate, validate, resolve, and close.

Rules define when and how an action can be taken. For example, the ability to use a validate action to move a change request from a *resolved* state to a *closed* state might be restricted to quality assurance engineers.

A process model can be illustrated in a diagram, as shown in Figure 1. In this diagram states are shown as rounded rectangles and actions are represented by arrows.

Figure 1 A Change Management Process Model



Process models may differ between change request types (that is, the process for correcting defects may have differences from the process for adding new features). They usually need to be customized to address the needs and practices of each development team and each project.

ClearQuest: A Defect and Change Tracking System for Change Management

ClearQuest is an automated defect and change tracking systems that allows you to implement and manage consistent change management process models.

For each change request type, you can:

- Define a process model with states and actions.
- Program customized rules that control when and how actions can be performed.
- Create forms that allow participants in the system to access change requests, communicate with each other, and perform appropriate actions.
- Build reports and queries that give managers visibility into the status of individual change requests, classes of change requests, and all of the change requests associated with a project.

ClearQuest ensures that individual change requests go through a change request lifecycle according to the established policies and procedures of the organization, while simplifying the work of the members of the development team.

ClearQuest also provides management tools to assign work and manage workloads, track the progress of projects, improve teamwork, and continuously improve processes and products.

ClearQuest can also be implemented as an integrated part of the Rational Suite. Information about change requests can be transmitted to other Rational products used for development and testing. Information from those tools can be accepted back into ClearQuest, to allow the most complete tracking of the status of change requests.

Defining ClearQuest Schemas, Schema Repositories, Databases, and Connections

Schemas

A schema in ClearQuest is a complete description of the process model for one type of change request (the metadata for that process model). This includes a description of the states and actions of the model, the structure of the data that can be stored about the individual change requests, hook code or scripts that can be used to implement business rules, and the forms and reports used to view and input information about the change requests.

The schema is a pattern or blueprint for ClearQuest user databases. When you create a database to hold actual user data, the database will follow the blueprint defined in a schema. However, a schema is not a database itself: it does not hold any user data about

actual change requests, and it does not change when users add or modify data in the user databases.

You, as the administrator, can create and modify schemas using Rational ClearQuest Designer. Other members of the development team normally cannot create or modify schemas. However, you can delegate rights to make specific changes, such as modifying queries and reports

Schema Repositories

ClearQuest stores schemas in a special type of database called a schema repository, which is also sometimes referred to as the master repository or the master database.

A schema repository can store multiple schemas, for example, one schema for defect change requests and another schema for feature enhancement change requests.

The schema repository can also include multiple versions of the same schema. A new version is created each time changes are made to a schema, for example, by changing an action or adding a new report.

Databases

A database in ClearQuest is a collection of change request data for one process model.

You associate each database with a specific version of a specific schema (see Figure 2). The schema defines the way the data is stored and changed in that database (that is, the database is an *instance* of a version of a schema). Users change data in the databases when they add or modify information about change requests.

Databases contain a record for each change request. As the change request moves through its lifecycle the data stored in this record changes accordingly.

You can create and associate multiple databases with a single schema. For example, if you have three projects that use the same process model for correcting defects, you could create one database for each project and associate all three projects with the same schema.

ClearQuest provides three types of databases:

- A user database, also referred to as a production database, manages information entered by members of the development team. These contain the *production data* created in the course of working on actual change requests.
- A *sample database* is a working user database that provides 40 sample defect records of user information, as well as sample queries and reports. You can use the sample user database to familiarize yourself with how ClearQuest works, and to help train users. The sample user database works with all of the predefined

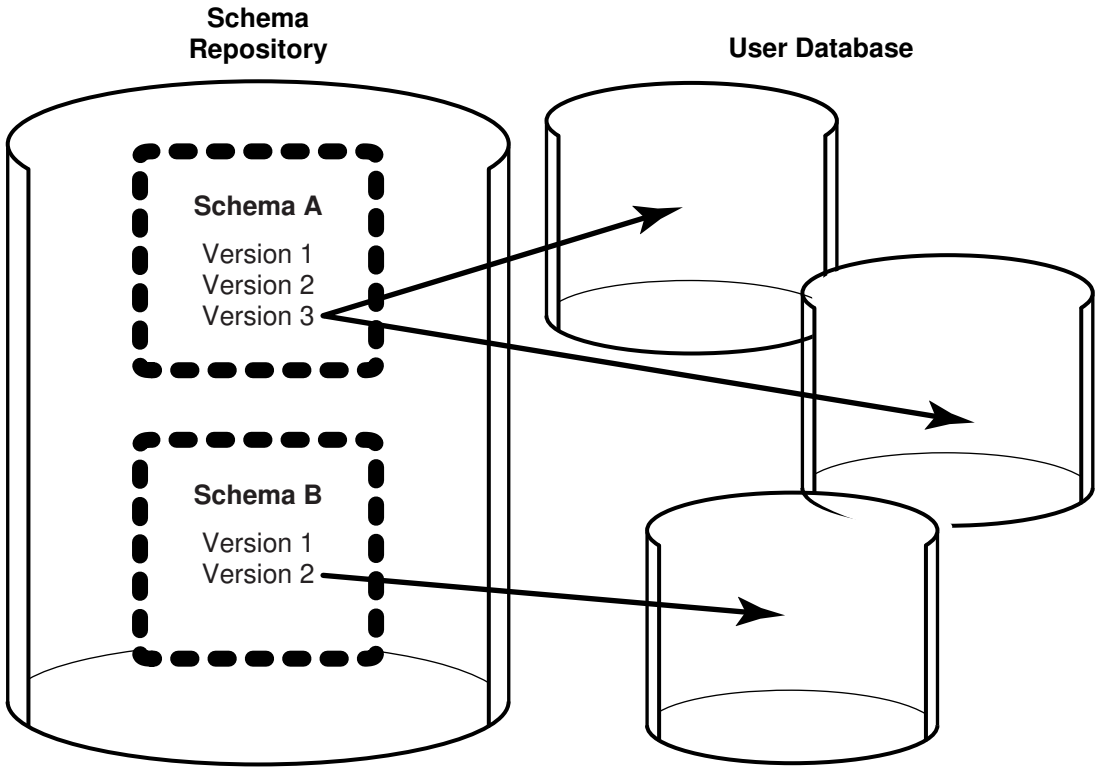
schemas that come with ClearQuest except for the Blank and Common schemas. You can create a sample user database when you create a schema repository with the ClearQuest Maintenance Tool. For more information about predefined schemas, see *ClearQuest Predefined Schemas* on page 278.

- A test database is a temporary database that you can create when you are using the Rational ClearQuest Designer to develop a new schema or create a new version of an existing schema. After you create a test database, you use the Rational ClearQuest client to add and view data in the test database to make sure that the new schema performs as expected. The test database allows you to experiment without running the risk of damaging production data in a user database.

Connections

A connection is a database set; it consists of one schema repository and all of its associated user databases, as illustrated in Figure 2.

Figure 2 A Schema Repository with Multiple User Databases



Typically, one connection will contain all of the schemas and databases for one project. However, you can have one connection support multiple projects, or you can set up multiple connections for one project.

Vendor Database Management Systems

The schemas, schema repositories, databases, and connections used in ClearQuest are stored and managed by one or more database management systems.

Sybase SQL Anywhere is shipped with ClearQuest, and is typically used with small or medium-sized installations. Other databases that can be used with ClearQuest include Microsoft Access, typically used with small installations; and Microsoft SQL Server, Oracle DBMS, or IBM DB2, which are suitable for larger and more distributed environments.

Understanding the Architecture and Components of ClearQuest

ClearQuest Architecture and Components for Windows Installations

The ClearQuest architecture and components differ somewhat between a Microsoft Windows installation and a UNIX installation. We will discuss the Windows installation first, and then discuss the differences in the UNIX installation. Note that some of the features of the Windows installation are required for both versions, so if you are going to use the UNIX installation, you should read both sections.

The ClearQuest architectural layout for Windows is available in Table 2.

Table 2 ClearQuest Architecture for Windows Installations

Layers	Components
Tier 1	▪ ClearQuest client ▪ ClearQuest Web ▪ ClearQuest Designer ▪ ClearQuest Maintenance Tool ▪ ClearQuest Import Tool ▪ ClearQuest Export Tool ▪ ClearQuest User Administration ▪ ClearQuest API ▪ ClearQuest MultiSite ▪ Command Utilities ▪ Rational E-Mail Reader
Tier 2	▪ ClearQuest core
Tier 3	▪ Microsoft Access ▪ SQL Anywhere ▪ Microsoft SQL Server ▪ Oracle RDBMS ▪ IBM DB2 ▪ ODBC Drivers

Starting at the bottom of Table 2:

- Tier 3, or the database layer, contains the database management system or systems that physically manage the information in the schemas and user databases. In the Windows environment, the database management system can be Microsoft Access (for small installations, typically five users or less), SQL Anywhere (for small or medium-sized installations) or Microsoft SQL Server, Oracle RDBMS, or IBM DB2 (for larger installations).
- Tier 2, or the ClearQuest core, is the layer of the architecture where application logic and business rules are stored and executed.

  The ClearQuest core communicates with the database layer (Tier 3) using Open Database Connectivity (ODBC) drivers such as the OpenLink Driver. For more information, refer to the *Installation Guide* for Rational Server Products.

The ClearQuest core also exposes limited amount of core functionality through the ClearQuest API (Application Programming Interface). These are typically used for hook scripting purposes.

- Tier 1 of the ClearQuest architecture includes a variety of modules and tools for entering and viewing data, creating and managing schema and databases, importing and exporting data from other environments and for replicating data across multiple locations. Tier 1 also provides the ClearQuest API (Application Programming Interface) for communication with the Tier 1 applications and tools.

If you have ClearQuest installed on your system, you can see the elements that make up Tier 1 by clicking **Start > Programs > Rational Software > Rational ClearQuest**.

The ClearQuest client, also referred to simply as Rational ClearQuest**,** is the part of Tier 1 used by you and by all members of the development team. It is a native module for Windows that works in client/server mode to allow users, managers, and administrators to enter and modify change requests, execute and view queries and reports, and communicate with each other.

The ClearQuest Designer, a client/server administrative application, is a tool you use to create and modify schemas and create and upgrade user databases. You use it to perform tasks such as defining the layout of the data in a schema and building a process model for each type of change request.

Tier 1 also includes several tools you can use to perform administrative functions, or that you can set up to provide additional capabilities to users of the development team. These include:

- ClearQuest Maintenance Tool, an administrative tool you use to create and manage schema repositories and connections. You can also use it to create sample databases and to upgrade schema repositories and databases to a new release level of ClearQuest.
- ClearQuest Export Tool, a utility you and ClearQuest users can use to export data from a ClearQuest database to an external file.
- ClearQuest Import Tool, a utility you and ClearQuest users can use to import data from an external file to a ClearQuest database.
- ClearQuest Web, a version of the ClearQuest client you can set up on a Web server that allows users with an Internet browser to perform many of the same functions available through the ClearQuest Client. ClearQuest Web uses Active Server Pages (ASP), and must be hosted on a Windows-based Web server running Microsoft Internet Information Server (IIS). It also requires that a Java Virtual Machine (JVM) be present on user's PCs or UNIX workstations.

- ClearQuest E-Mail, a feature that allows users to receive notification when specific events occur. For example, you could create a rule that whenever a defect is fixed, an e-mail notification be sent alerting the testing team. ClearQuest E-Mail also includes an application called the Rational E-Mail Reader that allows users to submit and modify change requests by e-mail messages, without using the ClearQuest client.
- ClearQuest User Administration, a tool you use to set up and administer users and user groups. You can perform functions like creating users and user groups, setting user IDs and passwords, assigning users to user groups, importing and exporting users and user groups among schema, and subscribing user groups to databases. The User Administration tool is also available within ClearQuest Designer.

Customizations: Hooks and the ClearQuest API

Most communication between users and administrators and the ClearQuest core can be handled using the standard features available in the Tier 1 modules. However, you may have a need to customize business rules and provide sophisticated handling of data. ClearQuest makes this possible by making provisions for *hooks* that can access the ClearQuest core through an Application Programming Interface (API).

Hooks are scripts or triggers that can run under a designated set of conditions to perform a specific task. For example, hooks can be used to:

- Control the data collected in a field by inserting default values in the field, creating a choice list with valid values, or checking an entry to make sure it is the correct data type or falls within a specified range.
- Define access controls to limit who can perform an action and when that action can be performed.
- After a specific action has been performed, *trigger* another event, such as sending an e-mail notification.

ClearQuest supports hooks written in VBScript or Perl in the Windows environment, and Perl only in the UNIX environment. When working with Windows, administrators must choose to use either VBScript or Perl exclusively from the beginning. ClearQuest cannot run hooks in mixed scripting language.

Hooks execute under ClearQuest's direction when a user performs an action on a record. Hooks access the ClearQuest core through the ClearQuest API, a robust Application Programming Interface that provides a set of objects, methods, and functions that can be called from hook code. For example, you can use hooks to set a list of allowed values within a field based on factors such as invoking a user's identify, time of day, or status of another database record.

External applications can also call the ClearQuest API, providing they support OLE automation or embedded Perl. For example, you can use the external applications to create and run queries, run charts and reports, and examine the contents in the Client Workspace.

For more information about hooks, external applications and the ClearQuest API, see the *API Reference* for Rational ClearQuest and click the Rational Developer Network at www.rational.com.

ClearQuest Architecture for UNIX Installations

The ClearQuest architecture for UNIX installations, shown in Table 3, has the same three layer structure as the architecture for Windows installations, but the components differ in number and function.

In addition, on Windows a ClearQuest installation is required to configure, customize, and administer schemas and databases using the ClearQuest Maintenance Tool and ClearQuest Designer.

The ClearQuest architectural layout for Windows is available in Table 2.

Table 3 ClearQuest Architecture for UNIX Installations

Layers	Components
Tier 1	▪ ClearQuest client ▪ ClearQuest Web ▪ ClearQuest API ▪ ClearQuest MultiSite ▪ Command Utilities ▪ ClearQuest administrative applications on a Windows installation
Tier 2	▪ ClearQuest core
Tier 3	▪ Microsoft Access ▪ SQL Anywhere ▪ Microsoft SQL Server ▪ Oracle RDBMS ▪ IBM DB2 ▪ ODBC Drivers

Starting at the bottom of Table 3:

- Tier 3, or the database layer, contains the database management system or systems that physically manage the information in the schemas and user databases. In the UNIX environment, the database management system can be Oracle RDBMS on a UNIX server, Microsoft SQL Server or IBM DB2 on a Windows server.
- Tier 2, or the ClearQuest core, is the layer of the architecture where application logic and business rules are stored and executed.

  The ClearQuest core communicates with the database layer (Tier 3) using Open Database Connectivity (ODBC) drivers such as the OpenLink Driver and OpenLink Request Broker. For more information, refer to the *Installation Guide* for Rational ClearCase Product Family.
- Tier 1 of the ClearQuest architecture includes tools that are used by users and administrators to enter and modify change requests, create reports, and communicate with each other, and for replicating data across multiple locations. Tier 1 also provides the ClearQuest API for communication with the Tier 1 applications and tools.

The ClearQuest client, also referred to simply as Rational ClearQuest, is the part of Tier 1 used by you and by all members of the development team. It is a native module for UNIX systems that allows users, managers, and administrators to enter and modify change requests, run and view queries and reports, and communicate with each other.

Browsers on a UNIX workstation can be used with ClearQuest Web, provided that a Java Virtual Machine (JVM) is present on the workstation and that ClearQuest Web and the Microsoft IIS web server are set up on a Windows server in the environment. Refer to the *Installation Guide* for Rational ClearQuest Product Family and the *Installation Guide* for Rational Server Products.

Tier 1 for a UNIX installation also includes the capability to create a distribution point on a UNIX server from which users can download and install the ClearQuest Client for UNIX.

The ClearQuest installation for UNIX does *not* include the ClearQuest Designer, ClearQuest Maintenance Tool, ClearQuest Import Tool, ClearQuest Export Tool, ClearQuest User Administration Tool, ClearQuest Web, or the Rational E-Mail Reader.

On Windows, a ClearQuest installation with the ClearQuest Designer, ClearQuest Maintenance Tool, ClearQuest User Administration and the Rational E-Mail Reader is required to configure, customize, and administer schemas and databases within the UNIX environment.

Similarly, on Windows a ClearQuest installation with the ClearQuest Import Tool and ClearQuest Export Tool is needed to import and export data from external files.

Understanding the Functions of the ClearQuest Administrator

As a ClearQuest administrator, you will typically be responsible for installing ClearQuest, creating a release area and installing service releases, defining and managing schemas (this is, designing and modifying process models), creating and managing user databases, enabling Web and e-mail access, creating and maintaining *public* queries, charts and reports, and managing user accounts.

These functions are described in this section. The descriptions include pointers to sections of this document and to other ClearQuest documents that provide detailed information on each function.

A complete list of the administrator's tasks is in Table 4.

Table 4 ClearQuest Administrator's Tasks

Administrator's task	Application
Design the change management process.	
Install and configure the RDBMS.	Vendor Database
Install ClearQuest.	Rational ClearQuest Setup Wizard
Create a schema repository and connection.	ClearQuest Maintenance Tool
Create user databases.	ClearQuest Designer
Document process workflows.	
Select predefined schemas that already fit the workflows. And/Or Modify predefined schemas to fit the workflows. And/Or Create new schemas to fit the workflows.	ClearQuest Designer
Determine appropriate code page settings and perform administrative tasks to ensure data integrity.	Vendor database tools, ClearQuest installutil command utility
Upgrade user databases to new schemas.	ClearQuest Designer
Enable Web access and e-mail notification (optional).	ClearQuest Designer, Rational E-Mail Reader

Table 4 ClearQuest Administrator's Tasks

Administrator's task	Application
Create public queries, charts, and reports.	ClearQuest, Rational SoDA and/or Seagate's Crystal Reports
Set security controls.	ClearQuest Designer
Delegate administrative functions.	
Manage user accounts.	ClearQuest User Administration
Ongoing maintenance, such as customize and manage schemas.	ClearQuest Designer
Install and set up ClearQuest MultiSite.	Rational ClearQuest Setup Wizard, ClearQuest MultiSite Administration Tools, Rational Shipping Server

These tasks are described in the sections that follow.

Designing and Implementing the System

Describing all of the steps required to design and implement a change request management system is beyond the scope of this guide. However, you may need to invest the time to create a comprehensive implementation plan. A significant effort is typically required to analyze existing processes (since these are rarely documented) and to achieve consensus on how they should work in the future.

To design and implement a change request management process you must:

- Identify the scope of the process.
- Determine the roles of the different types of users.
- Construct the change management process model.
- Obtain sign-off on the model from users and management.
- Train users and deploy the system.
- Enforce the change management process.

A team may be jointly responsible for many of these tasks, but as the ClearQuest Administrator you must make sure that none are overlooked.

Configuring the Database Management System

Before installing the ClearQuest software, you should install and configure the third-party database management system that you want to use to store the ClearQuest schemas, schema repositories, and databases.

To review recommendations for configuring third-party vendor databases, see *Installation Guide* for Rational Server Products for Windows and *Installation Guide* for Rational ClearQuest Product Family for UNIX.

Installing ClearQuest

Set up the license server and get the license key for ClearQuest, then install the ClearQuest software on client and server systems as required to support the users.

For Windows installations, refer to the *Installation Guide* for Rational Desktop Products and the *Installation Guide* for Rational Server Products for Windows. The latter document also specifies the platform and software requirements for Rational ClearQuest. For UNIX, refer to *Installation Guide* for Rational ClearQuest Product Family.

Subscribing to the ClearQuest User Group

In addition, after installing ClearQuest you may join the ClearQuest User Group. The ClearQuest User Group is an e-mail forum where you can share your experiences, pose questions, or obtain useful information from other ClearQuest users. To subscribe to the group, visit the Rational Web site at: http://www.rational.com/support/usergroups/

Your e-mail address will not be given out to anyone.

Creating and Managing Schema Repositories and Connections

Use the ClearQuest Maintenance Tool to create and manage schema repositories (also referred to as the master repositories or master databases).

When you create a new schema repository, you also create a new connection (or database set) that keeps track of all of the databases associated with schemas in that schema repository. You then use the ClearQuest Maintenance Tool to manage these connections.

For more information, see *Managing Databases* on page 43.

Performing Code Page Tasks

When you use ClearQuest, each client that accesses your ClearQuest databases has its own operating system code page. A code page specifies the set of characters that are

valid in a given environment. In other words, a code page controls which characters will be manipulated correctly on a particular client.

In a multilanguage environment, ClearQuest clients may run different language-based operating systems with different operating system code pages. In this type of environment, only characters common to all the operating system code pages must be entered in ClearQuest. ClearQuest has specific features to help a ClearQuest administrator protect these multilanguage environments and prevent users from entering characters that do not display correctly on clients.

These features include a ClearQuest data code page value for schema repositories and user databases, and a CharacterSetValidation package for installations that have clients running older versions of ClearQuest. A ClearQuest administrator must perform special code page tasks during the ClearQuest deployment to take advantage of these features.

For more information about these tasks, see Chapter 2, *ClearQuest and Code Pages*.

Creating a User Database

Use the ClearQuest Designer to create a user database and associate it with a predefined schema.

For more information, see Chapter 3, *Managing Databases*.

Creating the Schema (the Process Model)

First, document the process model or models that are going to be captured in the schema. This might include:

- Descriptions of the types of users and their roles.
- A diagram of the states and actions used to model the process.
- Descriptions of the rules controlling the actions.
- Sketches of forms and reports for viewing and submitting data, or lists of fields to include in these forms and reports.

After the requirements are known, use the ClearQuest Designer tool to create a new schema or to modify an existing schema to build the desired process model.

To help with this task, ClearQuest provides eight predefined, or out-of-the-box, schemas (Table 5).

Table 5 Predefined Schemas

Schema	Description
AnalystStudio	A schema with fields and rules that allow ClearQuest to be used with Rational AnalystStudio. Contains customizations for use with Rational RequisitePro.
Blank	Contains only system fields. A minimum configuration base for building schemas.
Common	A base-level configuration with the most commonly used schema elements.
DefectTracking	A schema with features for defect-tracking processes.
DevelopmentStudio	A schema with fields and rules that allow ClearQuest to be used with Rational DevelopmentStudio. Also contains fields and rules that work with Rational Purify, Quantify, and PureCoverage.
Enterprise	A schema with fields and rules that allow ClearQuest to be used with Rational Suite Enterprise. Contains fields and hooks that work with all Rational products.
TestStudio	A schema with fields and rules that allow ClearQuest to be used with Rational TestStudio. Also contains fields and rules that work with Rational TeamTest, RequisitePro, Purify, Quantify, and PureCoverage.
UnifiedChangeManagement	A schema with fields and rules that allow ClearQuest to be used with Rational UCM.

To create a schema that models the desired process flow, you can use ClearQuest Designer to:

- Create a new schema by starting with the Blank schema and add fields to the record for change requests, create a State Transition Matrix that defines the actions that move change requests between states, and creating forms, queries, and reports to view and modify data.

- Create a new schema from an existing schema (either a schema you created previously or the Common schema or Defect Tracking schema included with ClearQuest), and modify the new schema as required.
- Modify an existing schema.
- Add packages to an existing schema. A ClearQuest package is a set of schema components (record types, fields, tabs, forms, scripts, and queries) that can be added in one step to an existing schema to add a complex feature or function. Examples of capabilities added through packages include sending e-mail notifications, documenting the history of a change request, and adding attachments to a record. The packages available can be accessed from the Package menu of ClearQuest Designer.
- Add hooks to provide custom processing.
- Add integrations with other Rational software products. Special packages are available to integrate ClearQuest with other Rational products, including Rational ClearCase, Unified Change Management, ClearQuest Project Tracker, Rational Suite or third party integrations, such as Microsoft Visual Source Safe.

For more information, see Chapter 4, *Working with ClearQuest Schemas*, Chapter 5, *Customizing a Schema*, Chapter 6, *Applying Packages*, Chapter 7, *Working with Forms*, Chapter 10, *Using Hooks to Customize Your Workflow*, and Appendix A, *ClearQuest Schemas and Packages*.

Managing User Databases

When you create or modify a schema, use the ClearQuest Designer to create or upgrade user databases to work with the latest version of the schema.

For more information, see Chapter 8, *Administering Users*.

Enabling E-Mail Notification Through the Rational E-Mail Reader

You have the option of setting up the Rational E-Mail Reader to allow users to submit and modify change requests by e-mail, without using the ClearQuest client.

To do this you set up at least one dedicated e-mail account. Users can send e-mail messages to this e-mail account when they want to submit or modify a change request. The e-mail message must follow a set format so that the data in it can be interpreted by the Rational E-Mail Reader that runs as a Windows Service on the database server. You may want to create and distribute template e-mail messages that follow the set format, to assist users in submitting e-mail messages that can be parsed and processed by the Rational E-Mail Reader.

You also have the option of creating rules that cause ClearQuest users to be notified by e-mail when a specified event occurs.

To do this you apply the E-mail package to all schemas that you want to be able to accept information through e-mail. This package creates a stateless record type called Email_Rule. You can then create a set of rules that will trigger an e-mail notification when a specific event occurs, such as the transition of a change request to a new state.

This requires setting configurations in ClearQuest and creating an account on an e-mail server.

For more information, see Chapter 12, *Administering ClearQuest E-Mail*.

Enabling Web Access

You have the option of setting up ClearQuest Web to allow users with an Internet browser to perform many of the same functions avaialable through the ClearQuest client. To do this, you set up ClearQuest Web and the Microsoft IIS web server software on a web server. For more information, see Chapter 11, *Administering ClearQuest Web* and the *Installation Guide* for Rational Server Products for Windows.

Importing and Exporting Data

You have the option of importing data from other data sources using the ClearQuest Import Tool and exporting data to other data sources using the ClearQuest Export Tool.

For more information, see Chapter 13, *Importing and Exporting Data*.

Creating Queries, Charts, and Reports

You have the option of creating or modifying queries, charts, and reports using the ClearQuest Designer. These queries, charts, and reports can be stored in the Public Queries folder to be made accessible to all ClearQuest users. Alternatively, you have the option of delegating these tasks to other users such as project managers.

Note: To create reports, you must acquire a license for a reporting tool, such as Rational's SoDA software or Seagate's Crystal Reports.

For more information, see ClearQuest Client Help and Chapter 8, *Administering Users*.

Managing User Accounts

Manage user accounts by using ClearQuest Designer to create user accounts, subscribe users to databases, create user groups, and subscribe them to databases.

For more information, see Chapter 8, *Administering Users*.

Setting Security Controls on Records

One of the most powerful capabilities related to managing user accounts is the ability to set security controls on records so that you can control exactly which user groups get to see which information. For example, you may want to allow customers to submit and view change requests, but without being able to see sensitive information about change requests submitted by other customers or by members of your development team.

To achieve this type of security, you can make records visible only to specific user groups. You can also restrict access to fields on a record's form tab to specific user groups.

You set security controls on records by creating a security context field in the record type, and then assigning user groups permission to see records that have a certain value in this field.

For more information, see Chapter 9, *Using Security in Rational ClearQuest*.

Delegating Administrative Functions

As the ClearQuest Administrator, you can set yourself up to have super user privileges that allow you to perform all of the administrative functions available through ClearQuest tools.

However, to keep your workload manageable and avoid becoming a bottleneck for your development team, you can use the ClearQuest Designer to delegate some administrative functions to others. You can set up other super users or create other types of users and administrators with more limited permissions.

- A Super User has all ClearQuest permissions, and is the only type of user who can create and delete schemas and user databases.
- An Active User is the typical development team member who can log on to ClearQuest, change his or her own password, name, e-mail address and phone number, and view (but not change) schema and database information.
- A user with the All Users/Groups Visible permission can view (but not change) information about other users and groups.
- A Schema Designer can change schemas and upgrade databases (but cannot create or delete databases or edit other users' information).
- A User Administrator can add and edit user and group information, and can grant and revoke user permissions.
- A Public Folder Administrator can control the content of the Public Queries folders; for example, creating and modifying queries.

- A Dynamic Choice List Administrator has full control of the content of dynamic choice lists that appear on forms.
- A SQL Editor can edit the SQL code generated by SQL queries.
- A Security Administrator can determine which user groups have access to which record types. A Security Administrator also has the rights of an SQL Editor.

For more information, see Chapter 8, *Administering Users*.

Setting Up ClearQuest MultiSite

You have the option of installing and configuring ClearQuest MultiSite. ClearQuest MultiSite automatically replicates data across multiple locations so that developers in different cities and countries can work on the same project.

For more information, see the *Administrator's Guide* for Rational ClearQuest MultiSite.

Getting Started with ClearQuest

To start any of the administrative applications in ClearQuest, such as the Designer, Maintenance Tool, User Administration Tool, Import Tool and Export Tool, click **Start > Programs > Rational Software > Rational ClearQuest**.

If you have just installed ClearQuest, you can log on by entering *admin* as the user name, leaving the password field blank, and clicking **OK**.

Setting a Password for Your Administrative Account

When you first install ClearQuest, you are provided with a default admin account with Super User privileges that you can use to access all of the ClearQuest administrative tools, including ClearQuest Designer, the Maintenance Tool, Import and Export Tools. However, this default admin account allows a logon without requiring a password.

For security reasons, it is recommended that you use the User Administration Tool to set a password for this default admin account.

For more information, see *Editing Users* on page 169.

Toolbars, Palettes, Menus, and Shortcuts

Most of the ClearQuest administrative tools, other than ClearQuest Designer, display their primary tool icons and controls in a single window.

ClearQuest Designer, however, provides a wide range of features that are accessed through toolbars, palettes, dynamic menus, and keyboard shortcuts that are not always

visible. Before you begin performing administrative activities, you should become familiar with the features used to navigate and perform actions in ClearQuest Designer.

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 pulldown 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

To display a shortcut menu for the clicked item, right-click a cell in a grid, a form control, an error message in the Validation Output window, or an item in the Workspace. 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.

Using Keyboard Shortcut and Mouse Actions

ClearQuest Designer includes the following keyboard shortcuts (Table 6) and mouse actions (Table 7).

Table 6 Keyboard Short Cuts

Keys	Results
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.
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.
Form Layout Menu	
F7	Sets the size of a control based on its content.
CTRL + Left arrow	Aligns the left side of selected controls.

Table 6 Keyboard Short Cuts

Keys	Results
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 + 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 nonadjacent items, such as cells in a grid.

Table 7 Mouse Actions

Mouse action	
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.

Table 7 Mouse Actions

Mouse action	
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.

ClearQuest and Code Pages

2

Each client that accesses your Rational ClearQuest databases has its own operating system code page that specifies a valid set of characters for that environment. In other words, a code page controls which characters are displayed correctly on a particular client.

In a multi-language ClearQuest environment, clients may run various language-based operating systems with different operating system code pages. In this type of environment, only characters common to all operating system code pages can be entered in ClearQuest.

Important note: On Windows, ClearQuest supports and is tested and certified to run a variety of different code pages and languages (see Table 10 on page 31). On UNIX, ClearQuest supports only ASCII, or printable English characters. If you run UNIX clients, you must plan appropriately for your ClearQuest deployment.

In versions of ClearQuest prior to v2002.05.01 Patch 02, if you entered characters from different code pages into a database, several problems could occur, including data display problems, data corruption, and MultiSite synchronization failures. This version of ClearQuest (v2003.06.00) has several new features to ensure that only data that can be appropriately handled by the database can be entered. The sections in this chapter explain these features and the procedures you need to follow.

Note: Replication of data from more than one code page causes MultiSite synchronization failure. Therefore it is *mandatory* for all ClearQuest MultiSite customers to choose a valid, tested, and supported code page for their deployment (see Table 10 on page 31).

Code Page Task Roadmap

Table 8 lists the tasks an administrator must perform to enforce a single code page in a ClearQuest environment.

Table 8 Code Page Related Administrator Tasks

Administrator task	For more information
Determine the appropriate data code page value for a schema repository.	See *Selecting a ClearQuest Data Code Page* on page 29.
Set the character set of all vendor databases.	See *About the Vendor Database Character Set* on page 32.
Set the data code page value for each schema repository.	See *Setting the ClearQuest Data Code Page Value* on page 35.
If your ClearQuest environment includes clients running versions of ClearQuest earlier than v2003.06.00, apply the CharacterSetValidation package and copy the data code page value to all user databases.	See *Interoperation with Previous Versions of ClearQuest* on page 38.

About the ClearQuest Data Code Page

ClearQuest has a setting called the "ClearQuest data code page" that is specified for each schema repository. All user databases associated with a schema repository use the same ClearQuest data code page value. This value enforces a single code page for the database set and prevents characters not in the selected code page from entering the databases.

ClearQuest Data Code Page and New Schema Repositories

After you create a new ClearQuest schema repository, you must set the ClearQuest data code page value to an appropriate value (see *Selecting a ClearQuest Data Code Page* on page 29). If you do not set the data code page value, the default value of ASCII is used and your databases are limited to ASCII (printable English characters) data entry only.

Reminder: On UNIX, ClearQuest supports *only* ASCII. Customers with UNIX clients must opt to either set the ClearQuest data code page to ASCII (recommended) or require users on UNIX systems to enter data only with the Web client.

However, if you use non-ASCII characters for the names or properties when you create a new schema repository and sample user database, ClearQuest automatically sets the ClearQuest data code page value to the operating system code page of the system running the Maintenance Tool.

Note: The automatic setting of the ClearQuest data code page occurs only when you create a schema repository.

For instructions on how to set or change the ClearQuest data code page value, see *Setting the ClearQuest Data Code Page Value* on page 35.

ClearQuest Data Code Page and Existing Schema Repositories

If you have ClearQuest databases that were created with previous versions of ClearQuest, they may contain data from a variety of code pages. When you set the ClearQuest data code page, the data in your databases is not converted to characters in the selected code page. If your database contains characters that do not map to the new code page characters, data corruption will occur.

Important note: Before you set the ClearQuest data code page of an existing schema repository, you must convert the data in your vendor databases to the correct code page characters. For instructions, see your vendor database documentation.

For instructions on how to set the ClearQuest data code page for an existing schema repository, see *Existing Schema Repositories and User Databases* on page 36.

Selecting a ClearQuest Data Code Page

The ClearQuest data code page value you select for a schema repository depends on several factors, including the platforms of your clients and servers, the versions of ClearQuest you are running, and the character sets that your vendor database supports. Use the following guidelines and the flowchart in Table 9 to determine the appropriate ClearQuest data code page for your database set.

Guidelines for Selecting a ClearQuest Data Code Page

- To ensure data integrity, you must select a code page that is supported by Rational. These code pages have been thoroughly tested and approved for use with ClearQuest. For a list of Rational supported code pages, see Table 10 on page 31.
- You may choose to use a ClearQuest data code page that is not tested or approved by Rational. Setting the value to an unsupported code page will minimally allow for ClearQuest MultiSite to operate, and prevent divergence (but not data corruption).
- The ClearQuest data code page value you choose must be supported by the vendor databases you use for all ClearQuest databases. You must set the vendor database character set of all user databases to match the ClearQuest data code

page. For more information about the vendor database character set, see *About the Vendor Database Character Set* on page 32.

- All Windows clients must run the same operating system code page, and that code page must match the ClearQuest data code page. If you have mixed-platform environments (both Windows and UNIX clients), or clients using different operating system code pages, you must set the ClearQuest data code page to **20217** (ASCII), which is the common character set of all code pages. An alternate usage model is to set the ClearQuest data code page to a non-ASCII value and require that all users with UNIX systems interact with ClearQuest only with the Web client.

Table 9 Guidelines for Selecting a ClearQuest Data Code Page Value

Question	Yes	No
Do any of your ClearQuest clients[1] run on UNIX?	Set the ClearQuest data code page to **20127** (ASCII).	Skip to the next question.
Do your Windows clients and servers[2] use different operating system code pages?	Set the ClearQuest data code page to **20127** (ASCII).	Skip to the next question.
Do any of your Windows clients or servers run versions of ClearQuest earlier than v2002.05.01 Patch 02?	Set the ClearQuest data code page to **20127** (ASCII).	Set the ClearQuest data code page to any one of the Rational Supported Windows code pages in Table 10. Note that the ClearQuest data code page value you choose must be supported by your vendor database.

[1] The term “clients” refers to the native ClearQuest client, ClearQuest Designer, and the E-Mail Reader.
[2] The term “servers” refers to the ClearQuest Web server and the Rational Shipping Server.

Note: If you *do not want* to enforce a single code page for a schema repository, you can set the ClearQuest data code page to **NOCHECKING**. When you use this option, ClearQuest does not verify that the data you enter can be stored in the database or displayed by ClearQuest clients without being corrupted. Rational does not recommend that you set the ClearQuest data code page to this value. ClearQuest MultiSite *does not work* if the **NOCHECKING** value is specified.

Rational Supported Windows Code Pages

Table 10 lists the Windows code pages supported by Rational.

Note: If you have UNIX clients, **20127** (ASCII) is the only supported ClearQuest data code page.

Table 10 Rational Supported Windows Code Pages

Language	Windows code page
English	20127 (ASCII)
Danish, Dutch, English, Finnish, French, German, Italian, Norwegian, Brazilian Portuguese, Spanish, Swedish	1252 (Latin-1)
Chinese – Simplified	936 (GBK)
Japanese	932 (Shift-JIS)

Consequences of Your ClearQuest Data Code Page Selection

After you select a data code page value, you need to be aware of the effect this code page value will have on your databases. When the code page value is set, users may only be able to open certain databases in read-only mode, or they may not be able to open the database at all. For example:

- If you set the ClearQuest data code page to a non-ASCII value, users can only modify data in that database from a Windows client running the same operating system code page. If the code pages do not match, the database is opened in read-only mode.
- If you set the ClearQuest data code page to a non-ASCII value, UNIX clients will have read-only access to the databases. (UNIX users can choose to use the only the Web client, which prevents data corruption if a non-ASCII data code page value is selected.)
- If you set the ClearQuest data code page to a non-ASCII value, invalid characters can still potentially enter the database without being detected by ClearQuest. For example, ClearQuest cannot validate text that you cut from an e-mail or Web page and paste into a database record. If the e-mail or Web page text contains characters outside of the ClearQuest data code page, the characters are corrupted during display and may show up as invalid characters (for example, a question mark (?) character).

- ClearQuest v2003.06.00 validates all data entry, including the data entered through the Web client or the Rational E-Mail Reader. Data entered from integrations, such as the ClearQuest integration with Microsoft Project 2000, is also validated. The default setting of the ClearQuest data code page is **20217** (ASCII).
- Users cannot open a database with the ClearQuest Web client when the ClearQuest data code page value does not match the operating system code page of the Web server.
- ClearQuest does not validate data entered in attachments (for example, data in a Word document attached to a ClearQuest record). If you need to include data in a record that is not from your chosen ClearQuest data code page, you can enter it in an attachment. However, you cannot query for data stored in attachments, and reports and charts do not make use of this data.

Changing the ClearQuest Data Code Page Value

You can change the ClearQuest data code page of your schema repository from **20127** (ASCII) to a non-ASCII code page at any time without corrupting the data.

However, if you initially set the ClearQuest data code page to a non-ASCII value, changing it to any other code page may cause data corruption. Before you change the ClearQuest data code page value from non-ASCII, you must convert all existing data into characters that are supported by the new code page. For instructions, see your vendor database documentation.

Also, if you change the ClearQuest data code page value from **NOCHECKING** to any other value, data corruption may occur. Before you change the ClearQuest data code page value from **NOCHECKING**, you must convert all existing data into characters that are supported by the new code page. For instructions, see your vendor database documentation.

In a ClearQuest MultiSite environment, if you change the data code page from any non-ASCII value, incorrect data may also be in your oplogs. Therefore, Rational recommends that you remove all replicas, clean the database at the mastering site, scrub the oplogs, and then re-create your replicas. For information about removing and re-creating replicas, see the *Administrator's Guide* for Rational ClearQuest MultiSite.

About the Vendor Database Character Set

Each vendor database in your ClearQuest environment must use a character set that supports data from the ClearQuest data code page you chose for the schema repository. If the character set of the vendor database does not support the ClearQuest data code page, your database will not be able to correctly store the information entered by

ClearQuest. ClearQuest tries to minimize this occurrence by validating that the character set of your vendor database is compatible with the ClearQuest data code page value of your schema repository. The validation occurs in either of the following situations:

- When you manually set the ClearQuest data code page (see *Setting the ClearQuest Data Code Page Value* on page 35)
- When ClearQuest automatically sets the data code page if it encounters non-ASCII values in any of the fields when you create a schema repository

The following example shows how ClearQuest prevents data corruption by validating that the vendor database character set is compatible with your ClearQuest data code page setting.

Example

A ClearQuest administrator uses the Maintenance Tool to create a schema repository and sample user database. The administrator enters non-ASCII characters from the **1252** (Latin-1) code page into the description field. ClearQuest detects these non-ASCII characters and then validates that the character set of the SQL Server database can handle them. The validation status is displayed in the Maintenance Tool:

```
You entered non-ASCII characters for either the Master and/or Sample
database properties.

Verifying data code page support for Master database; 1252  (ANSI -
Latin I) character set.

Verifying data code page support for Sample database; 1252  (ANSI -
Latin I) character set.
```

Because the schema repository or sample user database properties contained characters from the Latin-1 code page, ClearQuest sets the ClearQuest data code page to **1252** (Latin-1):

```
Setting Master database to platform code page; 1252  (ANSI - Latin
I) character set.

Validating that database MASTR supports code page 1252  (ANSI -
Latin I)...

Successfully validated all databases.

Set Master database code page to 1252  (ANSI - Latin I) character
set.

Validating destination location...

Logging on to source...

Logging on to destination...
```

```
Initializing schema...
```

Setting the Vendor Database Character Set

Each vendor database has one or more character sets that are equivalent to the ClearQuest data code page values supported by Rational (see Table 10 on page 31). Table 11 on page 34 lists the equivalent character sets for Oracle, SQL Server, SQL Anywhere, and DB2 databases. For a complete list of character sets and for instructions about how to set the character set of your specific vendor database, see Rational Solution **167217701**. This solution is accessible from the Rational Support Web site at http://www.rational.com/support.

Note: If you have existing ClearQuest databases, you may be required to change their character sets to support data from your chosen ClearQuest data code page. Changing the character set of existing databases typically involves moving the old data into new databases. If your existing data is not supported by one of the ClearQuest data code pages, then you must first convert the data to values in one of the supported code pages (see Table 10 on page 31). The database vendors provide tools that you can use to analyze and convert your data. For more information, see Rational Solution **167217701**.

Table 11 Database Vendor Character Sets and Windows Code Pages

ClearQuest data code page	Oracle	SQL Server	SQL Anywhere	DB2
20127	US7ASCII	SQL_Latin1_General_CP437	437	ASCCSID
1252	WE8MSWIN1252	SQL_Latin1_General_CP1	English: 437 European: 850	SCCSID or 1252
936	ZHS16GBK	Chinese_PRC	EUC_CHINA	1386
932	JA16SJIS	Japanese	SJIS	943

Access Note:
If you have an Access database, you do not need to set the vendor database character set.

Note: When the ClearQuest data code page is set to **NOCHECKING** there is no code page validation. ClearQuest MultiSite does not work if the **NOCHECKING** value is used.

Setting the ClearQuest Data Code Page Value

The following sections explain how to set the ClearQuest data code page value for a new schema repository, for a schema repository created with an older version of ClearQuest, and for an existing schema repository in a MultiSite environment. If your ClearQuest environment has clients running earlier versions of ClearQuest, see *Interoperation with Previous Versions of ClearQuest* on page 38 for additional information.

New Schema Repositories

This procedure assumes that your schema repository does not have any user databases and that no users are accessing the schema repository from ClearQuest Designer.

1. **Note:** The commands in this example use the **2003.06.00** schema repository.Choose a ClearQuest data code page value for your schema repository. See *Selecting a ClearQuest Data Code Page* on page 29.
2. Determine the character set support for your vendor database. See *About the Vendor Database Character Set* on page 32.
3. Open a command prompt and change the directory to the location of your ClearQuest installation (for example, C:\Program Files\Rational\ClearQuest).
4. Run the **installutil lscodepage** command to determine the code page of your operating system.

   installutil lscodepage -dbset 2003.06.00 admin_user admin_password

   ```
   Code page of 2003.06.00:  20127 (US-ASCII) (default)
   Code page of client:  1252  (ANSI - Latin I)
   ```

   Note: In this example, the operating system code page is **1252** (Latin-1). The ClearQuest data code page value of the 2003.06.00 schema repository is currently the default value of **20127** (ASCII).
5. Run one of the following **installutil** subcommands to set the ClearQuest data code page to the appropriate value:
   - On Windows, if the operating system code page value listed by **lscodepage** is supported by Rational, run the **installutil setdbcodepagetoplatformcodepage** command to set the ClearQuest data code page to this value. For a list of code pages supported by Rational, see Table 10 on page 31.

     installutil setdbcodepagetoplatformcodepage -dbset 2003.06.00 admin_user admin_password
   - To set the ClearQuest data code page to **20127** (ASCII), run the **installutil setdbcodepagetoascii** command.

installutil setdbcodepagetoascii -dbset 2003.06.00 admin_user admin_password

- If the operating system code page value listed by **lscodepage** is an unsupported code page, and you choose not to prevent data corruption in your database, you can use the **–force** option to set the ClearQuest data code page to this value. However, using a code page other than the ones listed in Table 10 on page 31 is not supported by Rational.

  installutil setdbcodepagetoplatformcodepage -dbset 2003.06.00 -force admin_user admin_password

- If you do not want to enforce a single code page, you can set the ClearQuest data code page to **NOCHECKING**.

  Warning: Setting the ClearQuest data code page to **NOCHECKING** may result in data corruption if ClearQuest clients use different operating system code pages and enter non-ASCII data. This option is not recommended for new schema repositories and ClearQuest MultiSite *will not function* if you choose this option.

  installutil setcodepagetonochecking -dbset 2003.06.00 admin_user admin_password

6 The ClearQuest data code page for your schema repository is set and any user databases you create inherit this code page setting.

Existing Schema Repositories and User Databases

Note: The commands in this example use the **2002.05.00** database set.

To set the ClearQuest data code page value of an existing schema repository or user database:

1 Choose a ClearQuest data code page value for your schema repository. See *Selecting a ClearQuest Data Code Page* on page 29.

2 Determine the character set support for your vendor database. See *About the Vendor Database Character Set* on page 32.

Note: If your existing vendor database character set does not match the ClearQuest data code page you selected in Step 1, you may need to change the vendor database character set and clean up any data that is not supported. For instructions on how to set the character set for your specific vendor database, see your vendor database documentation. Additionally, Rational Solution **167217701** contains information on how to locate vendor database specific documentation. This Solution is accessible from the Rational Support Web site at http://www.rational.com/support.

3 Make sure that no ClearQuest clients are connected to the schema repository or user databases. Otherwise, after you set the ClearQuest data code page value, connected clients can still enter data outside of the new code page.

4 Backup the schema repository and all user databases.

5 Run the **installutil lscodepage** command to determine the code page of your operating system.

installutil lscodepage -dbset 2002.05.00 admin_user admin_password

```
Code page of 2002.05.00:  20127 (US-ASCII) (default)
Code page of client:  1252  (ANSI - Latin I)
```

Note: In this example, the operating system code page is **1252** (Latin-1). The ClearQuest data code page value of the 2002.05.00 schema repository is currently the default value of **20127** (ASCII).

6 Run one of the following **installutil** subcommands to set the ClearQuest data code page to the appropriate value:

- On Windows, if the operating system code page value listed by **lscodepage** is supported by Rational, run the **installutil setdbcodepagetoplatformcodepage** command to set the ClearQuest data code page to this value. For a list of code pages supported by Rational, see Table 10 on page 31.

  installutil setdbcodepagetoplatformcodepage -dbset 2002.05.00 admin_user admin_password

- To set the ClearQuest data code page to **20127** (ASCII), run the **installutil setdbcodepagetoascii** command.

  installutil setdbcodepagetoascii -dbset 2002.05.00 admin_user admin_password

  Note: If you change the ClearQuest data code page value to **20217** (ASCII) from another code page, all data that cannot be represented by ASCII characters can be corrupted.

- If the operating system code page value listed by **lscodepage** is an unsupported code page, and you choose not to prevent data corruption in your database, you can use the **–force** option to set the ClearQuest data code page to this value. However, the use of this option is not supported by Rational.

  installutil setdbcodepagetoplatformcodepage -dbset 2002.05.00 -force admin_user admin_password

- If you do not want to enforce a single code page, you can set the ClearQuest data code page to **NOCHECKING**.

Warning: Setting the ClearQuest data code page to **NOCHECKING** may result in data corruption if ClearQuest clients use different operating system code pages and enter non-ASCII data. This option is not recommended for new schema repositories and ClearQuest MultiSite will not function if you choose this option.

installutil setcodepagetonochecking -dbset 2002.05.00 admin_user admin_password

7 The ClearQuest data code page for your schema repository is set and any user databases you create inherit this code page setting.

Interoperation with Previous Versions of ClearQuest

Rational recommends that you upgrade all ClearQuest clients to ClearQuest v2003.06.00. (For MultiSite specific requirements, see *MultiSite and the ClearQuest Data Code Page Value* on page 40.) Previous versions of ClearQuest do not have features to enforce a single code page in your ClearQuest environment. Also, if you decide to run versions of ClearQuest prior to ClearQuest v2002.05.01 Patch 02, you are limited to ASCII data entry only (see Table 9 on page 30).

ClearQuest v2003.06.00 includes a CharacterSetValidation package that can help prevent data corruption in ClearQuest records if you use earlier clients. However, other ClearQuest objects, such as schemas, query names, record names, report formats, and user profile information are all vulnerable to data corruption. You must train your users to create these objects using only characters valid for the ClearQuest data code page.

A ClearQuest administrator should apply the CharacterSetValidation package to all schemas, and copy the ClearQuest data code page value to all associated user databases.

About the CharacterSetValidation Package

The CharacterSetValidation package prevents clients running earlier versions of ClearQuest from entering data in a user record from a code page other than the data code page value of that database. If you do not apply the CharacterSetValidation package to your schemas, it is possible for users to enter unsupported data from the client and for data to be corrupted when modified on certain clients.

The CharacterSetValidation package adds two Perl hooks for all user modifiable record types:

- The base access control hook “CheckCodePageMismatch” allows read-only access to records if there is a mismatch between the ClearQuest data code page and the operating system code page of the client.
- The base validation hook “CharacterSetValidation” rejects any characters that are not in the ClearQuest data code page after modifying a record.

This Package also adds a new stateless record type, _ratl_data_code_page, that provides support for clients that run older versions of ClearQuest.

Applying the CharacterSetValidation Package

ClearQuest v2003.06.00 includes a script to install the CharacterSetValidation package that simultaneously applies the package to every record in a schema. You can also use the Package Wizard to apply the package, but it takes longer because it applies the package to one record and reruns until all records are enabled. You must reapply the package each time you add a new record type to a schema.

Note: You do not need to apply the CharacterSetValidation package if all clients are running ClearQuest v2003.06.00 or later.

For each schema, follow these steps to apply the CharacterSetValidation package:

1. From the ClearQuest Designer, check in the schema and make sure that each user database is using the latest version of the schema. Otherwise, when you upgrade your user databases in Step 4 you may include schema changes that are not related to applying the package.
2. Run the script *cquest-home-dir*\apply_character_set_validation_package.bat.

   apply_character_set_validation_package *admin-username admin-password myDB myDBSet schema*

   where

myDB	Any one of the user databases using the schema being modified
myDBSet	The schema’s database set name
schema	The schema to which the CharacterSetValidation package is being applied

3. The **apply_character_set_validation_package** script attempts to check in the new version of the schema and fails if validation errors occur. Contact Customer Support if validation errors occur on schema check in.
4. Log in to the ClearQuest Designer and upgrade the user databases.

5 Copy the ClearQuest data code page value to all user database. For instructions, see *Copying the ClearQuest Data Code Page Value to User Databases*.

Copying the ClearQuest Data Code Page Value to User Databases

After you apply the CharacterSetValidation package to a schema, run the **installutil setuserdbcodepage** command to copy the ClearQuest data code page value to your user databases. Copying this value to allows clients running earlier versions of ClearQuest to access the data code page information and prevent characters outside of the chosen code page from entering the database.

- To copy the ClearQuest data code page for a specific user database, specify a database name:

  installutil setuserdbcodepage -dbset *myDBSet admin_user admin_password user_db_name*

- To copy the ClearQuest data code page for all user databases, do not specify a database name:

  installutil setuserdbcodepage -dbset *myDBSet admin_user admin_password*

Note that if you change the ClearQuest data code page, you need to run the **setuserdbcodepage** command again to recopy the value to your user databases.

MultiSite and the ClearQuest Data Code Page Value

All MultiSite synchronization servers should run on machines with the same operating system code page and must use ClearQuest v2003.06.00 or later. There is a *temporary* restriction that all MultiSite synchronization servers must be run on Windows machines.

Note: You must upgrade your MultiSite synchronization servers to v2003.06.00 *before* you upgrade your clients.

After you set the data code page value in a MultiSite environment, the data code page value of a database is compared to the operating system code page of the client during export of a packet. The export operation succeeds only if the code pages are the same, or if the ClearQuest data code page value is set to **20127** (ASCII). The ClearQuest data code page value is then written in the header of the packet being exported. Import succeeds only if the ClearQuest data code page value in the header is the same as, or a subset of, the operating system code page of the importer process.

Remember that on UNIX, ClearQuest supports only ASCII, or printable English characters. If you run UNIX clients, you must set the ClearQuest data code page to **20127** (ASCII), or enter data only from the Web client.

MultiSite Synchronization Failure Example

With ClearQuest MultiSite, if the same code page is not used by all clients that access a replica, invalid data can enter the oplog. If this occurs, synchronization may fail with the following error:

```
Multiutil: Cannot convert XML data to local codepage.

Offending characters were found at:
line 30
column 0
byte 920

in oplog serial number 185458

The packlet being exported was oplog serial number 185458 of database
'SAMPL', replica 'SITEA' (UUID =1802CAEC-44D1-11D5-AF98-00B0D0682333).

Multiutil: The syncreplica -export command failed.

Multiutil: Error: Cannot remove working directory 'c:\temp\export'
(The directory is not empty.)
```

You cannot synchronize your replicas until the invalid data is removed from the record and the oplog is repaired

ClearQuest Integrations and the ClearQuest Data Code Page Value

Because all data in a ClearQuest database must be from the same code page, there are several things you need to keep in mind when using ClearQuest integrations:

- To successfully set up an integration, the integrated product must be running on a machine with the same operating system code page as the ClearQuest data code page of the database set, or the ClearQuest data code page must be set to **20217** (ASCII).
- To enter non-ASCII data into a ClearQuest database through an integration, the integrated product must be running on a machine with the same operating system code page as the ClearQuest data code page of the database set. ClearQuest blocks any non-ASCII data from an integrated product that uses a different code page.
- If your ClearQuest data code page value is set to **20217** (ASCII), you should be aware that the integration will reject all characters from a non-ASCII code page. However, the products that ClearQuest integrates with do not limit data entry to only ASCII characters. You will not be warned by the integrated product about the non-ASCII characters, but a ClearQuest database set with an ASCII data code page value will reject all non-ASCII characters coming from the integrated product.

Examples

The following examples show the behavior of ClearQuest integrations with different combinations of operating system code pages and ClearQuest data code pages.

- This example shows how you are prevented from setting up the ClearQuest integration with Microsoft Project 2000 if there is a code page mismatch:

  A ClearQuest administrator creates a schema repository and user database and sets the ClearQuest data code page to **1252** (Latin-1). The administrator attempts to link to this new database from Microsoft Project 2000 running on a system with a Japanese operating system code page. The link fails and the administrator sees this error message:

  ```
  The ClearQuest database codepage is not compatible with the client
  machine's codepage. Because of this, it is not possible for Project
  Tracker to continue.
  ```

- This example shows how a code page mismatch can prevent Project Tracker from functioning, even though the integration itself was successful:

  A ClearQuest administrator creates a schema repository and user database and sets the ClearQuest data code page to **20127** (ASCII). A user installs ClearQuest on a Japanese Windows operating system. The user creates and modifies defects in ClearQuest. The user successfully sets up Project Tracker. The user creates a plan in Microsoft Project 2000 that contains Japanese characters.

  From Microsoft Project 2000, the user selects the Project Tracker function **Export new tasks to ClearQuest**. ClearQuest determines that the tasks contain non-ASCII characters and prevents the export from completing. The user needs to remove the non-ASCII characters from the Microsoft Project 2000 tasks and run the export again.

Managing Databases 3

A significant portion of the Rational ClearQuest administrator's responsibilities involve managing databases and related entities. Therefore, you need to have a good understanding of databases to maintain ClearQuest databases with the ClearQuest Maintenance Tool and the ClearQuest Designer.

With the ClearQuest Maintenance Tool, you can create and manage schema repositories and connections, create sample databases, and upgrade schema repositories and user databases to new feature levels of ClearQuest.

With the ClearQuest Designer, you can create, manage, and upgrade user databases to a new version of a schema.

The topics covered include:

- *Understanding the Process for Working with Databases* on page 44.
- *Establishing Procedures to Back Up Databases* on page 47.
- *Creating New Schema Repositories and Sample Databases* on page 49.
- *Managing Connections* on page 55.
- *Moving Schema Repositories* on page 62.
- *Upgrading Databases to New Feature Levels of ClearQuest* on page 66.
- *Moving and Deleting User Databases* on page 68.
- *Vendor Database Parameters* on page 73.

Understanding the Process for Working with Databases

As the ClearQuest administrator, you create and manage different types of databases (primarily schema repositories and user databases). For an overview of schemas, schema repositories, user databases, and database connections, refer to Chapter 1, *Understanding ClearQuest Administration*.

To manage these databases, you use the ClearQuest Maintenance Tool, ClearQuest Designer, and tools from the vendor of the database management system you use with ClearQuest.

A typical sequence of activities that might be performed over the life of a project are summarized in Table 12.

Table 12 ClearQuest Database Activities

Activity	Tool/interface	Comments
Install the database management system.	Database vendor tools	A runtime version Microsoft Access is installed when you install ClearQuest. SQL Anywhere is available on the ClearQuest installation disk, but you must choose to install it. Other databases must be licensed from the database vendors and installed from their media.
Establish procedures to back up databases and schema repositories.	Database vendor tools	Perform backups regularly. Be sure to back up schema repositories and user databases at the same time. See *Establishing Procedures to Back Up Databases* on page 47 and the database vendor's documentation.
Create empty vendor databases and database aliases.	Database vendor tools	For Microsoft Access and SQL Anywhere, you can create the databases from within the ClearQuest Maintenance Tool. For Oracle and DB2, you must also create a database aliases. See the *Installation Guide* for Rational Server Products for Windows and the *Installation Guide* for Rational ClearQuest Product Family.

Table 12 ClearQuest Database Activities

Activity	Tool/interface	Comments
Set the character set (code page) for the vendor databases.	Database vendor tools	The character set of your vendor databases should match the ClearQuest data code page value of your schema repository. For information about choosing an appropriate data code page value, see *Selecting a ClearQuest Data Code Page* on page 29. For information about setting the vendor database code page, see *Setting the Vendor Database Character Set* on page 48.
Create a new schema repository and connection.	ClearQuest Maintenance Tool	Built on one of the empty vendor databases created earlier. See *Creating New Schema Repositories and Sample Databases* on page 49.
Set the ClearQuest data code page value for the schema repository.	installutil command line utility	The ClearQuest data code page value prevents data corruption in your database set. For information about choosing an appropriate data code page value, see *Selecting a ClearQuest Data Code Page* on page 29. For instructions on setting the data code page value, see *Setting the ClearQuest Data Code Page Value* on page 35.
Apply the CharacterSetValidation package to the schema repository if it will be accessed by clients running versions of ClearQuest earlier than v2003.06.00.	ClearQuest Designer	The CharacterSetValidation package verifies that only valid data is entered by clients running older versions of ClearQuest. For instructions, see *How to Upgrade Packages* on page 140.
Create a sample user database with sample data.	ClearQuest Maintenance Tool	Use the sample database to explore the schema and process model. See *Creating New Schema Repositories and Sample Databases* on page 49.
Create a new user database and associate it with a schema.	ClearQuest Designer	The user database uses the blueprint of the schema. See Chapter 4, *Working with ClearQuest Schemas*.

Table 12 ClearQuest Database Activities

Activity	Tool/interface	Comments
If you applied the CharacterSetValidation package to the schema repository, set the ClearQuest data code page value for each user database (test and production).	installutil command line utility	The ClearQuest data code page value is used with the CharacterSetValidation package to verify that only valid data is entered by clients running older versions of ClearQuest. For instructions, see *Setting the ClearQuest Data Code Page Value* on page 35.
Manage connections and create connection profiles.	ClearQuest Maintenance Tool	Connections are used to give users access to a schema repository and the associated user databases. See Chapter 3, *Managing Connections*.
Create a new schema or modify an existing schema.	ClearQuest Designer	The schema version number is automatically incremented. Modifying the schema does not change any user databases until they are upgraded. See *Working with ClearQuest Schemas* on page 79.
Create a test database.	ClearQuest Designer	Used to test the new or modified schema. See *Working with ClearQuest Schemas* on page 79.
Check in the revised schema and upgrade user databases.	ClearQuest Designer	Upgraded databases now reflect changes made to the schema. See Chapter 4, *Working with ClearQuest Schemas*.
Allow users to add and modify data in the user databases.	ClearQuest client, ClearQuest Web	Use only ClearQuest tools to modify ClearQuest data and schema. Using tools from the database vendor can destroy the integrity of the databases. Also, if the Rational E-Mail Reader and the Rational ClearQuest Mail Service is implemented, users can submit and modify change requests using e-mail.
Delete and undelete user databases; move user databases to other locations; change vendor database management system managing the user databases.	ClearQuest Designer	When deleting a user database, ClearQuest Designer removes the link between the schema repository and the database, but does not physically delete the database. Use the database vendor tool to delete the physical database. See *Moving and Deleting User Databases* on page 68.

Table 12 ClearQuest Database Activities

Activity	Tool/interface	Comments
Move a schema repository and a connection to another location; change the vendor database management system managing the schema repository.	ClearQuest Maintenance Tool	See *Moving Schema Repositories* on page 62.
Upgrade schema repositories and user databases to new feature levels of ClearQuest.	ClearQuest Maintenance Tool	See *Upgrading Databases to New Feature Levels of ClearQuest* on page 66. Before taking any action, carefully review the upgrade guidelines described in the *Upgrade Guide* for Rational Suite for Windows and the *Installation Guide* for ClearQuest Product Family for UNIX.

Establishing Procedures to Back Up Databases

Before creating databases of any kind, it is important that you establish procedures and schedules to perform 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 perform regular backups.

Whenever you perform database backups, you must back up all of your ClearQuest databases at the same time, including your schema repository and all associated user databases. This preserves both your data and customizations, ensuring that the schema repository and user databases are synchronized in case you need to restore from a backup.

Also, be sure to use only ClearQuest tools to directly modify ClearQuest data or tables.

If you need help backing up databases, contact Rational Support.

Creating Empty Vendor Databases and Database Aliases

ClearQuest schema repositories and user databases are both types of databases that are physically managed by a database management system. With ClearQuest, you have a choice of several database management systems, which, depending on your platform, may include Sybase SQL Anywhere, Microsoft Access, Microsoft SQL Server, Oracle DBMS, or IBM DB2.

Before you can create a schema repository and user databases, you must first create empty databases.

For Microsoft SQL Server, Oracle DBMS, and IBM DB2, you must create these empty databases using tools provided by those vendors. For instructions on configuring vendor databases, see the *Installation Guide* for Rational Server Products for Windows and the *Installation Guide* for the Rational ClearQuest Product Family for UNIX.

For IBM DB2 and Oracle DBMS, you need to create database aliases. For DB2, you need to create aliases on the server and on both Windows and UNIX clients. For Oracle, you need to create aliases on the server and on Windows clients. For Oracle, you do not need to create aliases on UNIX clients, because Open Link is used for database connectivity, but you do need to have Open Link on the Oracle server. For information on how create DB2 and Oracle aliases, see the *Installation Guide* for Rational Server Products and the *Installation Guide* for Rational Desktop Products for Windows, and the *Installation Guide* for the ClearQuest Product Family for UNIX.

For SQL Anywhere and Microsoft Access, you create the empty database when you create the schema repository or user database; you do not need to perform this step in advance.

Setting the Vendor Database Character Set

ClearQuest requires that a single code page or character set is used across all vendor databases, schema repositories, and user databases. Enforcing a single code page helps ensure that each database contains only data that is valid for that database set.

When you create a new vendor database, you must set its character set to a value that corresponds to the ClearQuest data code page value of its schema repository. For information about the ClearQuest data code page value for a schema repository, see *About the ClearQuest Data Code Page* on page 28.

Each vendor database has several character sets that are equivalent to the ClearQuest data code page values supported by Rational. Table 13 contains a list of the Oracle database character sets and their corresponding data code page values that can help you choose an appropriate character set. For example, if your schema repository has a ClearQuest data code page value of **932**, and you are using an Oracle database, you would set the vendor database character set to **JA16SJIS**.

For instructions about how to set the character set for your vendor database, see the vendor database documentation.

Note: Changing the character set value of an existing database may not convert all data to the new character set. Before you change the character set of a databases, consult your vendor database documentation.

Table 13 Oracle Character Sets and Windows Code Pages

ClearQuest data code page value	Oracle character set	Languages
1252	WE8MSWIN1252	Danish, Dutch, Finnish, French, German, English, Brazilian Portugal, Italian, Norwegian, Spanish, Swedish
20127	US7ASCII	English/ASCII
932	JA16SJIS	Japanese
936	ZHS16GBK	Simplified Chinese

Creating New Schema Repositories and Sample Databases

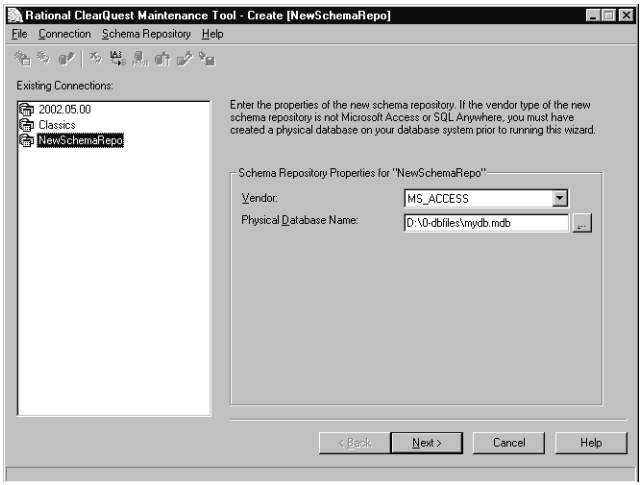
You use the ClearQuest Maintenance Tool to create a new schema repository. A schema repository, which is also sometimes referred to as a master repository or a master database, is a special type of database used to store and manage a group of schemas, including all of the versions of those schemas.

When you create a new schema repository, the ClearQuest Maintenance Tool automatically creates a connection, which is also referred to as a database set. A connection consists of one schema repository and all of the user databases associated with schemas in that schema repository.

To use Microsoft SQL Server, Oracle DBMS, or IBM DB2 as the database management system for your schema repository, you must have already created an empty database using tools provided by the database vendor. Also, for DB2 and Oracle, you need to establish an alias name for the database server machine before creating a schema repository. You can have several schema repositories point to the same alias; you do not need to create a new alias for every schema repository.

To create a schema repository:

1 In the ClearQuest Maintenance Tool, select **Schema Repository > Create**.



2 In the **Existing Connections** pane, enter a name for the schema repository connection in the highlighted item and press ENTER.

The first connection name defaults to ClearQuest version 2003.06.00. You can choose to rename the connection later.

3 In the **Schema Repository Properties** area, select a database **Vendor** and enter the required database properties. The required properties vary according to the database vendor you select.

If you select Microsoft SQL Server, Oracle DBMS, or IBM DB2 as the vendor database, enter the physical database name of one of the empty databases you created earlier using the database vendor's tools. If you select Oracle or DB2, you need to specify the alias name for the database server machine.

If you select SQL Anywhere or Access, enter a physical database name and the ClearQuest Maintenance Tool creates a new database with that name. For SQL Anywhere, the **HOST NAME** is required. For more information, see *Vendor Database Parameters* on page 73.

Note: With ClearQuest version 2003.06.00, you can only create new schema repositories with SQL Anywhere 8.0.

4 Click **Next**.

5 When you create a new schema repository, you are given the option of creating a sample user database. A sample user database is a working user database that provides 40 sample defect records of user information, as well as sample queries

and reports. We recommend that you use the sample user database to familiarize yourself with how ClearQuest works, and to help train users.

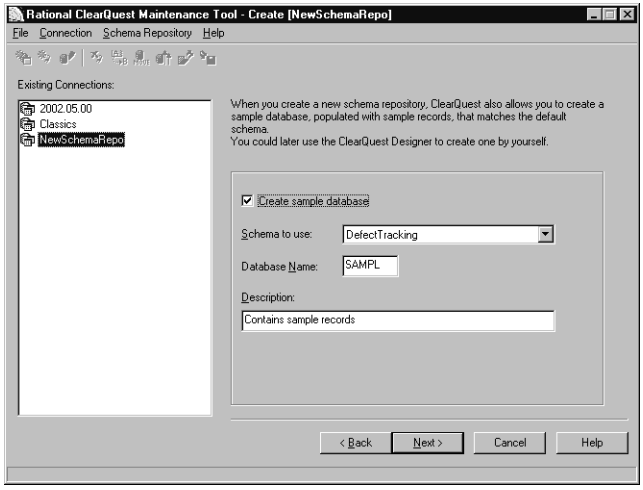
To create a sample database:

a Check **Create sample database**.

b Select a **Schema to use** for the sample database.

c Enter a **Database Name** of five alphanumeric characters or less. The default name is SAMPL.

d Enter a **Description** of the sample database you want.



6 Click **Next**.

7 In the Sample User Database Properties area, select the database **Vendor** for the sample user database and enter a **Physical Database Name** of no more than five alphanumeric characters.

8 Click **Finish**.

If the properties of your new schema repository and sample user database contain characters outside of the ASCII code page, ClearQuest automatically sets the ClearQuest data code page value of the schema repository. It is set to the operating system code page of the machine the Maintenance Tool is running on. Otherwise, the ClearQuest data code page value is still unset. If you do not explicitly set this value, your database is limited to ASCII data only and the ClearQuest Designer will display a warning message each time you open the schema repository.

For more information about the ClearQuest data code page value, see *About the ClearQuest Data Code Page* on page 28. For instructions to set or change the data code page value, see *Setting the ClearQuest Data Code Page Value* on page 35.

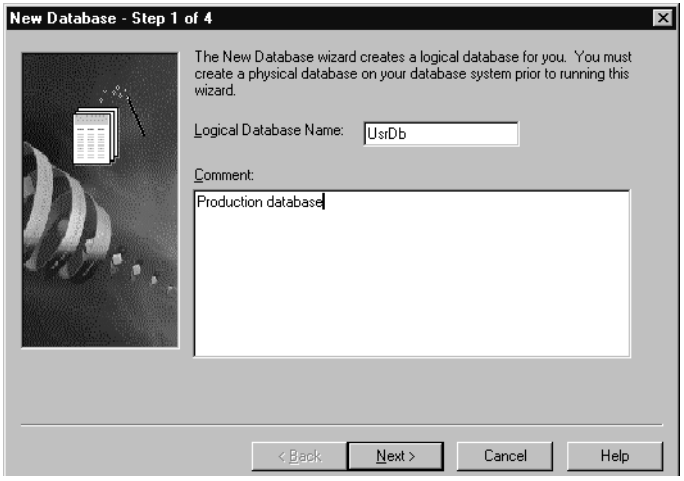
Creating User Databases

After you create a schema repository, you can use ClearQuest Designer to create new user databases. A user database manages information about change requests entered by members of the development team. A user database is always associated with a schema, which describes the data, process model, reports, and queries for that database.

Just as with creating a schema repository, if you want to use Microsoft SQL Server, Oracle RDBMS, or IBM DB2, as the database management system for your user database, you must first create an empty database using tools provided by the database vendor.

To create a new user database:

1. Start ClearQuest Designer by clicking **Start > Programs > Rational Software > Rational ClearQuest > ClearQuest Designer**.
2. If the **Open Schema** dialog box appears, click **Cancel**.
3. Click **Database > New Database**.
4. In Step 1 of the new Database Wizard, enter a **Logical Database Name** for the user database. The name must be five or fewer alphanumeric characters.

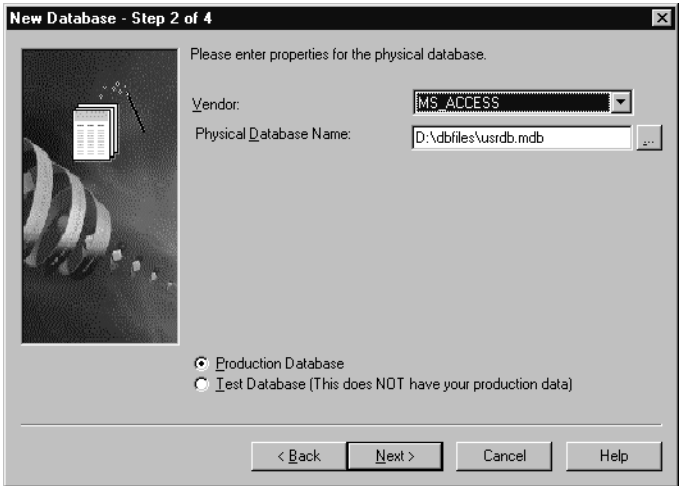


5. Enter a **Comment**, if desired, then click **Next**.
6. In Step 2 of the New Database Wizard, select the **Vendor** of the database to use for the user database and enter the required database properties. The required properties vary according to the database vendor you select.

If you select Microsoft SQL Server, Oracle RDBMS, or IBM DB2 as the vendor database, enter the physical database name of one of the empty databases you created earlier using the database vendor's tools.

If you select SQL Anywhere or Access, enter a physical database name and ClearQuest Designer will create a new database with that name. For SQL Anywhere, the **HOST NAME** is required. For more information, see *Vendor Database Parameters* on page 73.

Note: With ClearQuest version 2003.06.00, you can only create new schema repositories with SQL Anywhere 8.0.

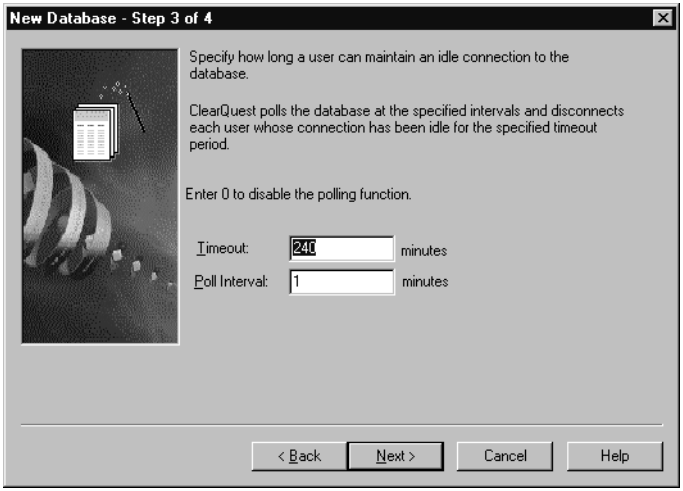


7 Select whether the user database will be a **Production Database** or a **Test Database**, then click **Next**.

8 At the bottom of Step 2, there are options to specify whether the user database is a either **Production Database** or a **Test Database**. The default is set to **Production Database**. Then click **Next**.

A *test database* is a temporary user database that you create when you are developing a new schema or a new version of an existing schema. With the test database you can make sure that the new schema performs as expected, without running the risk of damaging production data in a user database. For more information on test databases, refer to Chapter 4, *Working with ClearQuest Schemas*.

9 In Step 3 of the New Database Wizard, enter the number of minutes for **Timeout** and for **Poll Interval**. Set both to zero (0) if you do not want the database to time out for idle connections. Click **Next**.



10 In Step 4 of the New Database Wizard, select a schema from a list of predefined schemas, to associate with the new user database. The choice of schemas includes:

- Blank – A minimum configuration base for building schemas.
- Common – A base-level configuration with the most commonly used schema elements.
- Defect Tracking – A schema with features for defect-tracking processes.
- AnalystStudio – A schema with fields and rules that allow ClearQuest to be used with Rational AnalystStudio.
- DevelopmentStudio – A schema with fields and rules that allow ClearQuest to be used with Rational DevelopmentStudio.
- TestStudio – A schema with fields and rules that allow ClearQuest to be used with Rational TestStudio.
- UnifiedChangeManagement – A schema with fields and rules that allow ClearQuest to be used with Rational UCM.
- Enterprise – A schema with fields and rules that allow ClearQuest to be used with Rational Suite Enterprise.

11 Click **Finish**.

Note: If your new user database will be accessed by clients running versions of ClearQuest earlier than v2003.06.00, you must apply the CharacterSetValidation package and set the ClearQuest data code page value for the user database.

For more information, see *How to Upgrade Packages* on page 140 and *Setting the ClearQuest Data Code Page Value* on page 35.

Managing Connections

After you create a schema repository and user databases associated with that schema repository, use the ClearQuest Maintenance Tool to manage the connection.

A connection, also referred to as a database set, consists of one schema repository and all of its associated user databases.

Connections also help you manage the environment seen by team members. For example, when a team member logs on to the ClearQuest client, he or she is first asked to select the connection that contains the database to be used, and then to select the database itself. (This assumes the team is authorized to use databases in more than one connection; if the team member works in only one connection, then the dialog box offering this choice is not displayed at log in.)

This section describes how to do the following:

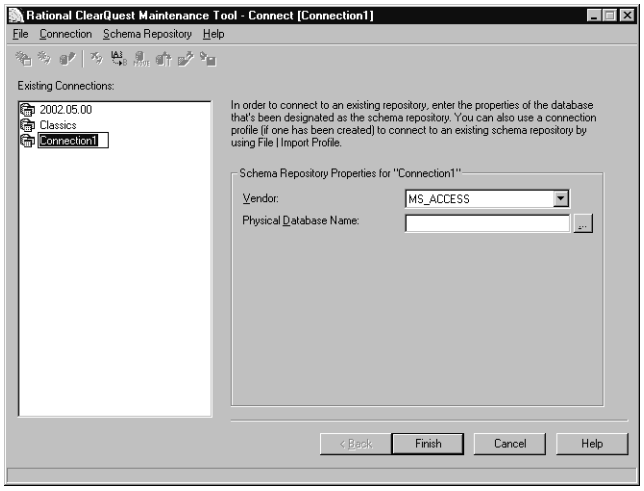
- Create new connections.
- Rename connections.
- Edit connections to change the underlying database management system, change the properties of the database, or change the location of the schema repository.
- Delete connections.
- Duplicate a connection in order to create a new connection similar to the existing one.

Creating New Connections

Note: You cannot create a connection if the ClearQuest data code page value of the schema repository does not match the operating system code page of the machine running the Maintenance Tool. For more information about the ClearQuest data code page value, see *About the ClearQuest Data Code Page* on page 28.

To create a new connection:

1 In the ClearQuest Maintenance Tool, select **Connection > New**.



2 In the **Existing Connections** area, enter a name for the connection in the highlighted item.

3 In the **Schema Repository Properties** area, select the **Vendor** of the physical database you are using for the schema repository, and enter the required properties. The required properties vary according to the database vendor you select.

4 Click **Finish**.

Modifying Connections

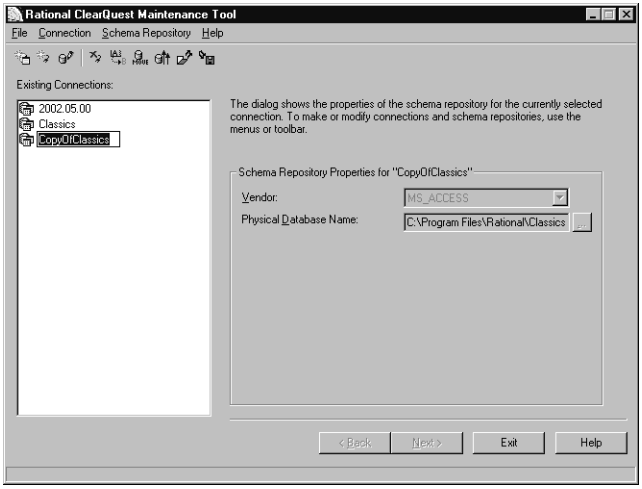
You can modify an existing connection at any time by using the ClearQuest Maintenance Tool. You can rename the connection, edit the database properties, and create duplicate connections.

Renaming a Connection

Renaming a connection does not change the schema or user database information associated with that connection. Connections are maintained on a machine basis. A schema repository connection can have different names on different machines.

To rename a connection using the ClearQuest Maintenance Tool:

1 In the **Existing Connections** area, select the connection you want to rename, then click **Connection > Rename**.



2 Type the new name for the connection, then press ENTER.

Editing a Connection

By editing a connection, you can change the database management system used to manage a schema repository, or change the database properties of the schema repository.

If you move a schema repository to a new location, you must edit the connection to reflect that change in location.

To edit a connection using the ClearQuest Maintenance Tool:

1 In the **Existing Connections** area, select the connection you want to edit, then click **Connection > Edit**.

2 In the **Schema Repository Properties** area, select the **Vendor** of the physical database you are using for the schema repository, and enter the required properties. The required properties vary according to the database vendor you select.

3 Click **Finish**.

Deleting a Connection

You can delete a connection when you no longer need it. Connections are kept on a machine basis and do not affect other ClearQuest users or the databases that the connection designates.

To delete a connection using the ClearQuest Maintenance Tool:

1 In the **Existing Connections** pane, select the connection you want to delete, then click **Connection > Delete**.

2 Click **Yes** to confirm the deletion.

Note: To retrieve a deleted connection, use the **Connection > New** option or import an existing Connection Profile.

For more information, see *Using Connection Profiles* on page 60.

Duplicating an Existing Connection

You may want 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 first duplicate the existing connection, then edit the properties of the duplicated connection to match the new schema repository.

To duplicate an existing connection using the ClearQuest Maintenance Tool:

1 In the Existing Connections pane, select the connection you want to duplicate, then click **Connection > Duplicate**.

2 Type a new name for the duplicate connection, then press ENTER.

Note: To edit the properties of a duplicated connection, click **Connection > Edit**.

Maintaining Connections with a UNIX Client

To create or update a connection for a ClearQuest client on UNIX, you can run the **cqreg add_dbset** command from that client:

- To connect to the schema repositories you set up in the previous release, move the connection profiles you preserved to the database directory in **$CQ_HOME/../ClearQuest_Databases/2003.06.00**.
- To establish a new connection, you may use the Register Database dialog box that appears the first time you run ClearQuest. This dialog box appears only once. After that, you need to run the **cqreg add_dbset** command to update your connections.
- To update an existing connection or establish a new connection, run the **cqreg add_dbset** command.

cqreg Options

cqreg add_dbset –v· endor *db_vendor* **–s· erver** *db_server* **–d· atabase** *database* **–u· ser** *user* **–p· assword** *password* **–dbset** *dbset_name* [**–co** *connect_options*]

Option	Description
–v· endor	The database vendor.
–s· erver	For Oracle and SQL Server, the name of the database server host. For DB2, the database node.
–d· atabase	For SQL Server and DB2, the name of the schema repository database. For Oracle, the SID.
–u· ser	The user name for accessing the database.
–p· assword	The password for accessing the database.
–dbset	The name of the connection. By default the name of the connection is the ClearQuest version (for example, 2003.06.00).
–co	(Applicable to Oracle only) The connect options. Use this option to specify the following values: "HOST=hostname;SID=sid_name;SERVER_VER=8.1"

The following are examples of adding connections for various vendor databases.

For Oracle:

cqreq add_dbset -v ORACLE -s cqtest1 -d CQ1 -u cq_unix_m -p cq_unix_m -dbset CQ_UNIX_TEST -co "SERVER_VER=8.1"

For SQL Server:

cqreg add_dbset -v SQLSERVER -s cqtest2 -d musitea_m -u mila -p mila -dbset CQ_DEV

For DB2:

cqreg add_dbset -v DB2 -s db2tcpnode -d musitea_m -u mila -p mila -dbset CQ_DEV

Using Connection Profiles

A connection profile is a text file that contains information about how to access a schema repository.

Without a connection profile, users need to enter information about the connection and its location each time they log on. With the connection profile, users can import one file that has all the necessary information, after which they are able to access the schema repository and associated databases. This reduces the time required to roll out a new schema repository to end users.

You can also use a connection profile to reconnect to a schema repository and associated databases after they have been moved to a new location.

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

Using connection profiles involves:

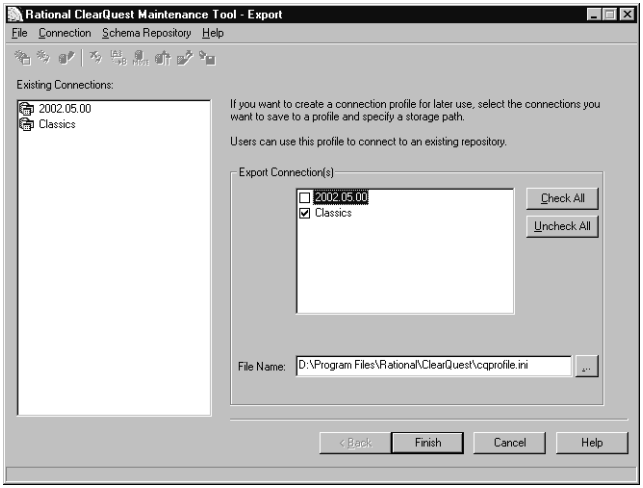
- *Exporting a Connection Profile* on page 60
- *Importing a Connection Profile* on page 61

Exporting a Connection Profile

To create a connection profile, you export information containing schema repository properties for a connection, or multiple connections, to a text file. This file can later be imported by users to access the connection.

To export a connection profile:

1 In the ClearQuest Maintenance Tool, select **File > Export Profile**.



2 In the **Export Connection(s)** area, check the connections you want to export in the connection profile.

  The ClearQuest Maintenance Tool lists all the connections available in your ClearQuest system. You can choose to select as many as you want and have all the information stored in one connection profile for export.

3 By default, the connection profile is saved in the ClearQuest installation directory in a file called cqprofile.ini. You can enter a new file name and location, if you want.

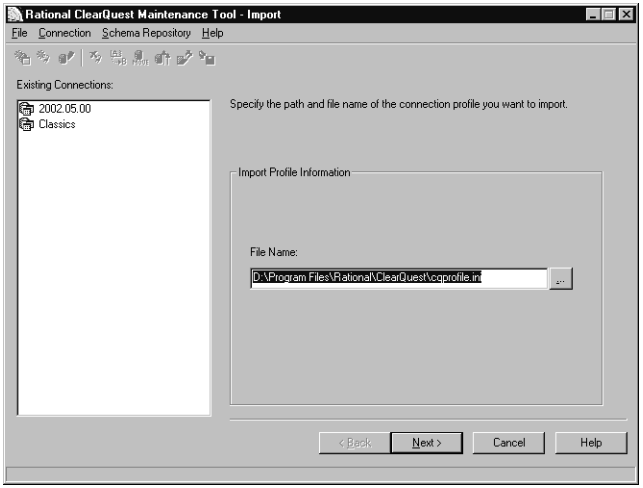
4 Click **Finish**.

Importing a Connection Profile

Note: You cannot import a connection profile if the ClearQuest data code page value of the schema repository does not match the operating system code page of the machine running the Maintenance Tool.

To use a connection profile, import the profile:

1 In the ClearQuest Maintenance Tool, click **File > Import Profile**.



2 For **File Name**, enter the path and file name of the profile you want to import.

If the profile is made up of several connections, the ClearQuest Maintenance Tool lists each of the connections' names and displays their corresponding schema repository properties.

3 If you want to change a connection's name, select a connection and click **Rename.** Enter the new name in the area highlighted.

4 Select the connections you want to import from the profile and click **Finish**.

Moving Schema Repositories

When you move a schema repository to a new location or to a different vendor database management system, ClearQuest copies the schema repository to the new location, locks the old schema repository, and marks the copy in the new location as the active schema repository.

Note: Use only ClearQuest tools to move and manipulate ClearQuest databases. ClearQuest 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. If this happens, you must update and edit the schema repository properties for the connection.

Preparing to Move Schema Repositories

To prepare to move a schema repository to a new location or to a different database management system:

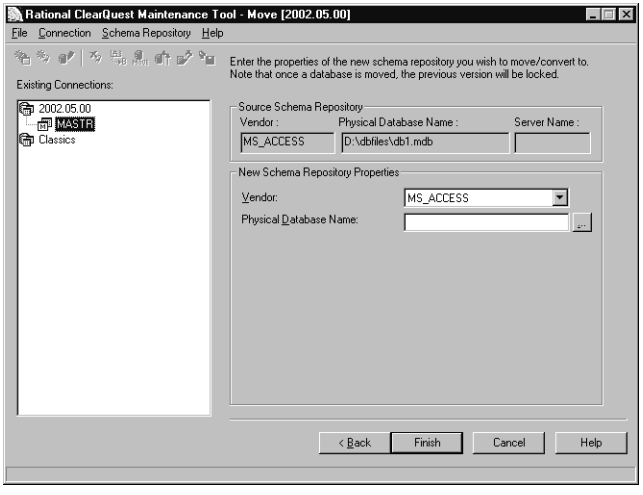
1. Create a new empty database in the new location using your vendor tool (you can create Microsoft Access and SQL Anywhere databases from within the ClearQuest Maintenance Tool). See the *Installation Guide* for Rational Server Products for Windows and the *Installation Guide* for ClearQuest Product Family for UNIX for configuring vendor 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 ClearQuest databases, even if you are moving only your schema repository, or only one user database. This will prevent data loss due to unforeseen circumstances, such as network failure, that might occur during the move.

Moving a Schema Repository

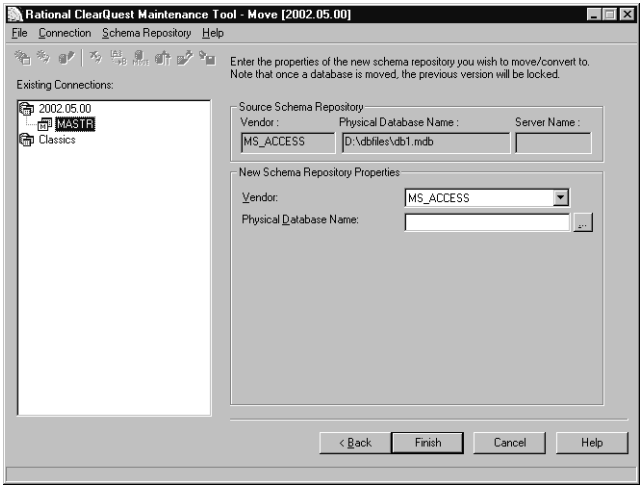
To move a schema repository, follow these steps with the ClearQuest Maintenance Tool:

1. In the **Existing Connections** pane, select the schema repository you want to move, then select **Schema Repository > Move**.



2 Enter your **Logon Name** and **Password**, then click **Next**. Keep in mind that databases are locked during the move process.

3 Enter the **New Schema Repository Properties** you want to move. Select a database **Vendor** and enter the required vendor database properties. The required properties vary according to the database vendor you select. For more information, see *Vendor Database Parameters* on page 73.



4 Click **Finish**.

5 Create a new connection profile by exporting information containing the updated schema repository properties. To export profile information using the ClearQuest Maintenance Tool, click **File > Export Profile**.

6 Have all ClearQuest users import the new connection profile with the ClearQuest Maintenance Tool to connect to the new schema repository. Be sure to provide the information they need to connect. To import a profile using the ClearQuest Maintenance Tool, click **File > Import Profile.**

For more information, see *Creating New Connections* on page 55.

Moving a Schema Repository Using Vendor Database Tools

You can move a schema repository using vendor database tools instead of the ClearQuest Maintenance Tool. However, you must then use the ClearQuest Maintenance Tool to update the connection with information about the new physical location of the schema repository. Super User privileges are required to perform this task.

Updating an Existing Connection

If you have a schema repository that has an existing connection, and you move that schema repository using vendor database tools, do the following to update the connection with the ClearQuest Maintenance Tool:

1. In the **Existing Connections** pane, select the connection you want to move or update, then click **Schema Repository > Update Properties > Selected Connection**.
2. In the **Schema Repository Properties** area, select the **Vendor** of the physical database you are using for the schema repository, and enter the required properties, including the new location of the schema repository. The required properties vary according to the database vendor you select. For more information, see *Vendor Database Parameters* on page 73.
3. Click **Next**. When the update is complete, click **Exit**.
4. Then create a new connection profile by exporting information containing the updated schema repository properties. To export profile information using the ClearQuest Maintenance Tool, click **File > Export Profile**.
5. Have all ClearQuest users import the new connection profile with the ClearQuest Maintenance Tool to connect to the new schema repository. Be sure to provide the information they need to connect. To import profile information using the ClearQuest Maintenance Tool, click **File > Import Profile**.

For more information, see *Creating New Connections* on page 55.

Updating Schema Repositories Without Connections

If your ClearQuest system has an existing schema repository that was created with a ClearQuest prior to version 2001.04.00, you cannot access the schema repository through a connection. However, if you have a need to move the schema repository using other database vendor tools, the ClearQuest Maintenance Tool does provide you with the ability to update its properties without having to associate it with a connection.

To update information about the schema repository's new physical location:

1. Click **Schema Repository > Update Properties > Other.**
2. In the **Schema Repository Properties** area, select the **Vendor** of the physical database you are using for the schema repository, and enter the required properties, including the new location of the schema repository. The required properties vary according to the database vendor you select. For more information, see *Vendor Database Parameters* on page 73.
3. Click **Next** and when the update is complete, click **Exit**.

4 Be sure to provide users with the information they need to connect to the schema repository with the applicable version of ClearQuest.

Upgrading Databases to New Feature Levels of ClearQuest

You can upgrade schema repositories and user databases to new feature levels of ClearQuest using the ClearQuest Maintenance Tool. The feature level defines what features the schema repository and user database supports.

However, before using the ClearQuest Maintenance Tool to upgrade your schema repository or user databases to new feature levels of ClearQuest, see the *Upgrade Guide* for Rational Suite for Windows and the *Installation Guide* for ClearQuest Product Family for UNIX. The following instructions assume you have reviewed the detailed guidelines in this documentation.

Upgrading from SQL Anywhere 5.0 to 8.0

If you have databases that were created using SQL Anywhere 5.5, you do not need to upgrade these databases immediately when you upgrade to a new version of ClearQuest that uses SQL Anywhere 8.0. ClearQuest allows databases created with SQL Anywhere 5.5 and 8.0 to coexist. However, you should create all new databases with SQL Anywhere 8.0, and over time migrate the older databases from 5.5 to 8.0.

Upgrading an Existing Connection

To upgrade a schema repository with its associated databases for an existing connection, do the following using the ClearQuest Maintenance Tool:

1 In the **Existing Connections** pane, select the connection you want to upgrade, then click **Schema Repository > Upgrade > Selected Connection**.

2 Verify the **Schema Repository Properties** of the schema repository you want to upgrade.

3 Click **Next**.

4 When asked if you want to upgrade the user databases associated with the schema repository, click **Yes** or **No**. Keep in mind that user databases are locked during the upgrade process.

5 In the New Schema Repository Properties area, enter the properties of the new **Vendor** database that will contain the upgraded schema repository and click **Next**.

6 If you chose to upgrade user databases, in the New User Database area, enter the properties of the new **Vendor** database that will contain the upgraded user database.

7 If you are upgrading more than one user database, click **Next** and repeat Step 6. If not, click **Finish**.

8 Create a new connection profile by exporting information containing the updated schema repository properties. To export profile information using the ClearQuest Maintenance Tool, click **File > Export Profile**.

9 Have all ClearQuest users import the new connection profile with the ClearQuest Maintenance Tool to connect to the new schema repository. Be sure to provide the information they need to connect. To import profile information using the ClearQuest Maintenance Tool, click **File > Import Profile**.

For more information, see *Creating New Connections* on page 55.

Upgrading Schema Repositories and User Databases Without Connections

If your ClearQuest system has an existing schema repository that was created with a ClearQuest version prior to 2001.04.00, you will not be able to access the schema repository through a connection. However, the ClearQuest Maintenance Tool does provide you with the ability to upgrade its properties and associated user databases without having to associate it with a connection.

1 In the ClearQuest Maintenance Tool, click **Schema Repository > Upgrade > Other**.

2 In the Schema Repository Properties area, enter the properties of the schema repository you want to upgrade.

3 Click **Next**.

4 When asked if you want to upgrade the user databases associated with the schema repository, select **Yes** or **No**. Keep in mind that user databases are locked during the upgrade process.

5 In the **Login Information** area, enter your **Logon Name** and **Password**, then click **Next**.

6 If you chose to upgrade user databases, select the user databases to upgrade and click **Next**.

7 In the **New Schema Repository Properties** area, enter the properties of the new vendor database that will contain the upgraded schema repository and click **Next**.

8 If you chose to upgrade a user database, in the New User Database area, enter the properties of the new vendor database that will contain the upgraded user database.

9 If you are upgrading more than one user database, click **Next** and repeat Step 8. If not, click **Finish**.

10 Create a new connection profile by exporting information containing the updated schema repository properties. To export profile information using the ClearQuest Maintenance Tool, click **File > Export Profile**.

11 Have all ClearQuest users import the new connection profile with the ClearQuest Maintenance Tool to connect to the new schema repository. Be sure to provide the information they need to connect. To import profile information using the ClearQuest Maintenance Tool, click **File > Import Profile**.

For more information, see *Creating New Connections* on page 55.

Moving and Deleting User Databases

You can use the ClearQuest Designer to move a ClearQuest user database to a new location or to a different database management system.

Preparing to Move a User Database

When you move a user database to a new location or to a different vendor database management system, ClearQuest copies the user database to the new location, locks the old user database, and marks the copy in the new location as the active user database.

To prepare to move a user database to a new location or to a different vendor's database management system:

1 Create a new empty database in the new location using your vendor tool (you can create Microsoft Access and SQL Anywhere databases from within ClearQuest Designer). See the *Installation Guide* for Rational Server Products 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 only your schema repository, or only one user database.

Note: You must back up *all* of your ClearQuest databases (the schema repository and all associated user databases) to prevent data loss due to unforeseen circumstances, such as network failure, that might occur during the move.

Moving a User Database

Use ClearQuest Designer to move a user database to a new location or to a different vendor database management system.

The instructions that follow show how to use ClearQuest Designer to move a user database called SAMPL to a new database vendor. You can use the same procedure to move a 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, see *Updating the Properties of a User Database* on page 70.

To move a user database:

1 Complete the steps in *Preparing to Move a User Database* on page 68.

2 Start ClearQuest Designer by clicking **Start > Programs > Rational Software > Rational ClearQuest > ClearQuest Designer**.

3 If the **Open Schema** dialog box appears, click **Cancel**.

4 Click **Database > Move User Database**.

5 In the Move User Database dialog box, select a **Logical Database Name** and click **Properties**.

Note: If you click **OK**, you will only update any changes you made to the **Comment** text box; you will *not* be moving the database.

6 In the Database Properties dialog box, select the new **Vendor** for the database and fill in the database properties, including whether the database is a **Production Database** or a **Test Database**.

7 When you finish, click **Move**. The Moving Database dialog box displays the Status of the database move.

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**, then continue to Step 8.

8 Click **OK** to close the Moving Database dialog box, then click **OK** in the database update message.

9 Click **OK** to close the Move User Database dialog box.

Updating the Properties of a User Database

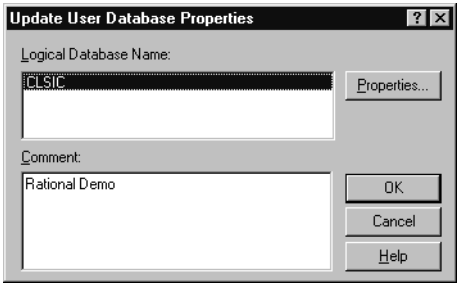
This section describes how to use the **Update User Database Properties** menu option to change a selected database's properties if you have moved your user database outside of ClearQuest. (If you have *not* moved your database outside of ClearQuest, follow the steps described in *Preparing to Move a User Database* on page 68 to change properties prior to moving the database.

1 Start ClearQuest Designer by clicking **Start > Programs > Rational Software > Rational ClearQuest > ClearQuest Designer**.

2 If the **Open Schema** dialog box appears, click **Cancel**.

3 Click **Database > Update User Database Properties**.

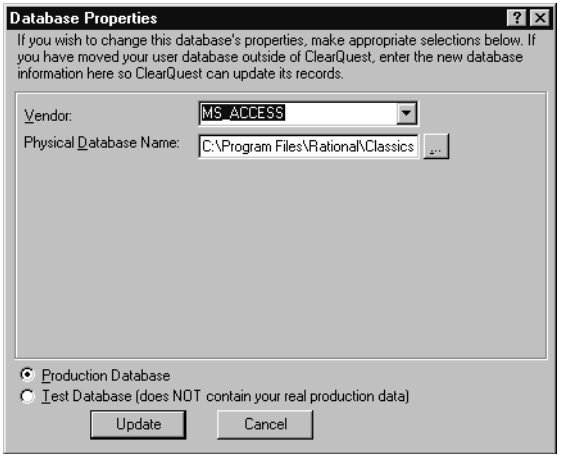
4 In the Update User Database Properties dialog box, select a **Logical Database Name**.



To update the **Comment** field only, enter your comments and click **OK**.

5 To modify the database properties, click **Properties.**

6 In the Database Properties dialog box, edit the properties you want. For more information, see *Vendor Database Parameters* on page 73.



7 When you are done, click **Update** to save your changes.

Deleting User 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.

For more information, see *Restoring a Deleted Database* on page 72.

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.

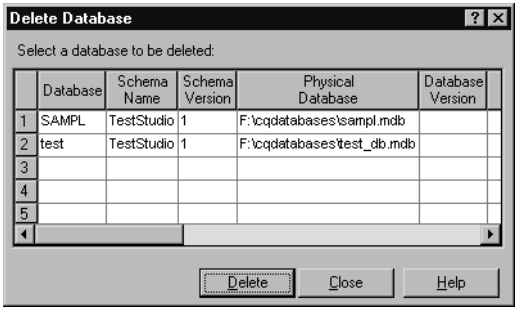
ClearQuest does not allow you to delete a schema repository.

To delete a user database from ClearQuest:

1 Start ClearQuest Designer by clicking **Start > Programs > Rational Software > Rational ClearQuest > ClearQuest Designer**.

2 If the **Open Schema** dialog box appears, click **Cancel**.

3 Select **Database > Delete Database**.



4 In the Delete Database dialog box, select the database you want to delete and click **Delete**.

After you delete a user database from ClearQuest (remove the link between the database and the schema repository), you can physically delete the user database; however, if you do this you will not be able to restore the deleted database. To physically delete an Oracle or SQL Server database, use your database vendor tools. For Microsoft Access and SQL Anywhere databases, you can manually remove the database files.

Restoring a Deleted Database

Note: Restoring a deleted database is possible only if the physical database still exists. You cannot restore a deleted database if you used your database vendor's tools to delete the physical database.

When you restore a deleted user database, you restore the link between the user database and the schema repository. 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.

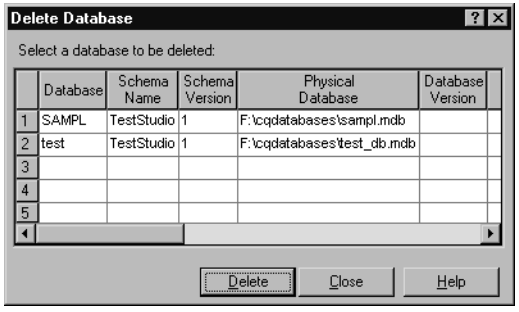
Restoring a deleted database should be done as soon as possible after you delete the database to avoid permanently losing its link to the schema repository. If you delete the schema version associated with the database or if you upgrade to a new version of ClearQuest, you cannot restore the link.

To restore a deleted user database:

1 Start ClearQuest Designer by clicking **Start > Programs > Rational Software > Rational ClearQuest > ClearQuest Designer**.

2 If the **Open Schema** dialog box appears, click **Cancel**.

3 Click **Database > Undelete Database** to open the Undelete Database dialog box.



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

Viewing Database Properties

To view the properties of a user database, do the following:

1. Start ClearQuest Designer by clicking **Start > Programs > Rational Software > Rational ClearQuest > ClearQuest Designer**.
2. If the **Open Schema** dialog box appears, click **Cancel**.
3. Click **Database > View Database Properties**.
4. In the Database Property dialog box, select the database you want and click **Properties**.

   The Database Properties window is read-only. To move or edit database properties, see *Moving and Deleting User Databases* on page 68 or *Updating the Properties of a User Database* on page 70.
5. Click **OK** to close the Database Properties window.

Vendor Database Parameters

When you create a ClearQuest schema repository or user database, you are prompted to enter the parameters for the vendor database that you have created for use with ClearQuest.

Keep in mind that when you specify database parameters, case sensitivity may vary per vendor database. In particular, it is mandatory that your database sensitivity settings are consistent in these two scenarios:

- When you are moving a database from one database vendor type to another.
- When you are using ClearQuest MultiSite and are using multiple synchronized databases created from different database vendor products.

Database Vendor	Default Setting	Possible Settings	Comments
Microsoft Access	Case-insensitive	Case-insensitive	Microsoft Access databases are case-insensitive.
SQL Anywhere	Case-insensitive	Case-insensitive, Case-sensitive	All databases are created as case-insensitive, this setting can be changed to make it case-sensitive.

Database Vendor	Default Setting	Possible Settings	Comments
DB2	Case-sensitive	Case-sensitive	Case sensitivity is applicable to data only and not database object names.
SQL Server	Case-insensitive, Case-sensitive	Case-insensitive, Case-sensitive	Case sensitivity can be decided either by using the installation time option or database creation time option. **Note:** Case sensitivity applies to database object names as well as data.
Oracle	Case-sensitive	Case-sensitive	The case sensitivity is applicable to data only and not database object names.

Microsoft Access

Note: ClearQuest Web and ClearQuest MultiSite do not support Microsoft Access databases.

- In the **Physical Database Name** box, enter the name for the schema repository and the full path to the schema repository file using a UNC style path. For example:

  ```
  \\DevServer\ProjectShare\CQ_DBS\schema_repo.mdb
  ```

  You can browse to the directory containing the database. Be sure to browse using the **Network Neighborhood** to preserve the UNC style path name.

SQL Anywhere

- In the **Physical Database Name** box, enter the name for the schema repository and the full path to the schema repository file using a UNC style path. For example:

  ```
  \\DevServer\ProjectShare\CQ_DBS\schema_repo.mdb
  ```

  You can browse to the directory containing the database by using **Network Neighborhood**, which preserves the UNC style path name.
- Enter the name of the database server in the **Database Server Name** box.
- Select the protocols used to communicate with the SQL Anywhere server.
- Enter the database host name in the **Host Name** box. The host name is required. The database host machine must be visible to the client machines using network share

and using one or more of the communication protocols. For TCP/IP, this can also be the IP address.

- Enter the version of SQL Anywhere being used in the **Connect Options** box. For SQL Anywhere 5.0, enter Server_Ver=5.0. For SQL Anywhere 8.0, enter Server_Ver=8.0. If you leave the field blank, the ClearQuest Maintenance Tool will default to SQL Anywhere 5.0.

  Note: With ClearQuest version 2003.06.00, you can only create new schema repositories with SQL Anywhere version 8.0.

SQL Server

- In the **Physical Database Name** box, enter the database name for SQL Server schema repository.
- Enter the name of the database server in the **Database Server Name** box.

  Note: Each ClearQuest native client must be able to access this server host name exactly as it appears in this entry.
- Enter the database owner (DBO or Administrator) username in the **Administrator Name** box.
- Enter the password for the database owner (DBO or Administrator) in the **Administrator Password** box.
- Enter the general purpose or read/write username in the **Read/Write User Name** box.
- Enter the password for the general purpose or read/write username in the **Read/Write User Password** box.
- Enter the schema repository or read-only username in the **Read-Only User Name** box.
- Enter the password for the schema repository or read-only username in the **Read-Only User Password** box.

Oracle

- Enter the TNS name (SQL*Net Alias pointing to the Oracle instance) in the **SQL*Net Alias** box.
- Enter the Oracle username you created for the schema respiratory in the **User Name** box.
- Enter the password for the username in the **Password** box.

- The following options need to be typed in the **Connect Options** box, separated by semi-colons. For example,

  ```
  Host=name; SID=name; Server_ver=8.1; Client_ver=8.1;
  Lob_type=LONG or CLOB
  ```

Option	Definition
HOST	The name of the database server host. If your ClearQuest environment includes UNIX and Windows clients, this name is mandatory.
SID	The name of the database instance. If your ClearQuest environment includes UNIX and Windows clients, this name is mandatory. On UNIX, the SID is case-sensitive.
SERVER_VER	Provide the version of the Oracle server you are using. (For Oracle 8i, use 8.1.)
CLIENT_VER	Provide the version of the Oracle client software you are using. (For Oracle 8i, use 8.1.) If ClearQuest client users want to use a different Oracle client version than what is entered here by the administrator, they'll need to override the set options manually. The default setting is 8.1.
LOB_TYPE	Set the type of data you will be using. The valid data types are either LONG or CLOB. If your existing databases default to LONG, do not change them to CLOB. Databases already implemented with the LONG LOB_TYPE property are incompatible with CLOB. However, there are limitations using the CLOB data type. For more information on restrictions and guidelines, consult the *Release Notes* for Rational ClearQuest.

Note: All ClearQuest users must use the same Alias or Net Service Name (**TNSname**), Database Instance (system ID or **SID**), **TCP/IP** protocol, and **Host** name (Oracle server host).

DB2

- Enter the database alias name (alias pointing to the DB2 database) for the schema repository in the **Database Alias** box.
- Enter the DB2 username created for the schema repository in the **User Name** box.

- Enter the password for the user name in the **Password** box.

  Note: With DB2, ClearQuest can point to a single, physical database server provided that each ClearQuest database uses a unique DB2 login name. You do not have to create an alias for each DB2 client.

Working with ClearQuest Schemas

4

A schema in Rational ClearQuest is a complete description of the process model for one type of change request (the *metadata* for that process model). It includes a description of the states and actions of the model, the structure of the data, *hook code*, forms, reports, and queries.

To modify how a type of change request is handled, you must understand how to check out, test and check in schema, and how to upgrade databases to reflect the modifications.

The topics covered include:

- *Procedures for Modifying a Schema* on page 80
- *Creating a Test Database* on page 80
- *Checking Out a Schema* on page 82
- *Creating a New Schema from the Blank Schema* on page 84
- *Selecting a Scripting Language* on page 85
- *Validating Schema Changes* on page 86
- *Setting the Test Database for the Schema* on page 87
- *Checking In a Schema* on page 88
- *Upgrading a User Database* on page 89
- *Saving Work in Progress* on page 90
- *Deleting a Schema or Schema Version* on page 91
- *Changing a Database to a Different Schema* on page 92

Later you will learn how to use the ClearQuest Designer to modify the individual elements of a schema (the process model, data structure, hook code, and so on) and how to change a schema by applying *packages* (bundles of schema components that add a complex feature to a schema).

Note: To work with schemas, you need Schema Designer or Super User privileges. To learn how to set user privileges, see Chapter 8, *Administering Users*.

ClearQuest includes several predefined schemas that provide common workflow models. For a list of predefined schemas and packages, see *ClearQuest Schemas and Packages* on page 277.

Procedures for Modifying a Schema

To modify schema and put the modifications into effect, or to create a new schema, use ClearQuest Designer to perform the following actions:

1. Create a test database and associate it with the schema you want to customize.
2. Check out the schema you want to customize from the schema repository, or create a new schema.
3. Select the scripting language to use for hook code.
4. Modify the schema, either by using ClearQuest Designer to change elements of the schema, or by applying packages.
5. Validate the customized schema using the Validate schema command in ClearQuest Designer.
6. *Set* the test database for the schema (that is, designate that the test database you created earlier be used to test the schema you checked out).
7. Use the test database to test the customized schema.
8. Save work in progress.
9. Check in the customized schema into the schema.
10. Upgrade the user database with the modified schema.

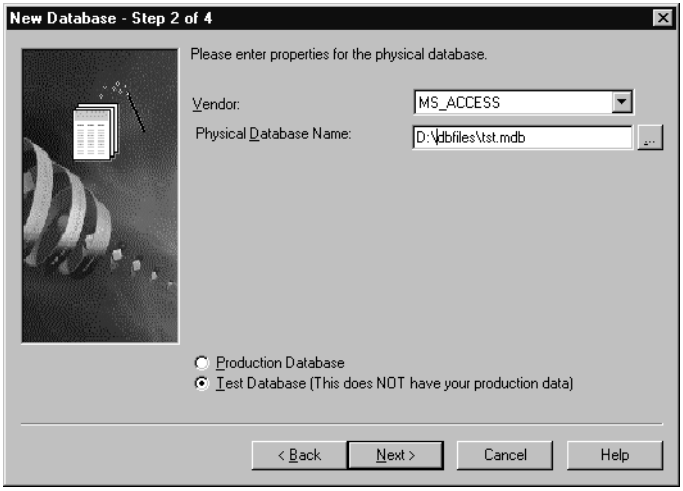
Creating a Test Database

A *test database* is a temporary database that you create when you are developing a new schema or a new version of an existing schema. With the test database you can make sure that the new schema performs as expected, without running the risk of damaging production data in a user database.

The procedure for creating a test database is the same as the procedure for creating a user database.

To create a test database:

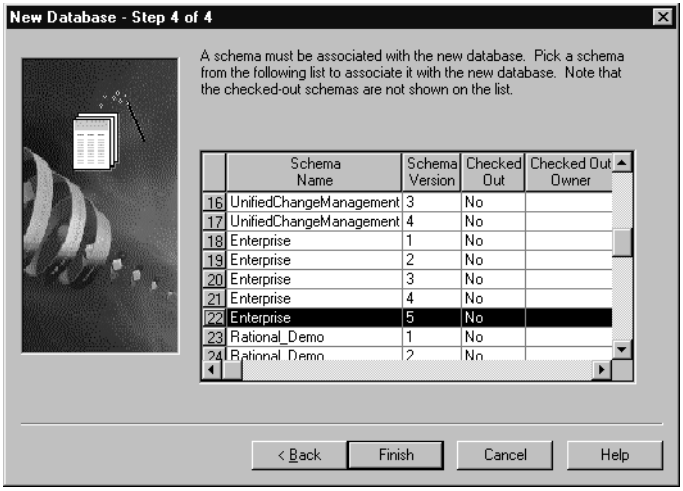
1. If ClearQuest Designer is not open, start ClearQuest Designer, and when the Open Schema dialog box appears, click **Cancel**. If ClearQuest Designer is already open, begin with Step 2.
2. In ClearQuest Designer, click **Database > New Database**.
3. The New Database Wizard prompts you through the process of creating a database. Give the database a name between one and five characters that you can recognize as a test database.
4. In Step 2 of 4, select **Test Database** near the bottom of the screen.



   For information about configuring database properties, see *Vendor Database Parameters* on page 73.

5. In Step 4 of 4, select the schema and schema version that you are modifying and associate this with the test database you are creating.

Note: If your test database will be accessed by clients running versions of ClearQuest prior to version 2003.06.00, you must apply the CharacterSetValidation package and set the ClearQuest data code page value for the test database. For more information, see *How to Upgrade Packages* on page 140 and *Setting the ClearQuest Data Code Page Value* on page 35.



You are now ready to check out the schema and modify it. Later, when you have finished modifying the schema and are ready to test it, you set this test database and use it for your testing. For more information, see *Setting the Test Database for the Schema* on page 87.

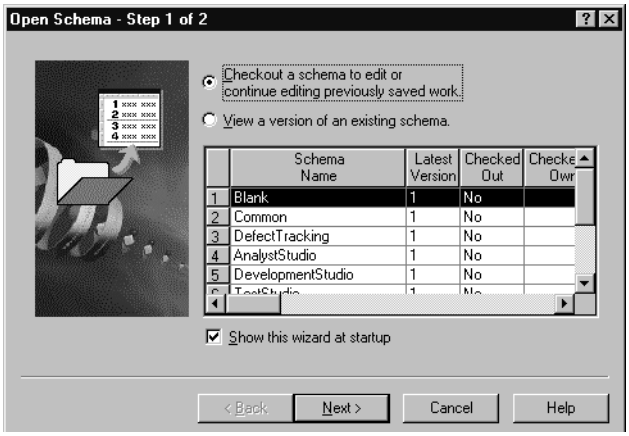
Checking Out a Schema

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.

Note: You cannot check out a schema if the ClearQuest data code page value of the schema repository does not match the operating system code page of the machine running the ClearQuest Designer. For more information about the data code page value, see *About the ClearQuest Data Code Page* on page 28.

To check out a schema for editing in ClearQuest Designer:

1. Open the Open Schema dialog box. The Open Schema dialog box appears each time you log on if you have previously checked the **Show this wizard at startup** option. Or you can display the Open Schema dialog box at any time by clicking **File > Open Schema**.
2. In the Open Schema dialog box, make sure **Check out a schema to edit** is selected.
3. Select a schema and click **Next**.



4 Enter comments in the Comment text box (optional) and click **Finish**. ClearQuest Designer opens the Workspace and displays the schema and its structure.

Note: If the ClearQuest data code page value for the schema repository is unset, you can enter only ASCII data into the schema.

Opening a Schema for Viewing Only

If you want to review the contents of a schema without making any changes, you do not have to check out the schema. This is called opening a schema for viewing only.

To open a schema for viewing only in ClearQuest Designer:

1 Open the Open Schema dialog box. The Open Schema dialog box appears each time you log on if you have previously checked **Show this wizard at startup**. Or you can access the Open Schema dialog box at any time by clicking **File > Open Schema**.

2 In the Open Schema dialog box, click **View a version of an existing schema**.

3 Select a schema version to view and 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 Designer closes the schema you are viewing.

To close a schema that you are viewing:

- Click **File > Close Work**.

Creating a New Schema from the Blank Schema

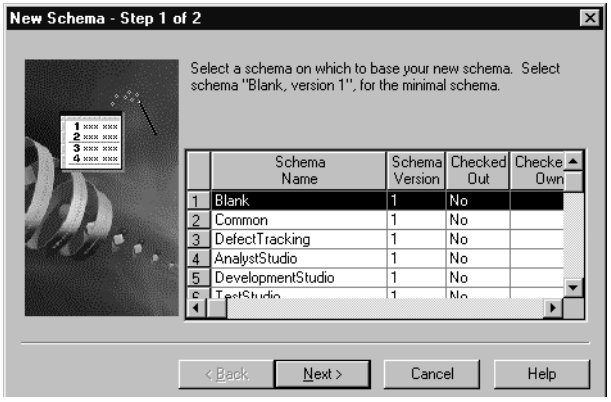
If you want to create a new schema that differs substantially from the predefined schemas, you can start by checking out and editing the Blank schema.

The Blank schema contains only system fields. The system fields are needed by ClearQuest and cannot be edited. Therefore, when you start with the Blank schema you create a completely new schema by adding all of the fields, components of the state model, and other pieces of your new schema.

To create a new schema from the Blank schema:

1 In ClearQuest Designer, click **File > New Schema**.

2 In the New Schema dialog box, select the **Blank schema** and click **Next**.



3 Enter a Schema Name, Comments (optional), and 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 (optional), and click **OK**.

Selecting a Scripting Language

Hooks are scripts or triggers that can be run under a designated set of conditions to perform a specific task.

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.

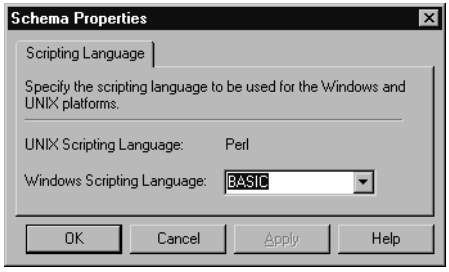
By default, ClearQuest runs scripts in BASIC on Windows and in Perl on UNIX.

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 in ClearQuest Designer:

1 In the Workspace, double-click **Schema Properties**.

2 In the Schema Properties dialog box, select **BASIC** or **PERL**.



For more information, see Chapter 10, *Using Hooks to Customize Your Workflow* to customize your workflow. For information on the scripting languages, see the following resources:

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

Validating Schema Changes

After you modify a schema, you must validate it before you check it into the schema repository.

In fact, when you try to check in a schema, ClearQuest automatically validates it and prevents the check in if there are any validation errors.

During validation, ClearQuest performs a number of tests on the schema, including validating that you have:

- Entered a unique name for each field and action.
- Assigned to each field a datatype and a behavior for each state of the record type.
- Supplied a record type to the reference_to property of each field with REFERENCE or REFERENCE_LIST datatype.
- Designated a source state and a destination state for all state transition actions.
- Defined a unique key for all stateless record types.
- Used SQL reserved words correctly.

If you have applied any packages to the schema, ClearQuest also runs the test with unique validation rules that apply to those packages.

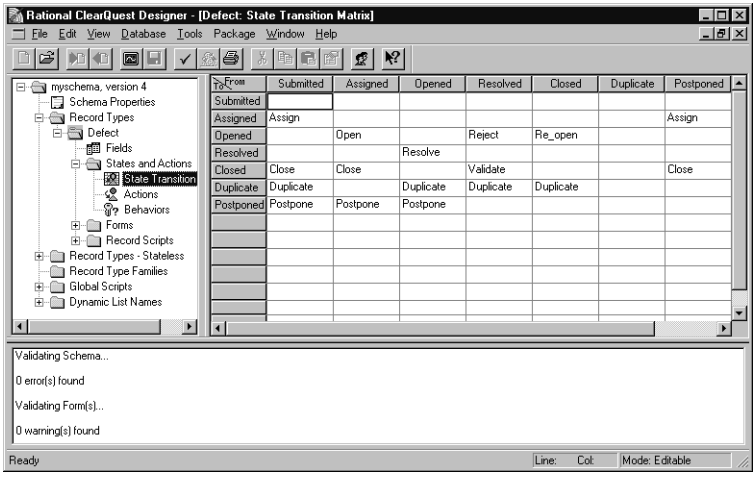
Note: Hook code can introduce subtle errors at runtime if not written correctly. To test hooks, click **Hooks > Compile** in the Scripts Editor and debug your hook code as needed. Also be aware that validation does not catch errors in hook code, and does not guarantee that a schema functions as you want. 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.

For more information, see *Testing the Schema with a Test Database* on page 88.

To validate a schema:

1 In ClearQuest Designer, click **File > Validate**.

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



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

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. For more information, see *Undoing a Schema Checkout* on page 89.

Setting the Test Database for the Schema

Before checking out a schema for editing you should have created a test database and associated it with the schema. For more information, see *Creating a Test Database* on page 80.

After you check out, modify, and validate the schema, you must *set* the test database for the schema (that is, designate that the test database you created earlier be used to test the schema you have checked out).

With the test database, you will be able to test the modifications you have made to the schema. See Testing the schema with a test database on *Testing the Schema with a Test Database* on page 88.

To set the test database for a schema, do the following:

1 In ClearQuest Designer, check out the schema you want to customize, if it is not already checked out.

2 Click **Database > Set Test Database** and select the test database you created to use with this schema.

Testing the Schema with a Test Database

Before you can test schema customizations, you must first create a test database and set the test database for the schema. For more information, see *Creating a Test Database* on page 80 and *Setting the Test Database for the Schema* on page 87.

To test the schema with a test database using ClearQuest Designer, do the following:

1. After modifying a schema in ClearQuest Designer, click **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. For more information, see *Validating Schema Changes* on page 86.

   ClearQuest then automatically updates your test database to the new schema version.

2. On Windows, ClearQuest automatically starts the ClearQuest client so you can test the schema to see if it works as expected.

   Note: To test the schema from the UNIX or Web clients, you must log into the test database manually. 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.

After you click **File > Test Work** during a schema editing session, you cannot undo the schema checkout.

You must keep your test database up to date with your latest schema changes. If you attempt to check in a schema without testing your work, ClearQuest prompts you to test your work first.

If you have a great deal of design, development, and modification work to do, you should save your work often. Click **File > Save Work** to save the changes without checking in the schema. This *save work* activity does not validate your changes and you cannot upgrade the test database with this saved version of the schema.

Checking In a Schema

After editing and testing a schema, you must check in 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.

When 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.

Checking in a schema does not affect the existing user databases. You must apply the new version of the schema to the user databases to have your changes take effect. See *Upgrading a User Database* on page 89.

To check in a schema:

1 In ClearQuest Designer, click **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.

When the schema passes validation, ClearQuest Designer displays the Check In dialog box with your original checkout Comments displayed. Edit the comments (optional), and click **OK**.

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:

- In ClearQuest Designer, click **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 lets you undo the schema checkout. You then need to create a new test database.

Upgrading a User Database

After checking in a schema, you may upgrade the user databases that used an earlier version of the schema. You do not have to perform this upgrade immediately, but until you do the changes to the schema are not available to the users.

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 already logged in.

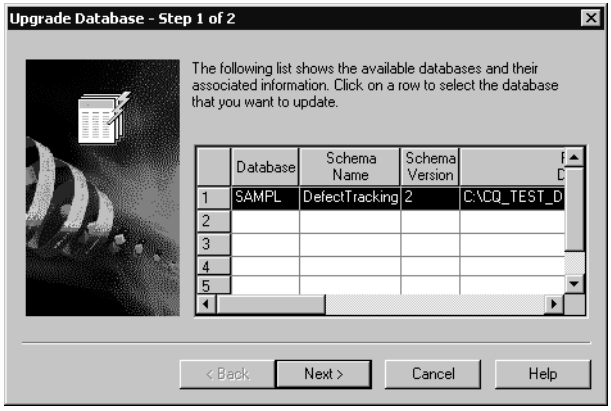
Keep in mind the following restriction when you apply the schema changes to a user database:

- After 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.

Warning: Upgrading the user database is a nonreversible process. Before applying schema changes to a user database, back up the user database.

To apply schema changes to a user database:

1 In ClearQuest Designer, click **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 regarding databases, see Chapter 3, *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 In ClearQuest Designer, click **File > Save Work**.

2 Click **File > Close Work** or **Exit** to end the editing session.

To reopen the schema and continue editing:

1 Open the Open Schema dialog box. If you are logging in to ClearQuest Designer, the Open Schema dialog box automatically appears. Or at any time from within ClearQuest Designer, click **File > Open Schema**.

2 In the Open Schema dialog box, click **Open a schema** for editing or continue editing previously saved work and select the schema version on which you previously worked.

   ClearQuest Designer opens the schema version for editing.

Note: Before checking the schema into the schema repository, be sure to validate the schema and test your work. For more information, see *Validating Schema Changes* on page 86, *Testing the Schema with a Test Database* on page 88, and *Checking In a Schema* on page 88.

Deleting a Schema or Schema Version

Warning: 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, click:

   - **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.

Changing a Database to a Different Schema

After 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.

However, you can move the *data* in an existing database to a new database that has been associated with a new schema.

To move the data in an existing user database to a database associated with a new schema, you must first create a new database and associate it with the new 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 Chapter 3, *Managing Databases*.
2. Create a new user database and associate it with the new schema.

   In ClearQuest Designer, click **Database > New Database** to create a new user database. ClearQuest Designer prompts you to associate the new database with a schema. Associate the new database with the new schema.
3. Use the ClearQuest Export Tool to export the data from your current user database. For more information, see Chapter 13, *Importing and Exporting Data*.
4. Use the ClearQuest Import Tool to import the data into the new database. For more information, see Chapter 13, *Importing and Exporting Data*.

You can use the **cqload** command line utility, along with the ClearQuest Import and Export Tools, to copy an entire schema or part of a schema into another schema repository. For more information, see Chapter 13, *Importing and Exporting Data* and Appendix E, *Command Line Utilities*.

Customizing a Schema

5

To modify how change requests are handled, you need to use Rational ClearQuest Designer to customize the records and fields of a schema.

The topics covered include:

- *Overview of Customizing a Schema* on page 93.
- *Working with Record Types* on page 94.
- *Working with Fields* on page 100.
- *Overview of State Models* on page 113.
- *Customizing State Models with the State Transition Matrix* on page 115.
- *Working with Actions and Action Types* on page 119.

Overview of Customizing a Schema

When you want to modify a schema, follow the standard procedure described in Chapter 4, *Working with ClearQuest Schemas*. This includes checking out the schema, modifying the schema with the ClearQuest Designer or by applying a package, validating the schema, testing the schema, checking in the schema, and upgrading user databases.

Modifying the schema with the ClearQuest Designer refers to a process that includes:

1. Defining the record types you need. If you have several change management processes that follow different process models (say working on defects and managing new feature requests) you need to create multiple record types.
2. Defining the fields that store the data for each record type and the behavior of those fields.
3. Defining the states in the state model for each record type.
4. Defining the actions of each state model.
5. Defining forms to add new records and work with existing records.
6. Adding *hooks* (scripts) to customize fields and to customize actions.

In this material we will look at the first two steps, defining record types and defining fields and their behavior, and we will also look at adding hooks to customize fields. The other steps will be discussed in subsequent sections.

Working with Record Types

In ClearQuest, a *record type* is the layout or format for a particular type of change request. It is roughly analogous to a table in a relational database. Each record type defines the data that can be collected for one type of change request. The information about an *individual* change request is called a *record*, and an individual piece of data about a change request is called a *field*.

Each record type is associated with its own state model, forms, and hooks, which control the collection and viewing of data for that type of change request.

State-Based and Stateless Record Types

ClearQuest supports both state-based and stateless record types.

A *state-based* record type is one that moves through a series of statuses or *states* (for example, Submitted, Assigned, and Resolved), as a result of actions performed by ClearQuest users.

A *stateless* record type is one that holds data, but does not change states. Examples of stateless record types are users, projects, and customers. The only actions you can perform on stateless record types are Submit, Modify, Delete, and Import.

State-based records can reference one or more stateless records. For example, a ClearQuest user might assign defects (a state-based record type) to a project (a stateless record type).

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

How Many Record Types?

A schema can contain more than one record type. For example, one schema might have separate record types for Software Enhancements and Hardware Enhancements. Or it might have different record types for Issues, Problem Reports, Change Requests, Defects, and Enhancement Requests.

Generally speaking, you should create separate record types when change request types have a different process model or require the tracking of different data. For example, if your organization has different process models for software enhancements and hardware enhancements, you should create two separate record types.

Alternatively, if the process model is the same for software and hardware enhancements, you should create a single Enhancements record type with a field to designate the type of enhancement.

You should carefully consider which record types to create. More record types allow you to capture more variations in process models. However, creating too many record types complicates administration and makes it harder to build queries and reports that cover large numbers of change requests. You also want to think ahead; if two types of change requests have the same process model today but you anticipate that they will diverge in the future, then it will be easier to create two record types in the beginning than to try to split them up later.

You should consider some of the same issues that come up in designing relational databases (or perhaps obtain assistance from a database administrator familiar with these issues). For example, instead of including *submitter*, *submitter's e-mail address*, and *submitter's phone number* in the Defects record type, you may want to include only *submitter* in the Defects record type and create an additional *Submitters* record type. This would allow users to avoid entering their e-mail address and phone number every time they submit a defect. You would then use a REFERENCE field to create a link between the Defect and Submitter record types so that you could display the submitter's e-mail address and phone number on forms and reports. See *Linking Records to Create a Parent/Child Hierarchy* on page 110.

Adding a New Record Type

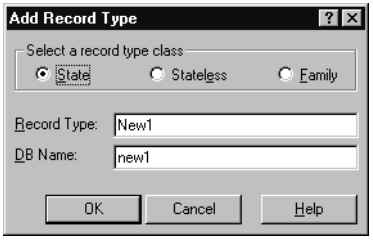
You can add new record types to a schema. When you create a new record type you make it either state-based or stateless; after it is created you cannot change it from one to the other.

To add a new record type:

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

2 In the Add Record Type dialog box, select the class for the record type, either **State** or **Stateless**.

3 In the Record Type box, type a name for the record type.



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

If you are adding a stateless record type, you must select one or more of its fields to be the unique key. For more information, see *Selecting a Unique Key for a Stateless Record Type* on page 96**.**

4 Click **OK**.

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. For more information, see *Selecting a Default Record Type for a Schema* on page 97**.**

Selecting a Unique Key for a Stateless Record Type

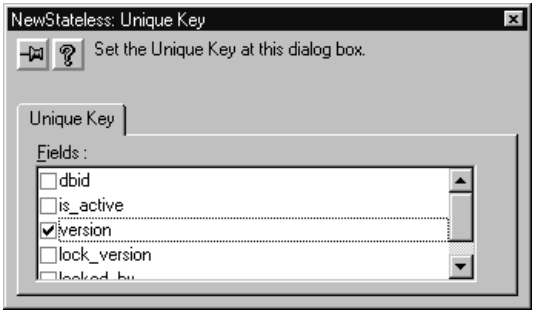
When you add a new stateless record type to a schema, you must select one or more of its fields to be the unique key. ClearQuest uses the unique key to keep track of the individual change requests.

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 fields of the new unique key.

To select the unique key for a stateless record type:

1 In the Workspace of ClearQuest Designer, expand the **Record Types – Stateless** folder and expand the NewStateless folder. Double-click **Unique Key**.

2 In the NewStateless: Unique Key dialog box**,** check one or more field names to be used as the unique key.



3 Close the NewStateless: Unique Key dialog box**.**

Selecting a Default Record Type for a Schema

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.

Making a Record Type the Default

In the Workspace of ClearQuest Designer, click the **Record Types** or **Record Types – Stateless** folder, right-click the record type you want to make the default, 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 lets ClearQuest users run a single 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.
- Because record types and record type families appear in the same window when ClearQuest users click **Query > New Query**, use a naming convention that helps users distinguish individual record types from record type families.

To create a record type family, you will perform the following tasks:

- *Adding a Record Type Family* on page 97.
- *Creating Common Fields* on page 99.

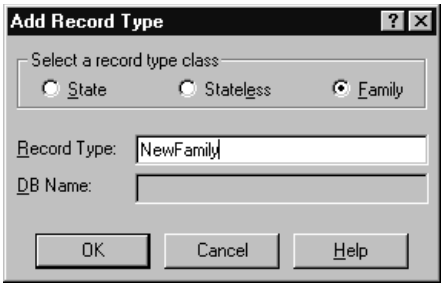
You can also remove members from a record type family, and rename or remove the record type family.

Adding a Record Type Family

To create a new record type family for a schema:

1. Click **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** box, enter a name for the record type family and click **OK**. ClearQuest Designer adds the new family to the Record Type Families folder.

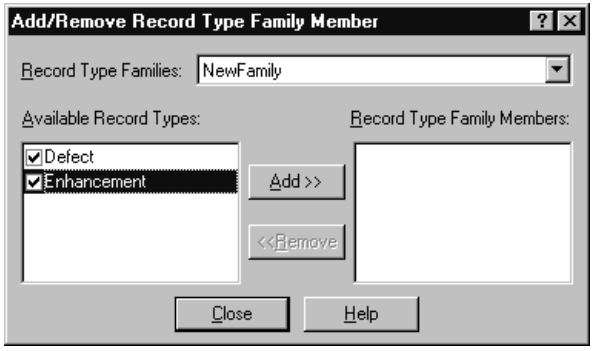


Note: ClearQuest uses the DB Name for the table. By default, it is the same as the record type family name.

Adding Members to a Record Type Family

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

1 Click **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 and click **Add** and then click **Close**.

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

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

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 in both record types (perhaps short string). To learn how to add fields to a record type and to define their data type, see *Working with Fields* on page 100.

Removing Members from a Record Type Family

To remove record types from a record type family:

1. Click **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** and then 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. Click **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** list.
4. Enter a new name in the **New Name** 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:

- Stateless system record types: ratl_replicas, history, attachments, groups, and users.
- Record types added by read-only packages. For more information, see Chapter 6, *Applying Packages*.
- Default record type. If you want to delete the current 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. Click **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 click **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:

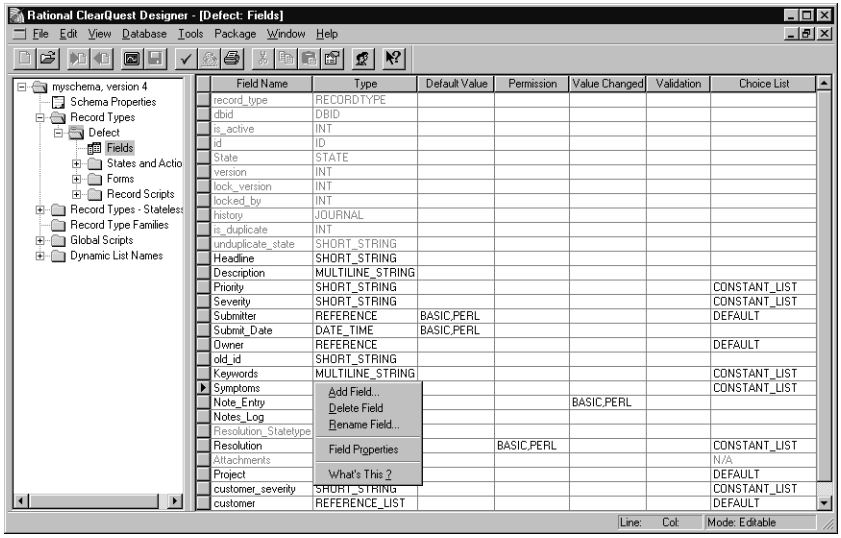
- Add and remove fields.
- Define the behavior of a field.
- Add Help text to a field.
- Use fields to link records.
- Customize a field by adding hooks.

Each record type has a Fields grid that shows the fields associated with that record type.

To display the Fields grid:

1. Expand a record type in the Workspace and double-click Fields.
2. Each field is displayed in a row, and its properties are displayed in the columns of that row.

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



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

Keep in mind a few key points about adding and modifying fields:

- 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 that version of the schema to the user database.
- After 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 to a record type:

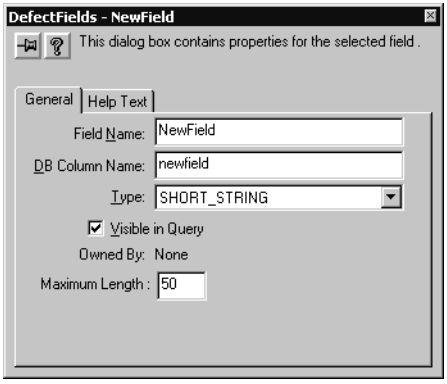
1. Add the field to the Fields grid.
2. Associate the new field with a form control to make the field available to ClearQuest client users. For more information, see *Adding Controls to a Form* on page 148.

3 Test the schema, check the schema back into the schema repository, and apply the schema changes to the user database. For more information, see Chapter 4, *Working with ClearQuest Schemas*.

To add a new field to the Fields grid:

1 In the Workspace of ClearQuest Designer, expand the **Record Types** folder, then expand the record type you want and double-click **Fields** to display the Fields grid.

2 Click **Edit > Add Field**.



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

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

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

4 Select a field data type in the **Type** box. The data type you select determines the type of values the user can enter in the field. For more information, see *Considerations for Selecting Field Data Types* on page 103.

- 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 which the field will refer. 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 109.

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

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. For more information, see *Adding Controls to a Form* on page 148.

Considerations for Selecting Field Data Types

When you add a field, you must select its data type (see *Adding a New Field*, step 4). The data type you select determines the type of values the user can enter 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.

Data	Description/Comments
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. When a user enters a value in a field of the type SHORT_STRING, ClearQuest automatically trims any leading or trailing spaces.
DBID	Reserved for system fields.
ID	Reserved for system fields.
JOURNAL	Reserved for system fields.
STATE	Reserved for system fields.

You should keep several factors in mind when selecting field types.

Allocate enough time to think through your selections and to test them, because it may be difficult to make changes later on. 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.

For fields of the data type SHORT_STRING, enter a value for Maximum Length that is larger than the largest string you think will occur. A larger Maximum Length causes ClearQuest to reserve more space on the disk drive, but it reduces the chance that ClearQuest will need to truncate entries.

A single sting field (MULTILINE_STRING and SHORT_STRING) is suitable for general free-form information, for example descriptions, comments, and known workarounds. However, use several smaller fields if you might use information in the field for sorting or filtering in a query. For example, instead of one *computing environment* field, you might want to have individual fields for *hardware vendor*, *operating system*, and *OS version*.

Use several smaller fields if you want to enforce content rules; for example, requiring that a telephone number be all digits.

SHORT_STRING fields are appropriate for fields that will contain a single selection from a list of choices (choicelist). The choicelist can either be hard coded into the schema (as a constant list) or maintained through the client interface (as a dynamic list). The latter method has the advantage of not requiring you to check out the schema, check in the schema, and upgrade the user database when the choices change. You can

enforce the policy that only the listed choices are valid by selecting the **Limit to list** check box on the choicelist property sheet. This prohibits anyone from entering an invalid entry.

If you want to provide a set of choices, but let someone select more than one of them, you can either provide different fields (and corresponding check box controls), or you can use a MULTILINE_STRING field and a list box control.

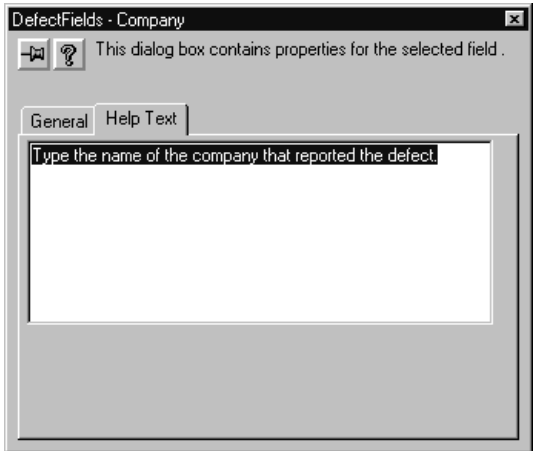
Adding Help Text to a Field

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 will be able to view the help text by right-clicking the field on the record form and selecting **Help** from the shortcut menu.

To add Help text to a field:

1 In the Field Properties dialog box, select the **Help Text** tab.

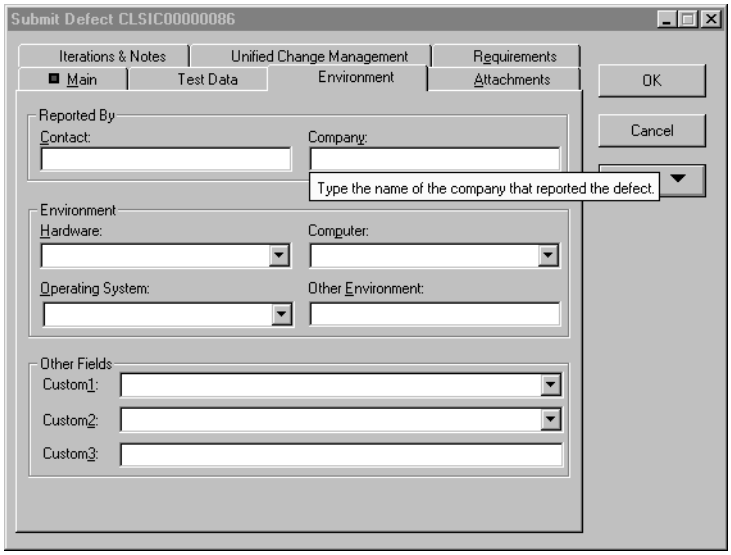
2 Enter the Help text. The text limit is 254 characters, which is about two-thirds of the space available on the Help Text tab.



3 To make the Help text available to ClearQuest client users, check in the schema and upgrade the user database.

To view the Help text in ClearQuest client:

In the ClearQuest client, right-click a **field** and click **Help** from the shortcut menu.



Note: To see the Help text in ClearQuest Web, pause the cursor over a field name and click when you see the question mark.

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 in 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.

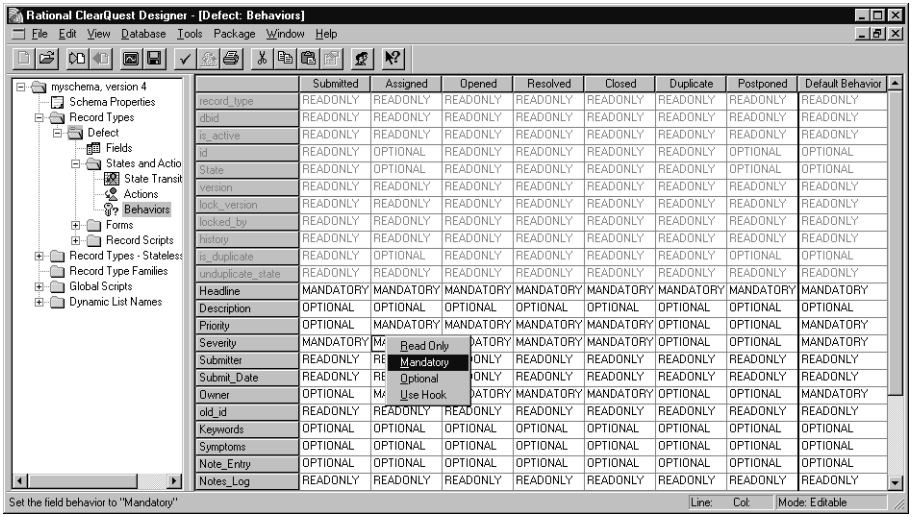
ClearQuest supports the following field behaviors:

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 of ClearQuest, expand the **Record Types** folder and the record type.

2 Expand **States and Actions** and double-click **Behaviors**.



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

Defining Default Field Behavior

The default behavior of a field applies to the field in every state where you have not explicitly set another behavior. Also, the default behavior is automatically applied when you add a new state to the record type.

To define the default behavior for a field, in the Behaviors grid, click the Default Behavior column of the desired field (the last column on the right) and select a behavior to use as the default.

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. However, 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 of ClearQuest Designer, expand the **Record Types** folder, 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 click **Field Properties** from the shortcut menu. You can also select the field and click **Edit > Field Properties**.

Changing the Name of a Field

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

To change the name of a field:

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

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

To delete a field:

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 Field* on page 108.

Using Fields to Link Records

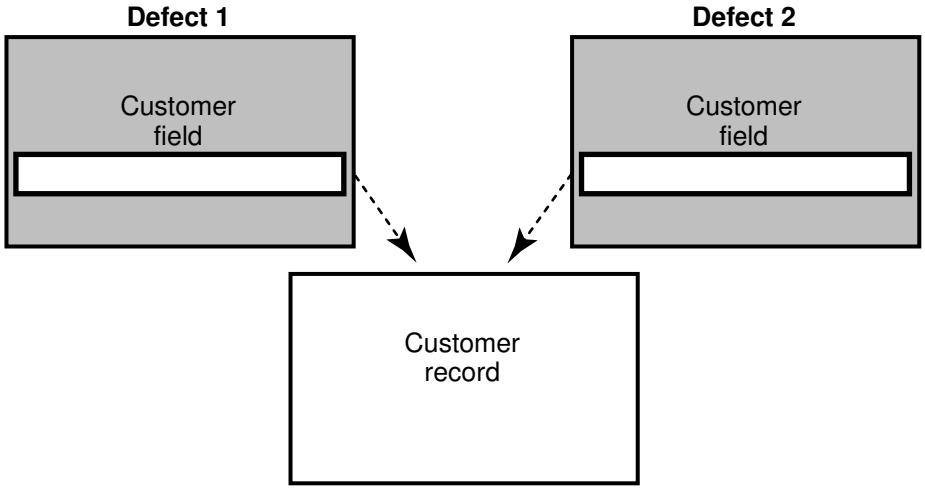
You can use fields to link records of the same type or records of different types. You can link records to:

- Share common data.
- Create a parent-child hierarchy.
- Link records to share common data.

You can use REFERENCE or REFERENCE_LIST type fields to link records in order to share common data. To link two records, use a REFERENCE field. To link multiple records, use a REFERENCE_LIST field.

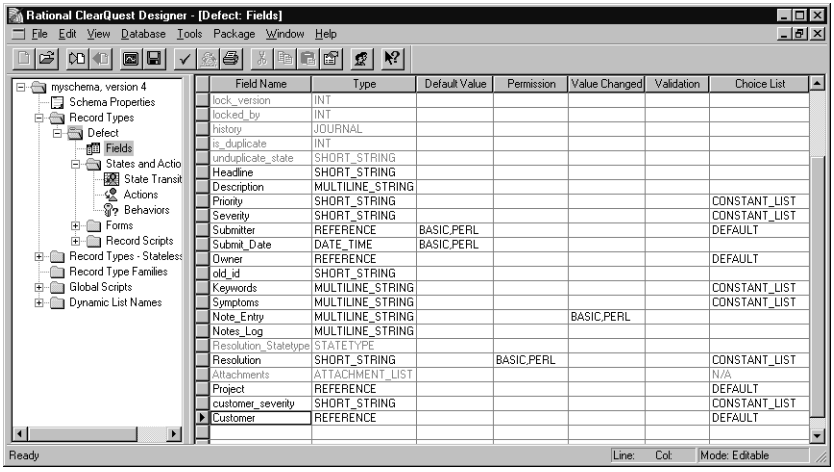
For example, you might have the same customer data that must be entered for multiple records (Figure 3).

Figure 3 Sharing Common Data with REFERENCE_LIST Fields

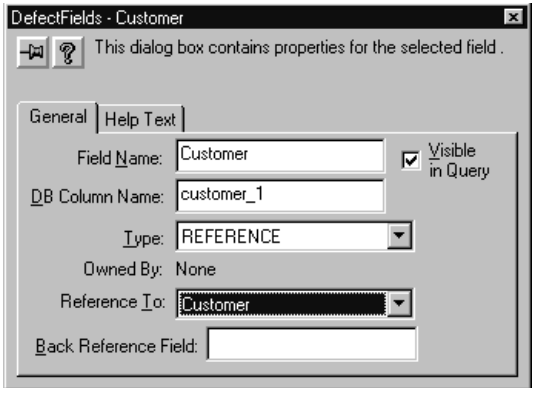


To link records to share common data:

1 Display the Fields grid for the record type and create a new field. Select **REFERENCE** or **REFERENCE_LIST** as its type.



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



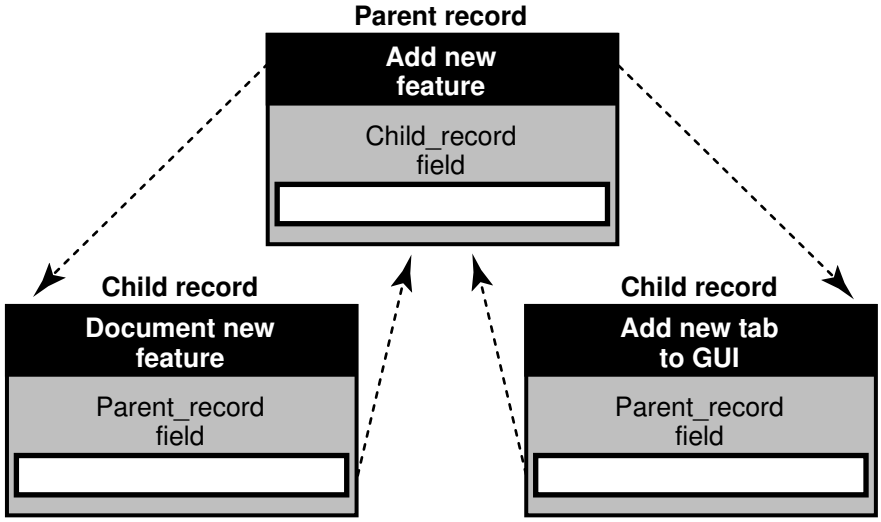
3 In the Field Properties dialog box, select the record type to refer to from the **Reference To** list.

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. For more information, see *Adding Controls to a Form* on page 148.

Linking Records to Create a Parent/Child Hierarchy

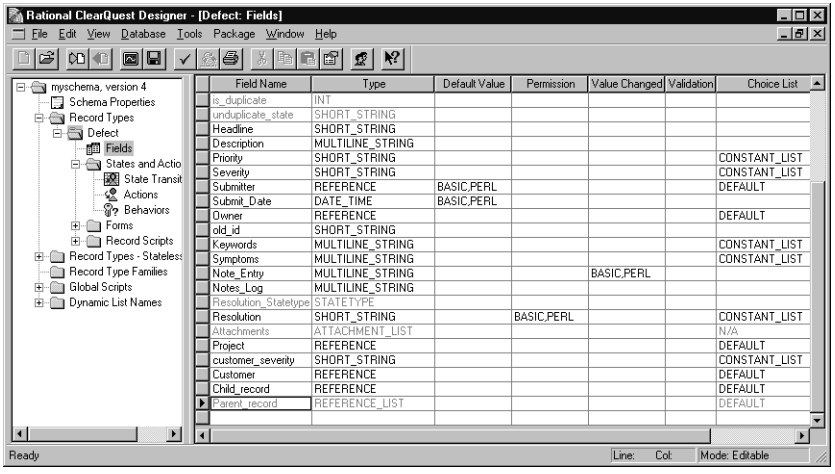
You can use REFERENCE or REFERENCE_LIST type fields to link records of the same type to create a parent/child hierarchy. For example, you can relate a parent record that requests the addition of a new feature to one or more child records that describe related tasks, such as documenting the new feature and adding a new tab to the interface (Figure 4).

Figure 4 Example of Parent/Child Hierarchy

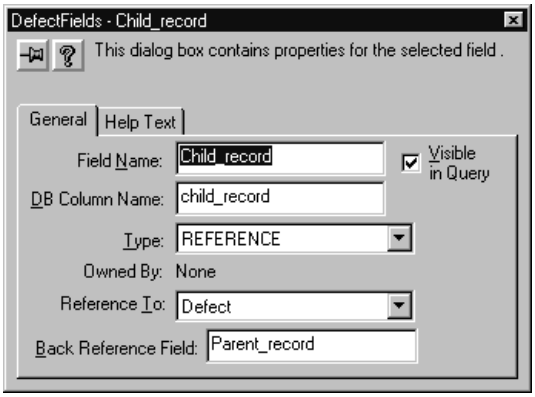


To link records to create a parent/child hierarchy:

1 Add a **REFERENCE_LIST** or **REFERENCE** type field to the parent record.



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



3 Enter a name for a **Back Reference** field. A Back Reference field is a read-only field that makes traversing parent-child relationships easier by allowing you to view the link from the perspective of the child record. This field is automatically added to the field list of the child record.

4 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 148.

For more information, see *Customizing Actions by Adding Hooks* on page 123.

Note: Reference_Lists can have a performance impact, so they should not be overused. Each time a form is displayed, or its contents are refreshed from the database (such as when selecting another row in the results grid of a query), the entire contents of that record is requested from the database. Then for each reference_list field, a subsequent query is performed to retrieve all the fields of the referenced records that are displayed on the form. If form controls display nested attributes several levels deep, this process can repeat itself until all the display data is retrieved.

Customizing Fields by Adding Hooks

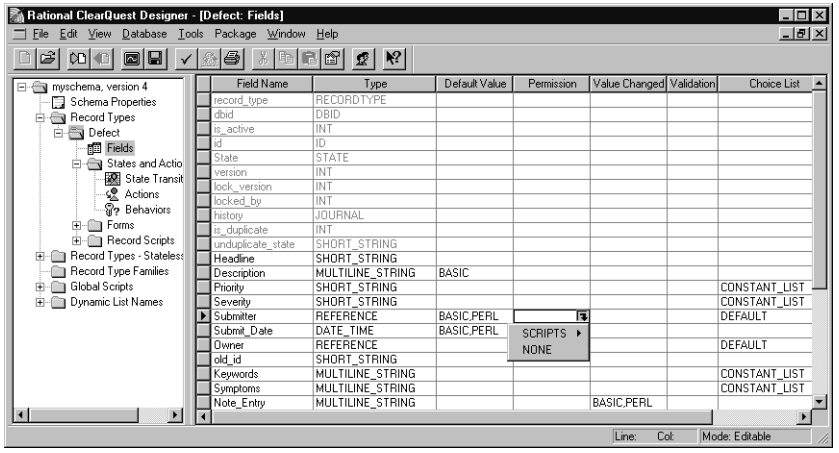
Hooks allow you to customize how fields work. For example, you can customize the schema so that field default values are assigned whenever someone submits a new record. See *Validating User Input in a Field* on page 203

ClearQuest provides several field hooks:

- Default Value
- Permission
- Value Changed
- Validation

- 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, click **Hooks > Compile** to check the syntax of your code.

For more information, see *Working with Field Hooks* on page 194.

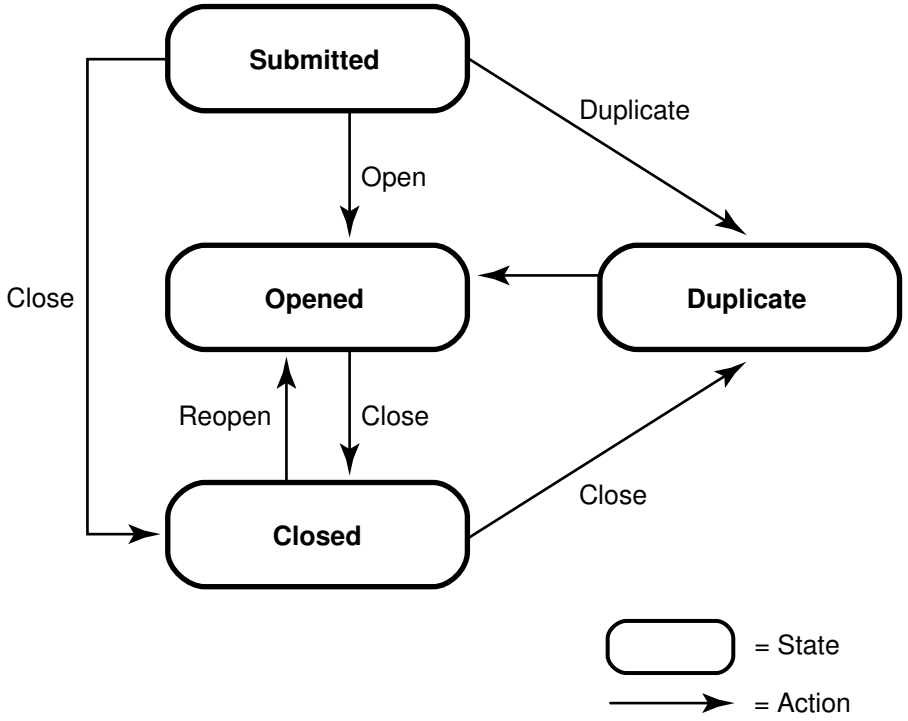
Overview of State Models

In ClearQuest, a state model is a representation of the process model for one type of change request. A state represents the status of an individual change request, for example Submitted, Assigned, Opened, and Closed. An action is an activity by a ClearQuest user that moves a change request from one state to another. Typical actions include Submit, Assign, Modify, and Close.

The movement of a change request from one state to another is called a state transition. A state transition consists of a source state (the current state), a destination state, and the action that moves the record from the source state to the destination state. Typical actions include Submit, Assign, Modify, and Close.

The state model shows all valid states and state transitions that apply for that type of change request. For example, the state model that follows 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 (Figure 5).

Figure 5 State Model for the Enhancement Record Type



The best way to begin designing a state model is by listing and describing all states that you think are valid for a specific record type (that is, a specific type of change request. For example, the following table describes the states for the EnhancementRequest record type.

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

ClearQuest gives you two methods to modify state models and actions, the State Transition Matrix and the Actions grid.

The State Transition Matrix represents the state model as a grid, similar to a spreadsheet, with the states listed on both the side axes and the top axis, and the actions that allow transitions between states shown in the cells.

Customizing State Models with the State Transition Matrix

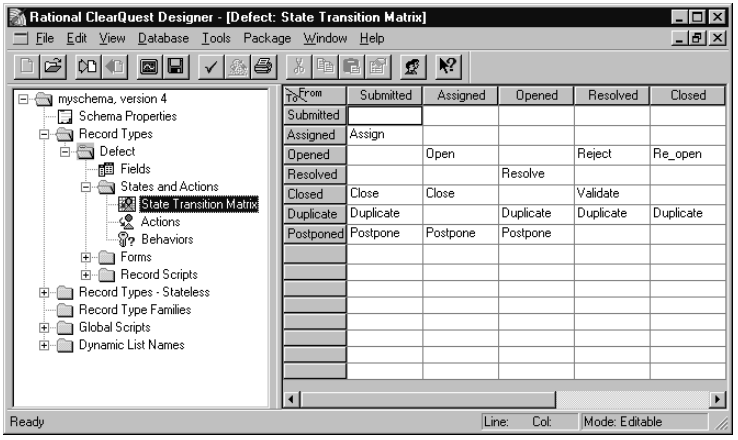
Each state-based record type has a State Transition Matrix that lists all states that are valid for that record type and the actions that move the record from one state to another.

You can use the State Transition Matrix to create, modify, and delete states.

Displaying a State Transition Matrix

To display the State Transition Matrix for the Defect record type, do the following:

1 In the Workspace of ClearQuest Designer, expand the **Record Types** folder, expand the **Defect** folder, expand the **States and Actions** folder, and double-click **State Transition Matrix**.



2 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 transition between those two states is not allowed.

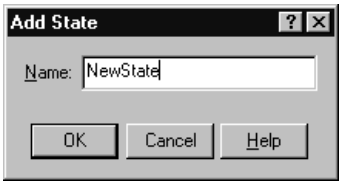
Note: You can have only one action that makes the transition between a given source state and a given destination state.

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 of ClearQuest Designer, expand the **Record Types** folder, select the record type, expand the **States and Actions** folder, and double-click **State Transition Matrix.**

2 Click **Edit > Add State**.



3 In the Add State dialog box, enter 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.

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 122.

After you add a state to your schema, you must connect it to at least one other state with an action. 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 116.

Mapping State Types

Some schema packages add hooks (scripts) to schema that run when a change request enters a specific state. The UnifiedChangeManagement (UCM) package and the Resolution package are examples of this type of package.

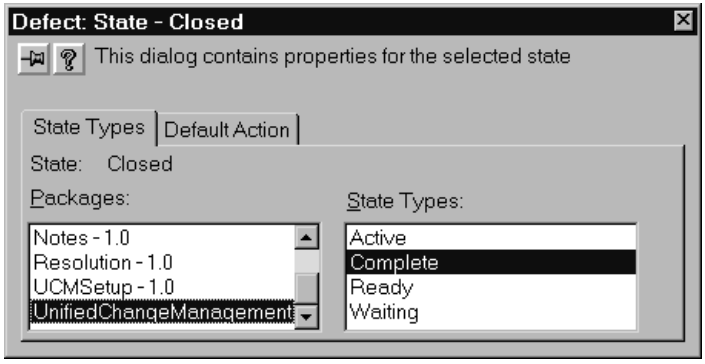
To ensure that the hook runs at the right time, you need to map each state of that record type to a state type of the package. Each state type of the package must have at least one state from the record type mapped to it (which ensures the hooks in the package will have at least one state in which they will run). You can map more than one state to a state type of the package.

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. In ClearQuest Designer, display the **State Transition Matrix** for the record type.
2. Right-click a **state** and select **Properties** to display the **Properties** dialog box for the state.
3. Click on the **State Types** tab, select a package that requires state type mapping, then select a state type.
4. Close the dialog box to save the state mapping.
5. Repeat this process for each new state in the record type.

For example, to map the Closed state in an existing schema to the Complete state select the UnifiedChangeManagement package. Then select the Complete state type to map it to the Closed state.



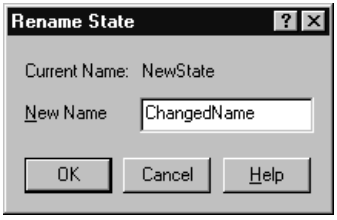
Note: If you are using the UCM schema or package, you must also assign default actions for your states; see *Setting the Default Actions for UCM* on page 308.

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 of ClearQuest Designer, expand the **Record Types** folder, select the record type you want, expand the **States and Actions** folder and double-click **State Transition Matrix**.
2. In the State Transition Matrix, click the state that you want to rename and click **Edit > Rename State**. Or right-click and select **Rename State** from the shortcut menu.



3 In the Rename State dialog box, enter 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 of ClearQuest Designer, expand the Record Types folder, select the record type you want, expand the States and Actions folder, then double-click State Transition Matrix.

2 In the State Transition Matrix, click the state that you want to delete and click **Edit > Delete State**. Or right-click and select **Delete State** from the shortcut menu.

Working with Actions and Action Types

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.

ClearQuest client users select actions by clicking **Actions** button on a form or by selecting the **Actions** menu on the tool bar. In both cases a list of valid actions displays, with the default action highlighted in bold text. You define the default action using the **Default Action** tab on the State Properties dialog box. See *Using Default Actions* on page 124. You can also call a default action from a hook.

An action can:

- Create a new record (change request) 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 123.
- Delete a record from the database.

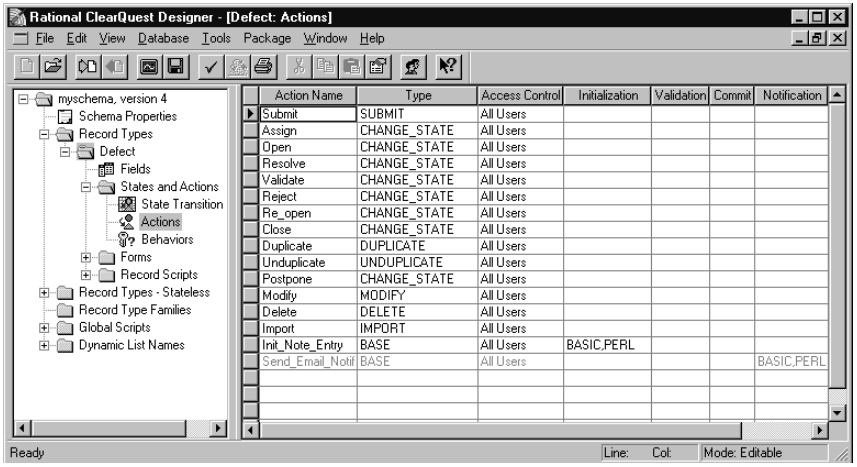
You can customize which users have access to specific actions, as well as when actions can be performed.

ClearQuest includes a number of predefined action types for commonly used actions, for example the Submit action and the Change State action. See *Supported Action Types* on page 120.

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 of ClearQuest Designer, expand the **Record Types** folder or the **Record Types - Stateless** folder, then expand the desired record type.
2. For state-based record types, expand **States and Actions**. Double-click **Actions**.



Supported Action Types

ClearQuest supports the following types of actions:

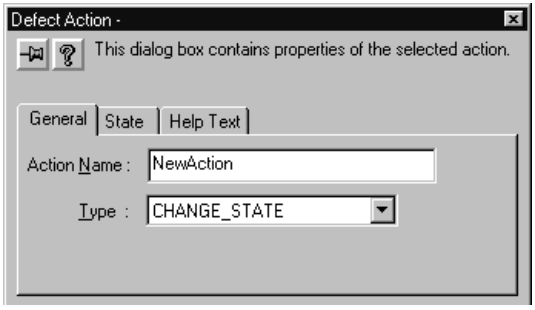
Action type	Description/Comments
Base	A Base action is a secondary action that runs as a side-effect of every other action. Base actions let you write an action hook only once, but use it with multiple actions. Each time an action fires, the Base action checks to see if the hook criteria is met; if it is, the base action completes its process. 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. Base actions do not appear in the list of actions in the ClearQuest client.
Change_state	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. Change_state actions appear in the list of actions in the ClearQuest client only if the current record is one of the source states.
Delete	Delete lets users delete a record from the database. Delete actions appear in the list of actions in the ClearQuest client.

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

Adding a New Action

To add a new action:

1. In ClearQuest Designer, display the Actions grid.
2. Click **Edit > Add Action**.
3. In the **General** tab of the Actions Properties dialog box, enter an **Action Name**.



4 Select an action **Type**. 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 states and a destination state for that action:

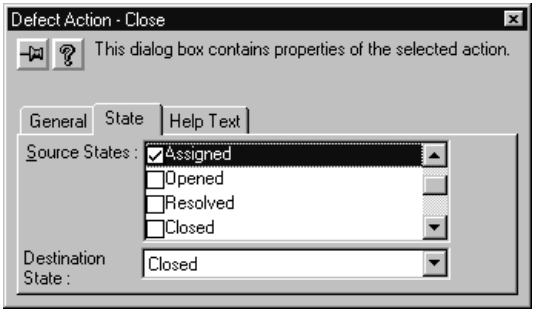
1 In the Workspace of ClearQuest Designer, open the Actions grid.

Expand the **Record Types** folder, select the record type you want, expand the **States and Actions** folder, 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 *Displaying a State Transition Matrix* on page 115.

Modifying Actions

To modify an action:

1 In the Workspace of ClearQuest Designer, expand the **Record Types** folder or the **Record Types - Stateless** folder.

2 Expand the record type you want.

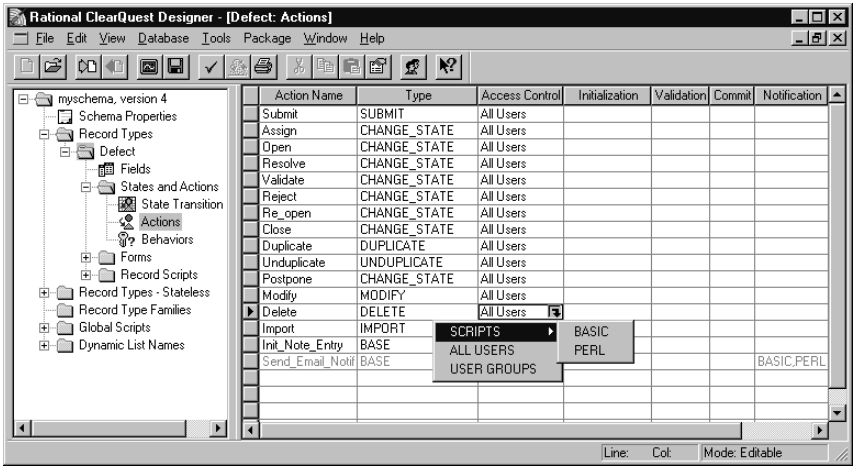
3 For state-based record types, expand the **States and Actions** folder. Double-click **Actions**.

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, click **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 10, *Using Hooks to Customize Your Workflow*. To learn how to create an access control action hook, see *Action Access Control Hook Example* on page 331. Also see *Selecting a Scripting Language* on page 85.

Using Default Actions

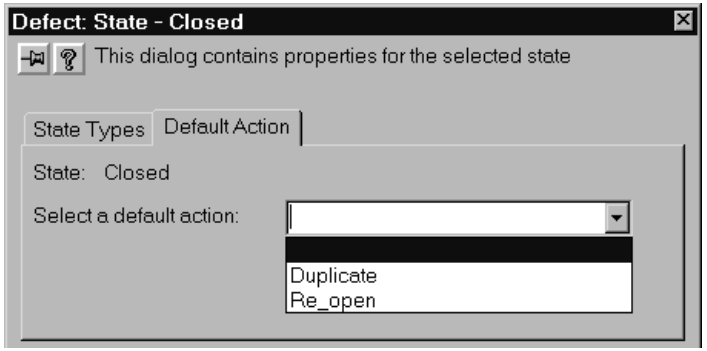
You can define default actions for states. A default action for a state appears in bold in the ClearQuest client Actions menu. Default actions:

- 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 308.
- Can be called from hook code.

Note: Before you can define the default action for a state, you must first create a state transition.

To define the default action for a state:

1. In the Workspace of ClearQuest Designer, 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 In the Workspace of ClearQuest Designer, open the Actions grid.

Expand the **Record Types** folder, select the record type you want, select the **States and Actions** folder, then double-click **Actions**.

Or, expand the **Record Types - Stateless** folder, select the record type you want and double-click **Actions**.

2 Select the row of the action you want to delete.

3 Click **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.

Considerations in Designing State Models

State models must be designed and managed with care.

Black Holes

You must make sure there are no *black holes* in the model where a change request can land and be overlooked. For example, if you had a Postponed state, and no one was ever assigned to process records in this state, you would start accumulating change requests that were never noticed or fixed.

Sometimes the causes of these black holes can be subtle. You might have three engineers assigned to handle change requests in an Open state where the value in the component field is red, green, and blue, respectively. However, if someone submits a change request with yellow in the component field, or leaves the component field blank, you might have change requests sitting unresolved.

How Many States?

There are trade-offs to consider between having a small number of states with more development activities in each one, versus a larger number of states with less development activity in each. For example, if some kind of verification activity occurs

at several points in the lifecycle of a change request, you might want to have a separate *verification* state instead of including verification activities as part of each of the other states. This would make it easier to ensure that verification was handled properly. On the other hand, creating too many states makes the model unwieldy and harder to maintain.

Duplicates

The Duplicate action can be very useful. It provides a means to identify one of a set of duplicate records in the database as the *active* record, and to mark all the others as duplicates of it, which protects them against further modification. This ensures that all work on this common issue is coordinated through one record in the database. This functionality uses built-in fields, and can be added to a schema by adding the Duplicate Dependent and Duplicate Base controls onto a form, and creating Duplicate and Unduplicate actions. The other notable feature of this is that Unduplicate returns the record to the state that it was at when it was designated as a duplicate.

Predefined Schemas

ClearQuest includes a variety of predefined schemas that illustrate certain features or configurations. You can use these as a starting point for developing a custom schema, but you should be careful to compare the features of the predefined schema with your requirements.

Some key points to consider about these predefined schemas:

- The Blank schema should be used as a basis when you want to design your own new schema from the start.
- The Common schema defines a single Defect record type with many features you may (or may not) need.
- The Defect Tracking schema demonstrates the record cross-referencing capabilities within the database and the navigational features of the ClearQuest clients. It is based on the Common schema and adds a Customer stateless record type with a Reference_List field to it from the Defect record type.
- The Analyst, Development, and Test predefined Studio schemas are based on the Common schema and facilitate integration with the corresponding Rational Suites products.
- The UnifiedChangeManagement schema is also based on the Common schema extended for the UCM ClearCase integration.
- The EnterpriseStudio schema is based on the Common schema, and facilitate integration with all the Rational Suites integrations as well as UCM.

Parallel Development

Designing schema to support parallel development of multiple products (or product variants) that share common artifacts can be challenging.

In these situations questions arise like: “If a defect is reported that is traced to shared artifacts requiring fixes to multiple products, how should this information be captured and handled by the system?” and “How should the system track the current state of the defect when multiple builds (on potentially different schedules) are required?”

Organizations using a single record to track these different efforts have difficulty dealing with these issues.

One approach to this problem is to submit multiple records for each product affected. This allows the status of each activity to be tracked independently. If you use the Save Default Values mechanism on the first record entered, and Load on subsequent records you can avoid users duplicating the data entry on each copy. This capability is not available on the Web client, however.

The downside to this approach is that each record is isolated from the others, and wasted effort can result from not knowing that work is being done on the other related issues.

A preferred approach is to use a hierarchical structure, where the parent characterizes the issue, and the children are used to track the common issue for each product, variant, or version. The parent record can be a different type than the children, or it can be the same.

In some cases, schemas have used the same record type so that all the information can be contained in the child (which reduces navigating between parent and child). Other cases have used an almost trivial child record type to implement a kind of *reminder* to address the same issue for the other affected products, variants, or versions, and to track their status.

Applying Packages

6

Rational ClearQuest packages provide an efficient method to add new functions to a schema, and to integrate schemas with other software packages.

The topics covered include:

- *Overview of Packages and Integrations* on page 129.
- *Cautions About Applying Packages* on page 131.
- *Packages Available* on page 132.
- *Examples: The Notes, Attachments, and E-Mail Packages* on page 133.
- *Checking Which Packages Have Been Applied to a Schema* on page 134.
- *How to Apply a Package* on page 135.
- *How to Upgrade Packages* on page 140.

Overview of Packages and Integrations

A Rational ClearQuest *package* is a set of schema components that can be added to an existing schema to provide a complex feature or function. The new feature or function then becomes a permanent part of the schema.

A package can add a function that has been found useful by many ClearQuest customers, such as the ability to append a list of notes to a defect record, or to send an automatic e-mail notification when a specific event occurs.

A package can also allow ClearQuest to exchange data with other Rational software products and with some third-party software packages such as Microsoft Visual Source Save (VSS) and Microsoft Project. These packages are called *integration packages* or *integrations*.

Schema components that can be added through a package include:

- Record types
- Form features, including tabs and new forms
- Field definitions and behaviors
- Hook code and global scripts
- Dynamic list names
- Schema validation tests
- Queries, charts, and reports

You can apply more than one package to the same schema.

Usually you can apply a package automatically in a single operation. In other cases you may be permitted to customize or configure a package as it is being installed, for example, changing the names of fields. In some cases you may be required to configure the package; for example, there are some integration packages where you are required to map the state types in the existing schema to the state types included in the package. These customizations are discussed further in *ClearQuest Schemas and Packages* on page 277 and in the *Upgrade Guide* for Rational Suite.

To install a package for the first time, use the Package Wizard included with ClearQuest Designer to help you select and install the new package. This is discussed in *How to Apply a Package* on page 135. You can also apply the CharacterSetValidation package with a script from the command line. For more information, see *About the CharacterSetValidation Package* on page 38.

As with other changes to a schema, functions in a package are available to users only after the package has been applied to the schema *and* the user databases have been upgraded to use the new version of the schema.

You do not need to worry about developing or maintaining installed packages, because Rational provides new versions and supports change management of existing packages. When you upgrade to a new version of ClearQuest, use the Upgrade Installed Packages feature to automatically upgrade all packages that have been installed for a given schema. This is discussed in the *How to Upgrade Packages* on page 140.

Note: You can only use the Package Wizard and the Upgrade Installed Packages feature with packages provided by Rational.

Packages and Predefined Schemas

Predefined schemas are schemas created from the Common schema and a set of packages.

For example, the Defect Tracking schema is derived from the Common schema after the following packages have been applied:

- Notes
- Resolution
- Attachments
- History
- E-Mail
- Customer

Cautions About Applying Packages

After you add a package to a schema, you cannot remove the package from that version of the schema. Therefore, before applying a package you should carefully review the contents to make sure it meets your needs.

Also, when you modify a schema by applying a package, do not upgrade any production user databases to the modified schema until you have used a test database to make sure that schema behaves exactly as you intend.

If you find that a modified schema does not behave as you intend, you can delete the schema version to which the package has been applied, *but only if you have not yet upgraded a production user database to this version*.

Also some components of a package either *cannot* be modified after the package has been applied, or *should not* be modified except with great care under exceptional circumstances. This is because either ClearQuest or an integration with ClearQuest may *expect* to see a certain configuration or a specific name for a component. For example, several integration packages will fail if a field called *headline* has been renamed, or if an action called *modify* has been renamed or removed.

To avoid this type of problem, some components that are installed by packages are read-only and cannot be modified either by changing their name or extending their capabilities. In other cases, you will not be prevented from making modifications, but you should avoid doing so unless absolutely necessary. For more information on dependencies of this type, see *ClearQuest Packages* on page 279.

Packages Available

An overview of the packages currently available in ClearQuest is included in *ClearQuest Schemas and Packages* on page 277. This appendix lists for each package its name, description, outline of features added or modified, and list of fields modified.

The packages currently available in ClearQuest, grouped by category, are outlined in Table 14.

Table 14 Overview of ClearQuest Packages

Package type	Package name
General	▪ Attachments ▪ E-Mail ▪ History ▪ Notes ▪ Resolution
Defect Tracking	▪ Project ▪ Customer
Data/Code Page Validation	CharacterSetValidation
UCM integration	▪ AMStatesTypes ▪ BaseCMActivity ▪ UCMPolicyScripts ▪ UnifiedChangeManagement
Rational Suites Integration	▪ Repository (Rational Repository) ▪ PQC (Purify, Visual Quantify, and Pure-Coverage in Rational Development Studio and TestStudio) ▪ TeamTest ▪ RequisitePro ▪ RequisiteProSupplement ▪ EnhancementRequest ▪ ContentStudio
Configuration Management (CM) Integration	▪ ClearCase ▪ VisualSourceSafe

Table 14 Overview of ClearQuest Packages

Package type	Package name
Rational ClearCase Project Tracker	▪ AMBaseActivity ▪ AMWorkActivitySchedule

The Repository package supports the integrated functionality of Rational Repository in Rational Suites. The PQC package supports the integrated functionality of Purify, Visual Quantify, and PureCoverage in Rational Development Studio and TestStudio. The TeamTest package supports the integrated functionality of Rational TeamTest. The RequisitePro and RequisiteProSupplement packages support the integrated functionality of Rational RequisitePro in Rational Suite products. The Enhancement Request package introduces a new state-based record type in addition to the previous defect record type to provide a broader range of applications for the defect and change tracking process. The ContentStudio package provides support for integration with Rational ContentStudio.

Examples: The Notes, Attachments, and E-Mail Packages

Three of the most commonly used packages are: Notes, Attachments, and E-Mail.

The *Notes* package provides a mechanism to log notes for a record type. The notes log is typically used to capture ongoing comments about a change request by ClearQuest users. These comments might cover elaborations or clarifications about the request, suggested solutions, comments about the priority or severity of the request, and other useful observations.

When you apply the Notes package to a schema, you select one or more record types (for example, *Defects* and *EnhancementRequests*). The Notes package will add a Notes tab with two controls for each selected record type. One control is for entering a new note, and one control is for viewing the full set of notes captured in the log. If a ClearQuest user clicks on the latter control, ClearQuest will display the existing notes separated by a note header that includes the state, user name, date, and time of the note.

The *Attachments* package lets users add and remove file attachments related to a record. When you apply the Attachments package to a schema, you select one or more record types, and the package will add an **Attachments** tab to each selected record type.

The *E-Mail* package provides a mechanism for automatically creating e-mail messages that notify team members when a specific action takes place. The package adds a stateless EmailRules record type that, when instantiated in the client, lets the user

define the criteria upon which an e-mail notification will be generated, who it will be sent to, and what it will contain. By creating multiple EmailRules records, you can define different conditions under which e-mail notification will be automatically generated. The e-mail notifications can be sent to different individuals and user groups.

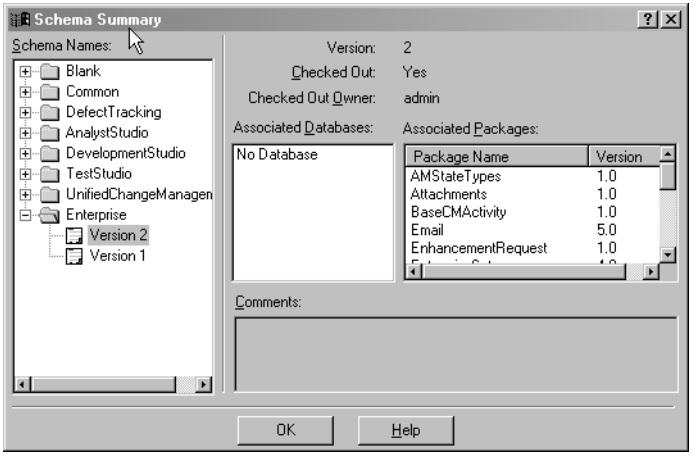
These examples also illustrate why you may want to exercise judgment when applying and configuring packages. For example, when the Attachments package is deployed with the ClearQuest for Windows Web client, uploading and downloading large attachments can create a bottleneck that slows response times for other ClearQuest Web users. It is important to understand that EmailRules are evaluated every time a record is written to the database, so too many of them can impact response time when the user clicks on the **OK** or **Apply** button.

You may be able to create custom hooks that reduce these problems, for example, by restricting the maximum size of attachments or by developing a custom e-mail notification hook. For more information about such solutions, refer to the Rational Developer Network on the Rational web site.

Checking Which Packages Have Been Applied to a Schema

Before applying new packages or package versions, you can check which package versions have already been applied to an existing schema.

1 In ClearQuest Designer, click **View > Schema Summary**. ClearQuest displays the Schema Summary dialog box.



2 Select a schema (for example, DefectTracking) and expand the folder to view the available versions. The Schema Summary dialog box displays a pane labeled Associated Packages that lists the packages and associated versions that have been applied to that schema.

How to Apply a Package

The process of applying a package to a schema can be as simple as selecting a package and clicking **Finish**, or can require as many as four steps:

- Selecting the schema.
- Selecting the package to be applied.
- Selecting the record types in the schema that will be modified by the package.
- Mapping the existing state types in the schema to the states in the package.

The last two steps can be performed either when you apply the package using the Package Wizard, or later from the ClearQuest Designer main window.

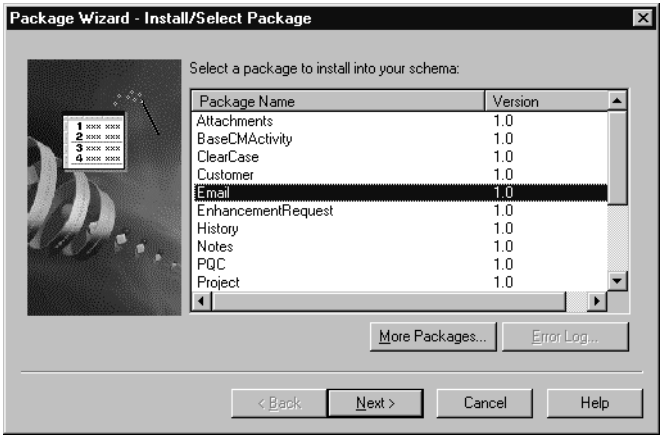
The order of the steps varies depending on whether you first start the Package Wizard and then select the schema to be modified, or whether you first check out a schema and then start the Package Wizard.

The procedure that follows describes the order to use if you start the Package Wizard first. As noted, depending on the schema and the package selected you may not need to perform all of the steps.

Note: You cannot also apply the CharacterSetValidation package with a script from the command line. For more information, see *About the CharacterSetValidation Package* on page 38.

To apply a package by starting with the Package Wizard, do the following in ClearQuest Designer:

1 Start the Package Wizard by clicking **Package > Package Wizard.** ClearQuest Designer displays the latest package versions.



2 If the package version you are looking for is not listed, click **More Packages**. If you select a version from this list and click **OK**, the package will display back in the original **Install/Select** dialog box.

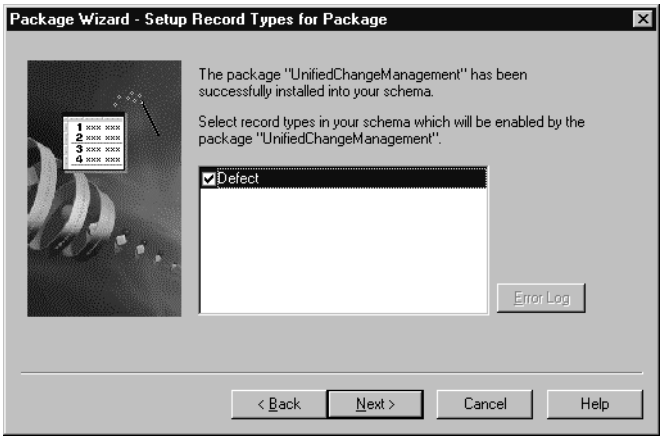
3 Select the package you want to apply from the Install/Select dialog box and click **OK**.

4 In the second page of the wizard, select the schema you want to modify with the package.

ClearQuest verifies that the package you selected can be applied to the schema. If the package cannot be applied to the schema, an error message appears. For example, if the package modifies a particular field, and that field does not exist in the schema, ClearQuest will not allow any of the components of the package to be applied to that schema.

ClearQuest then determines whether the package you have selected can be applied to the schema without affecting any of the schema components. If so, you will be able to click **Finish** to complete the process. If not, you will need to click **Next** and go on to the next step.

5 If ClearQuest determines that you need to modify record types in your schema to complete the package implementation, the Package Wizard Setup Record Types for Package dialog box appears.

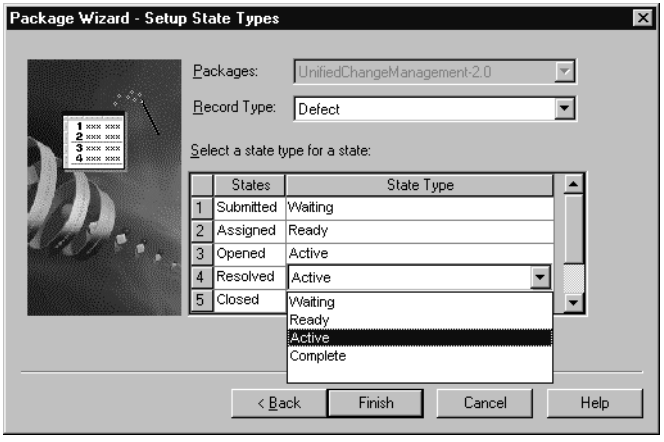


This dialog box lists record types in your schema that the package could *enable* or modify, for example by adding fields, actions or a script. If the package does not include customizations for enabling existing record types, the list is empty.

Check the record types that you want the package to enable. Click **Finish** or **Next**.

Note: You have the option of skipping this step now by clicking on **Finish** or **Next** without checking any of the boxes. You can enable the record types in the Package Wizard later by clicking **Package > Setup Record Types for Packages**. For more information, see *Enabling Record Types* on page 138.

6 When you add a package that uses state types, the Package Wizard Setup State Types dialog box appears.



In this page of the wizard, you must assign state types to the states for each record type that you choose to enable.

For each state in the package, click the **State Type** field and select the appropriate state type from the existing schema that should be mapped to this state. 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. When you are done, click **Finish**.

Note: You have the option of skipping this step now by clicking **Finish.** You can set the state types later by clicking **Package > Setup State Types**. For more information, see *Mapping State Types* on page 139.

If you want to apply a package by first checking out a schema (instead of first starting the Package Wizard), follow the process already described, except start by clicking **File > Open Schema** and selecting the schema and version to which you want to apply the package. Perform Step 1 through Step 3 for starting the Package Wizard and selecting the package you want to apply. Perform Step 5 and Step 6 for enabling record types and assigning state types to states (if necessary). You then validate and check in the modified schema.

Again, you have the option of skipping Step 5 and Step 6 (enabling record types and mapping state types) and performing them later from the ClearQuest Designer.

Enabling Record Types

When a package is set up to enable record types that already exist in your schema, you can decide which of these existing record types will be modified and which will be left unmodified. For example, a package might be set up to add new fields to existing record types, and you might decide not to add these new fields in some of the record types. Other types of modifications that you may or may not want applied include adding actions and scripts.

For descriptions of packages, including the record types they modify, see *ClearQuest Packages* on page 279.

To enable record types in ClearQuest Designer:

1. Click **Packages > Setup Record Types for Packages**. The Package Wizard Setup Record Types dialog box appears.
2. Check the record types you want to be enabled by the package and click **Finish**.

Mapping State Types

Some packages, such as the UnifiedChangeManagement package and the Resolution package, include a state model with state types. These packages include hooks that are triggered when a change request enters one of these states.

A problem would arise if you applied one of these packages to an existing schema, and the state model of the existing schema did not include a way of getting to a state that came with the package. For example, if the package included functionality that was triggered only when a change request enters the *Ready* state, but the existing schema did not include a *Ready* state, then the new functionality would never be triggered. Users would never benefit from this new functionality.

To prevent this situation, ClearQuest asks you to map the states in your schema to the state types in the package being applied (if the package includes any state types).

Every state type in the package must have at least one state from the existing schema mapped to it. This ensures that the new schema created by the application of the package will include a way of getting to every state type in the package.

For example, if there is functionality in the package that is triggered when a change request enters the *Ready* state, and you have mapped the *Assigned* state in the existing schema to the *Ready* state type in the package, then you know that the new functionality will be triggered when a change request enters the *Assigned* state.

To map state types using ClearQuest Designer:

1. When a package that uses state types is applied to a schema, click **Package > Setting State Types**. The Package Wizard Setup State Types dialog box appears. Select a record type.
2. For each state in the package, click the **State Type** field and select the appropriate state type from the existing schema that should be mapped to this state.
3. If more than one record type appears in the Record Type field, select each additional record type and then map the state types in the existing schema to the states in the package associated with that record type.
4. Click **Finish**.
5. When you have completed the last step of the Package Wizard, click **File > Validate** and ClearQuest will validate the modified schema.

Creating Default Actions

You can define default actions for states. Setting a default action ensures that there is always a way to move from a source state to a target state. Default actions are also useful for guiding users through your state model. A default action for a state appears in bold in the ClearQuest client Actions menu.

Certain schemas and packages, such as the UCM schema and package, require default actions. If you use the UCM schema or package, the default actions of your states must provide a valid path through the state type model.

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 122.

To assign a default action for a state:

1. In ClearQuest Designer, expand **Record Types**, then expand the record type you have enabled with the package.
2. Double-click the **State Transition Matrix** for that record type.
3. In the first column of the State Transition Matrix, right-click a state and select **Properties** to open the Properties dialog box for that state.
4. 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.

How to Upgrade Packages

Rational periodically provides new versions of packages. These new versions contain new functionality that has been requested by ClearQuest users.

If you have already applied a package to a schema, then you can take advantage of this new functionality by upgrading the schema to the new version of the package.

To upgrade a schema to a new version of a package, do the following:

1. In ClearQuest Designer, click **Package > Upgrade Installed Packages**. ClearQuest Designer displays the Upgrade Installed Packages Select a Schema to Upgrade dialog box.
2. Select the schema you want to upgrade.

   ClearQuest checks out the schema, analyzes which packages need upgrading, and determines whether there are prerequisites to those packages.
3. On the Status page, note the list of packages that need upgrading.

4 ClearQuest applies packages to the schema in the correct order and displays the status. When it finishes, read the status message to verify that the package upgrade was successful.

5 Validate the changes you made by clicking **File > Test Work**.

6 When you finish testing, check in the schema by selecting **File > Check In**.

7 After the new schema version is checked in, upgrade any user databases that use an earlier version of the schema. You can then repeat this process for the other schema in your schema repository.

You can then repeat this process for the other schema in your schema repository.

After you have completed this process, you can verify that the new package versions have been applied by following the procedure described in *Checking Which Packages Have Been Applied to a Schema* on page 134.

Working with Forms 7

Rational ClearQuest Designer provides you with the graphical capabilities to create and modify record forms.

The topics covered include:

- *Overview of Working with Forms* on page 143.
- *Creating and Modifying Forms* on page 144.
- *Working with Form Controls* on page 146.
- *Working with Tabs on Forms* on page 157.
- *Reusing Record Forms* on page 159.

For detailed descriptions of the properties for each control type, see Appendix D, *Form Controls*.

Overview of Working with Forms

Each record type can have two forms associated with it: a record form and a submit form. You must create a record form for each record type, but the submit form is optional. A submit form allows users to use a different form to submit records of that type. When a user first submits a new record, Rational 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 for both submitting records and for working with them.

You use the Forms window for:

- Creating and Modifying Forms. You can add and delete forms, change the name of a form, and change the fonts used on forms.
- Working with Form Controls. After creating a form, you can add controls for the fields you want to display on the form.
- Working with Tabs on Forms. By default, each record form contains one tab. You can add additional tabs to the form, restrict access to tabs, and change the order in which tabs appear on the form.

For More Information

- You can create one record form and use it in multiple schemas. See *Reusing Record Forms* on page 159.
- If you are creating forms to be displayed on multiple platforms, see *Creating Forms for Multiple Platforms* on page 146.
- You can use the ClearQuest API to submit, view, and edit records programmatically instead of through forms. See the *API Reference* for Rational ClearQuest.

Creating and Modifying Forms

You must create a record form for each record type in your schema. You can also define a submit form for each record type. By default, each record form and submit form has one tab; you can add additional tabs.

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. 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 of ClearQuest Designer, expand **Record Types** or **Record Types - Stateless** and the record type you want.
2. Right-click the **Forms** folder and click **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 uppercase and lowercase letters, numbers, and underscores.
4. Right-click the new form name and click **Record Form** or **Submit Form**.
5. After creating a form, you must add controls for each of the fields you want to display on the form. See *Working with Form Controls* on page 146. You can also add tabs to group controls. See *Working with Tabs on Forms* on page 157.

Note: Font and language differences can cause forms to look different on the various ClearQuest clients for Windows, UNIX, and Web. 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 of ClearQuest Designer, right-click the form name and click **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 *Working with Tabs on Forms* on page 157.

To change the size of a form:

1. In the Workspace of ClearQuest Designer, expand **Record Types** or **Record Types - Stateless** folder until you see the form you want to resize.
2. Double-click the form.
3. Click a corner of the form window and drag it to the shape and size you want.

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 click **Form Properties** from the shortcut menu.
2. In the Form Property Sheet, enter the width and height values, and 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 of ClearQuest Designer, right-click the form and click **Fonts** from the shortcut menu.
2. In the Font dialog box, select the font and font properties you want.
3. Close the form and then reopen 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 of ClearQuest Designer, expand the **Record Types** or **Record Types – Stateless** folder, then expand the desired record type and expand the **Forms** folder.
2. Right-click the form and click **Delete** from the shortcut menu.

Creating Forms for Multiple Platforms

The forms you create in ClearQuest Designer appear differently when viewed by the Windows and ClearQuest Web clients. For example, in Windows you click a tab to display it; 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.

Be sure to test your forms on all the platforms on which they will be used.

For more information, see Chapter 11, *Administering ClearQuest Web*.

Creating and Modifying Forms for 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.

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 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.

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 applied to a record.
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 multicolumn format.

Form control	Description
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.

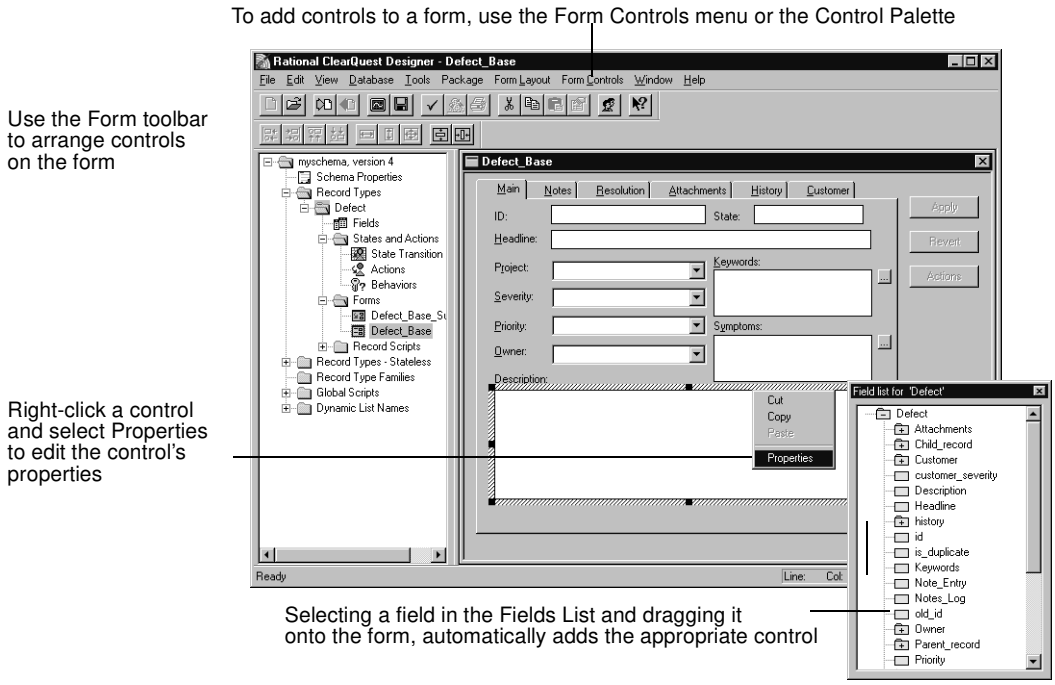
For detailed descriptions of the properties for each control type, see Appendix D, *Form Controls*.

Adding Controls to a Form

Before you can add a field to a form, you must first add the field in the Fields grid. For more information, see *Adding a New Field* on page 101.

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.
- Selecting a field in the Field List and dragging it onto the form, automatically adds the appropriate control to the form.
- 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.



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 In the Workspace of ClearQuest Designer, expand the **Record Types** or **Record Types – Stateless** folder until you see the form you want.

2 Double-click the form.

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

Adding a Control Using the Control Palette

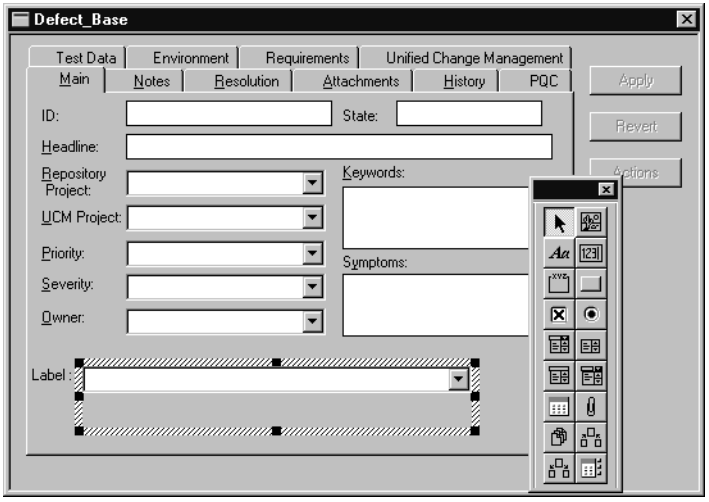
If the Control Palette is not visible when you open the form, click **View > Control Palette**.

To add a control using the Control palette in ClearQuest Designer:

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 size you want.

After 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.



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. For more information, see *Editing Control Properties* on page 153.

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 that field type. 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

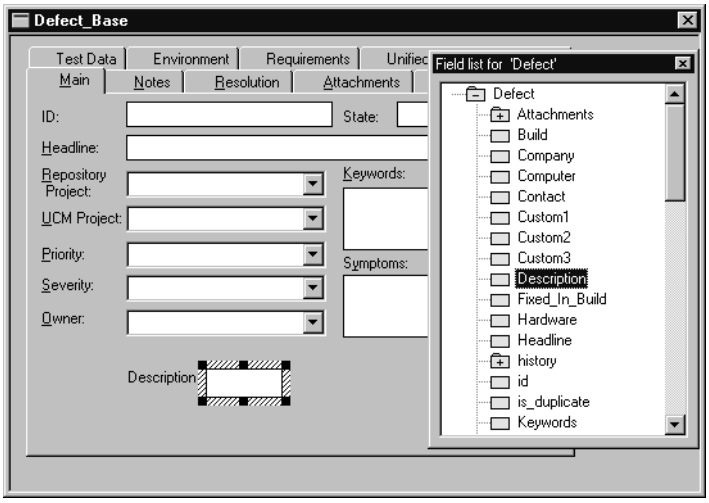
Field type	Default control
Constant List Choice List	Drop-down List Box
Date-Time	Text Box
Integer	Text Box
Multiline String	Text Box
Reference List	Parent/Child
Reference List Constant	List View
Reference Choice List=Default	Drop-down List Box
Short String Constant List	Drop-down List Box
Short String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

To add a control using the Field List:

1 If the Field List is not visible when you open the form, click **View > Field List**.

2 Select and drag a field from the Field List to a tab.

After 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.



3 After adding the field to the form, you can edit the control properties to set the display options of the control. Double-click the control to open its properties sheet. For more information, see *Editing Control Properties* on page 153.

Adding a Control Using the Form Controls Menu

To add a control using the **Form Controls** menu in ClearQuest Designer:

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 size you want.

After 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.

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. For more information, see *Editing Control Properties* on page 153.

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

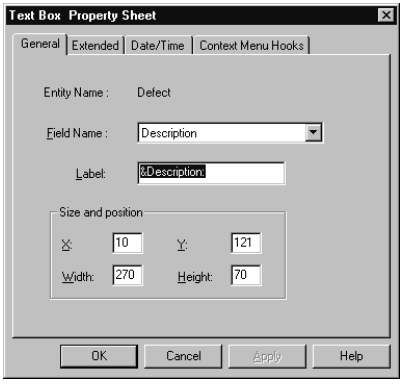
Note: If you use the Control Palette or the **Form Controls** menu to add a new control to a form, you must then edit the control properties to associate that control with an existing field.

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 click **Properties** from the shortcut menu.



2 In the **Properties Sheet**, edit the control properties. 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 Appendix D, *Form Controls*.

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. For more information, see *Deleting a Field* on page 108.

To delete a control from a form:

1. In the Workspace of ClearQuest Designer, expand **Record Types** or **Record Types – Stateless** until you see the form you want.
2. Double-click the form.
3. Select the control or controls to be deleted.
4. Click **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 following table 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.

Layout tool	Description
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.
Space evenly across	Arranges the selected controls horizontally so that there is an equal amount of space between all controls. The leftmost and rightmost 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 topmost and bottommost 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 either 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 click **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 click 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. In the Workspace of ClearQuest Designer, expand **Record Types** or **Record Types – Stateless** until you see the form you want.
2. Double-click the form.
3. Select the dialog tab containing the controls whose tab order you want to set.
4. Click **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.

Working with Tabs on Forms

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 to group the controls on the form. You can add as many tabs as you need.

Adding Tabs to a Form

To add a tab to a form:

1 In the Workspace of ClearQuest Designer, expand **Record Types** or **Record Types – Stateless** until you see the form you want.

2 Double-click the form.

3 Click **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 click **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 pressing 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 in ClearQuest Designer:

1 Right-click the tab on the form and click **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 reopen 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 click **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 reopen 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. In the Workspace of ClearQuest Designer, click **Record Types** or **Record Types – Stateless** until you see the form you want.
2. Double-click the form.
3. Click the tab and click **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 in ClearQuest Designer:

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

To paste the tab:

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

Reusing Record 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. After 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. For more information, see *Copying Tabs* on page 159.

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

To export a form:

1. In the Workspace of ClearQuest Designer, click **Record Types or Record Types - Stateless** until you see the form you want to export.
2. If the form is open, save the form by clicking its close box.
3. In the Workspace, right-click the form name and click **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 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 in ClearQuest Designer:

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 click **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.

Administering Users 8

Rational ClearQuest provides the User Administration Tool to set up and administer users and user groups. This includes the ability to create users and user groups, set user IDs and passwords, assign users to user groups, import and export users and user groups among schema, subscribe multiple users to databases, subscribe user groups to databases, and search for specific users by login name or full name.

ClearQuest stores information about users and user groups in the user database and in the schema repository.

The topics covered include:

- *Overview of User Administration* on page 161.
- *Working with Users* on page 164.
- *Restricting User Access to Actions* on page 174.
- *Exporting and Importing Users and User Groups* on page 175.
- *Administering Users in a MultiSite Environment* on page 176.

Note: To administer users and user groups, you must have User Administrator privileges. ClearQuest Designer includes a default User Name (*admin*) that you can use to get started. When ClearQuest is first installed, you can log in to this *admin* account without a password. The *admin* user account includes the Super User privileges that you need to perform many administrative functions. For more information, see *ClearQuest User Privileges* on page 163.

Overview of User Administration

To administer ClearQuest users, you use the User Administration dialog box. In this discussion, we access the User Adminstration Tool from the **Tools > User Administration** within the ClearQuest Designer.

You can also access the User Administration Tool from **Start > Programs > Rational Software > Rational ClearQuest.**

Use the User Administration dialog box to:

- Create new users and user groups and add users to user groups.
- 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 163. 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 174.
- 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 167.
- Export user data and import it into another schema repository. See *Exporting and Importing Users and User Groups* on page 175.

Note: Whenever you add or modify a user or user group, you must upgrade your user databases with the new information. See *Subscribing Users and Groups to Databases* on page 167.

Viewing Database Subscriptions

You can determine which users and user groups have been granted access to a specific database.

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

1. In ClearQuest Designer, click **Tools > User Administration**.
2. In the User Administration dialog box, click **DB Action > View Subscription**.
3. In the Subscribed Users and Groups dialog box, select a database. You will then be able to see which users and user groups have been granted access to this database.
4. Click **OK** to close the Subscribed Users and Groups dialog box.

ClearQuest User Privileges

ClearQuest supports the following privileges for users and user groups:

Privilege	Allows user or group to
Active User	A default privilege for all new ClearQuest users. 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. Log into ClearQuest Designer to view schemas and to view user administration information. Cannot edit schemas or change user information.
All Users/Groups Visible	A default privilege for all new ClearQuest users. Users can see information about all other users and groups. See *Administering Users in a MultiSite Environment* on page 176.
User Administrator	Perform all Active User tasks and: ▪ Use ClearQuest Designer to create users and user groups and assign and modify their user-access privileges. ▪ Create and modify user passwords including passwords of Super Users.
Dynamic List Administrator	Perform all Active User tasks and: ▪ Edit dynamic lists in ClearQuest client.
Public Folder Administrator	Perform all Active User tasks and: ▪ Create, modify, save, and delete public queries, charts, and reports in ClearQuest client.
SQL Editor	Perform all Active User tasks and: ▪ Edit SQL for queries in ClearQuest client and BuildSQL for API calls in hooks.
Security Administrator	Perform all Active User tasks and: ▪ Edit SQL for queries in ClearQuest client. ▪ Edit the context group list field for a security context record. ▪ View all records.

Privilege	Allows user or group to
Schema Designer	Perform all Active User tasks and: ▪ 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. **Note**: Cannot perform User Administrator tasks.
Super User	Perform all Active User, Schema Designer, User Administrator, Security Administrator, Public Folder Administrator, Dynamic List Administrator, and SQL Editor tasks and: ▪ Use ClearQuest Designer to create and delete databases and schemas. ▪ Edit ClearQuest Web settings. The admin user account that comes with ClearQuest Designer has Super User privilege.

Permissions granted supersede permissions denied. For example, a user will be able to access a specific database if he or she belongs to a group that has been granted access to that database, even if the user has not individually been granted access.

Working with Users

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

Adding a New User

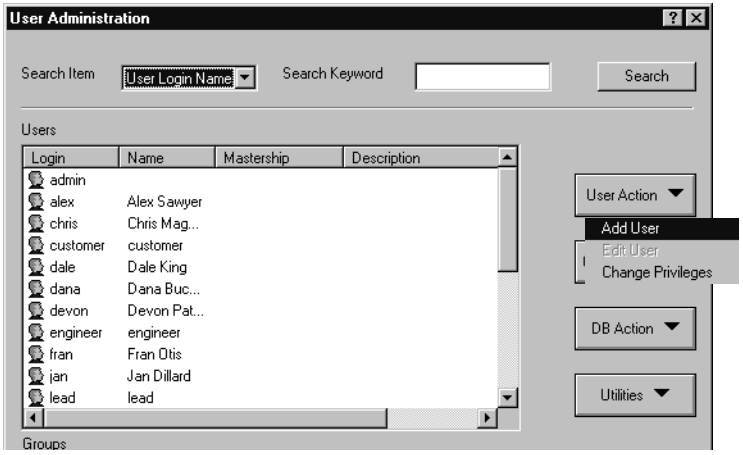
To add a new user:

1 In ClearQuest Designer, click **Tools > User Administration**.

2 In the User Administration dialog box, click **User Action > Add User**.

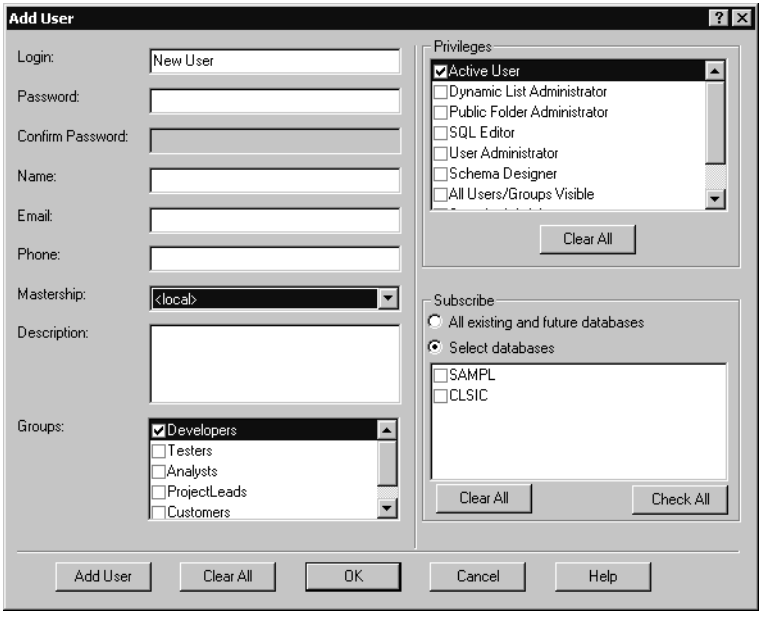
3 In the Add User dialog box, fill in the user information. Type the user's **Login**, **Password**, **Name**, **E-Mail** address, **Phone**, and if desired, a **Description**.

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: ! "" # $ % & ' () * + , . / : ; < = > ? @ [\ \ ] ^ ` { | } ~



Do not enter a hyphenated user name.

To use an ambiguous user name, use the keysite name as part of the user login name. If an ambiguous user name is used without the site-extension during login, you get an invalid Login error.



4 If you are using ClearQuest MultiSite, select the Mastership. For more information, see *Administering Users in a MultiSite Environment* on page 176.

5 Select the Groups to which the new user belongs. For more information, see *Working with User Groups* on page 172.

6 Select the Privileges for the new user. For more information, see *Assigning User Access Privileges* on page 166.

7 Select the databases to which the new user should be given access. You can select **All existing and future databases** to give access to all user databases in the system, or choose **Select databases** and then select individual databases for access.

8 Click **Add User** and click **OK**.

9 In the User Administration dialog box, click **DB Action > Upgrade** and click **OK** in the Upgrade dialog box, to add this information to the selected user databases. For more information, see *Applying Schema Changes to the User Database* on page 169.

Assigning User Access Privileges

You can assign user privileges to new users as you create them, and you can add or modify the privileges of existing users.

You must have a privilege before you can grant it to another user. By default, all users have the Active User privilege. For a description of access privileges, see *ClearQuest User Privileges* on page 163.

To assign user access privileges when you are creating a new user:

1 In ClearQuest Designer, click **Tools > User Administration**.

2 In the User Administration dialog box, click **User Action > Add User**.

3 In the Add User dialog box, check privileges for the user in the Privileges pane.

4 Click **OK**.

5 In the User Administration dialog box, select **DB Action > Upgrade** and click **OK** in the Upgrade dialog box to add this information to the user database. For more information, see *Applying Schema Changes to the User Database* on page 169.

To assign or modify user access privileges for an existing user:

1 In ClearQuest Designer, click **Tools > User Administration**.

2 In the User Administration dialog box, click **User Action > Change Privileges**.

3 Select the user and check or uncheck privileges for that user.

4 Click **OK**.

5 In the User Administration dialog box, click **DB Action > Upgrade** and click **OK** in the Upgrade dialog box, to add this information to the user database. For more information, see *Applying Schema Changes to the User Database* on page 169.

Subscribing Users and Groups to Databases

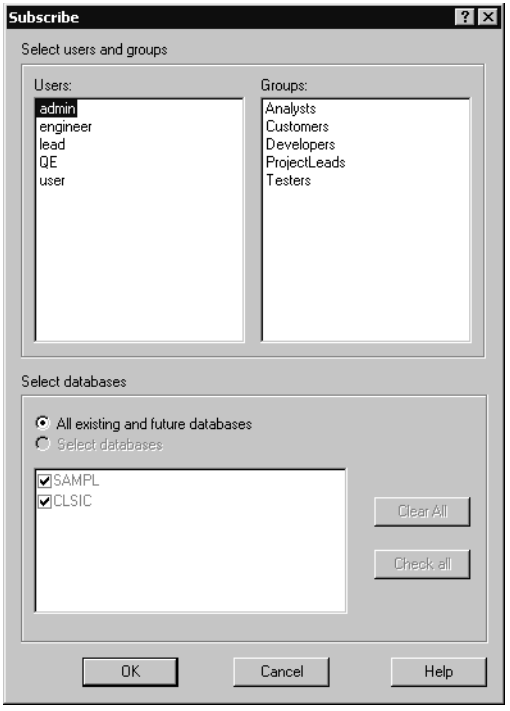
Subscribing a user or a user group to a database allows that user or the members of that group to access the database.

To subscribe users and user groups to selected databases:

1 In ClearQuest Designer, click **Tools > User Administration**.

2 In the User Administration dialog box, click **DB Action > Subscribe**.

3 In the Select Users and groups pane, select the users or user groups. In the Select Databases pane, select the databases to which the users should be given access. You can select **All existing and future databases** to give access to all user databases in the system, or check **Select databases** and select individual databases for access.



4 Click **OK**.

5 In the User Administration dialog box, select **DB Action > Upgrade** and click **OK** in the Upgrade dialog box, to add this information to the user database. For more information, see *Applying Schema Changes to the User Database* on page 169.

Unsubscribing Users and Groups from Databases

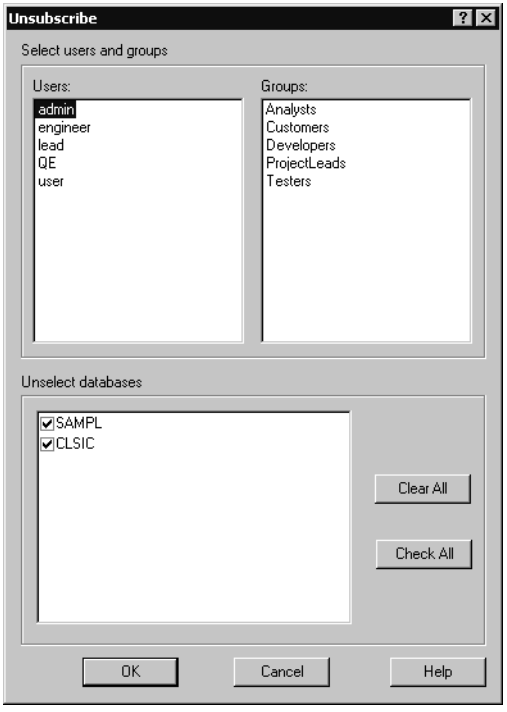
Unsubscribing a user or a user group from a database removes that user or group access to the database.

To unsubscribe users or user groups to selected databases:

1 In ClearQuest Designer, click **Tools > User Administration**.

2 In the User Administration dialog box, click **DB Action > Unsubscribe**.

3 In the Unsubscribe dialog box, select the users or user groups and clear the databases from which you want access revoked.



4 Click **OK**.

5 In the User Administration dialog box, select **DB Action > Upgrade** and click **OK** in the Upgrade dialog box**,** to add this information to the user database. For more information, see *Applying Schema Changes to the User Database* on page 169.

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 databases.

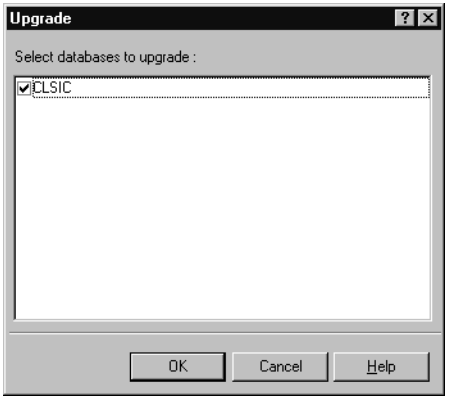
Warning: Before applying changes to a user database, make sure that there are no users logged in to the database. If users are connected to the user database, the upgrade will fail. ClearQuest prevents new users from logging in to a database while you are updating it, but it does not disconnect users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

To apply the schema changes to the user database:

1 In ClearQuest Designer, click **Tools > User Administration**.

2 In the User Administration dialog box, click **DB Action > Upgrade**.

3 In the **Upgrade** dialog box, select the databases to which you want the changes applied and click **OK**.



4 Click **OK** to close the User Administration dialog box.

Editing Users

To edit a user profile:

1 Log in to ClearQuest Designer with User Administrator or Super User privileges and click **Tools > User Administration**.

2 In the User Administration dialog box, click **User Action > Edit**.

3 Edit the user profile as desired. For more information, see *Adding a New User* on page 164.

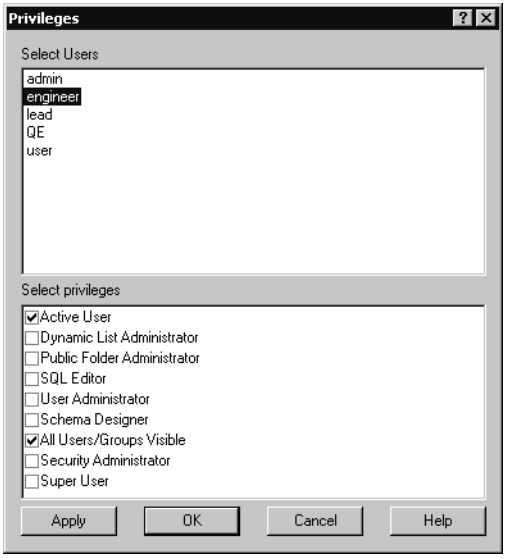
Note: ClearQuest users can edit some of their own user information from the ClearQuest client. See *Editing a User Profile from the ClearQuest Client* on page 170.

Changing User Privileges

To change user privileges:

1 In the User Administration dialog box, select the user and click **User Action > Change Privileges**.

2 In the Privileges dialog box, select a user and select or unselect the privileges you want.



3 Click **Apply** and **OK**.

4 In the User Administration dialog box, select **DB Action > Upgrade** and click **OK** in the Upgrade dialog box**,** to add this information to the user database. For more information, see *Applying Schema Changes to the User Database* on page 169.

Editing a User Profile from the ClearQuest Client

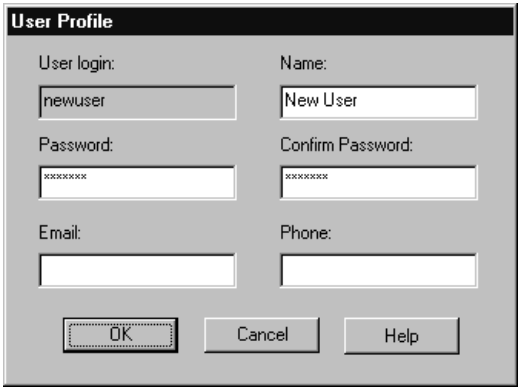
Active users can change some of their own user information, including their name, e-mail address, password, and telephone number, from the ClearQuest client. However, active users cannot change their own user login or privileges.

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

To modify a user profile from the ClearQuest client:

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

2 Make changes in the appropriate fields.



3 Click **OK.**

Making Users and Groups Inactive

You can make users and user groups inactive, which means they cannot log in to 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 to which the users were subscribed.

To make a user or user group inactive:

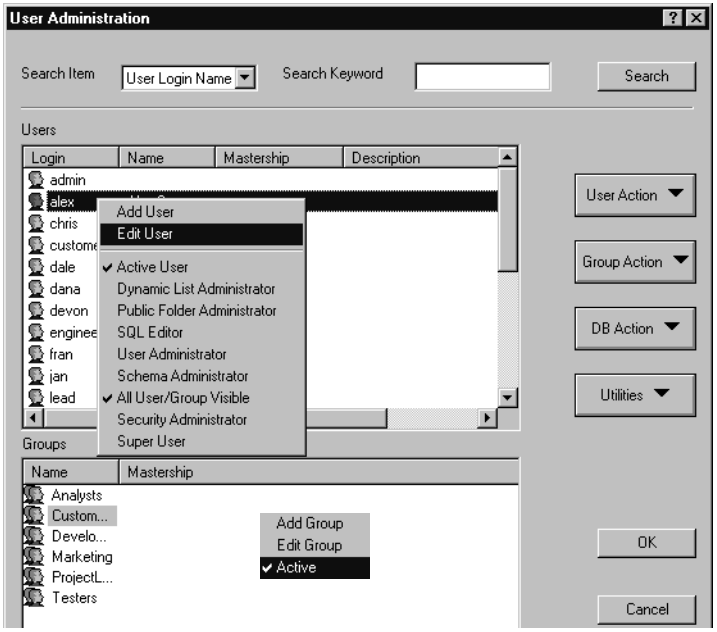
1 In ClearQuest Designer, click **Tools > User Administration**.

2 In the User Administration dialog box:

- To make a user inactive, right-click the user and use the shortcut menu to clear all user privileges, including **Active User**.
- To make a group inactive, right-click the group and clear **Active** in the shortcut menu.

3 Select **DB Action > Upgrade**, then click **OK** in the Upgrade dialog box, to add this information to the user database. For more information, see *Applying Schema Changes to the User Database* on page 169.

To make a user or user group active again, select **Active User** for the user or **Active** for the group, then select **DB Action > Upgrade** and click **OK** in the Upgrade dialog box.



Working with User Groups

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

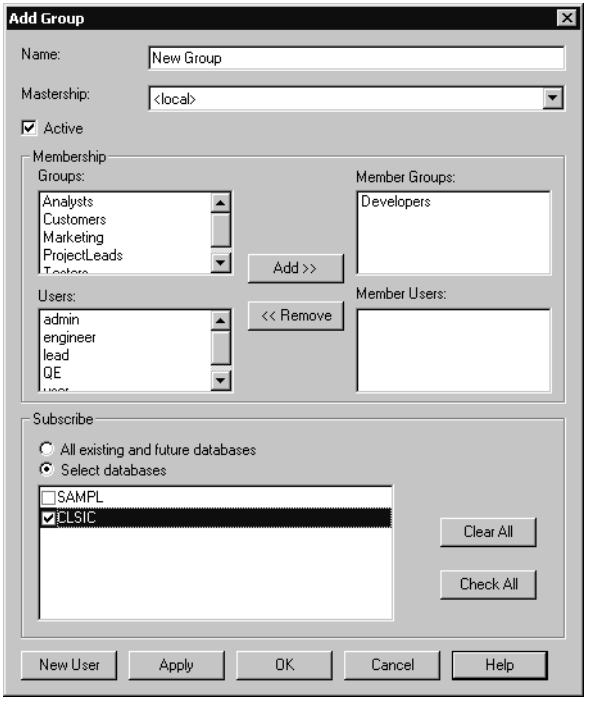
- Create groups such as *Software Engineers* or *Managers*, and subscribe them to different databases.
- Base an e-mail rule on a group (for example, notify the *Quality Assurance* group whenever a user submits a new defect).
- Write hook scripts or external applications with conditional logic for groups (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, click **Tools > User Administration**.
2. In the User Administration dialog box, click **Group Action > Add Group**.
3. In the Group dialog box, type a **Name** for the new group.

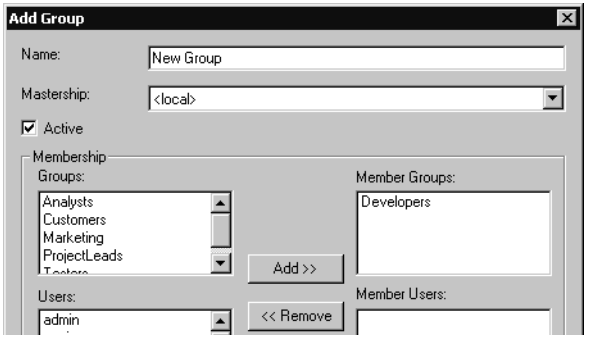
4 If you are using ClearQuest MultiSite, select the **Mastership**.



Creating Subgroups

To add subgroups to a group:

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



Adding Users to a Group

To add users to a group:

- In the Add Group dialog box, select the users 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 or Subgroups from a Group

To remove users or subgroups from a group:

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

Subscribing User Groups to Databases

To subscribe a group to databases:

1. In the Add Group dialog box, select the databases to which the users should be given access. You can check the **All existing and future databases** option to give access to all user databases in the system, or check the **Select databases** option and then select individual databases.
2. Click **Apply** to close the Group dialog box.
3. In the User Administration dialog box, click **DB Action > Upgrade** and click **OK** in the Upgrade dialog box**,** to add the information to the user database. For more information, see *Applying Schema Changes to the User Database* on page 169.

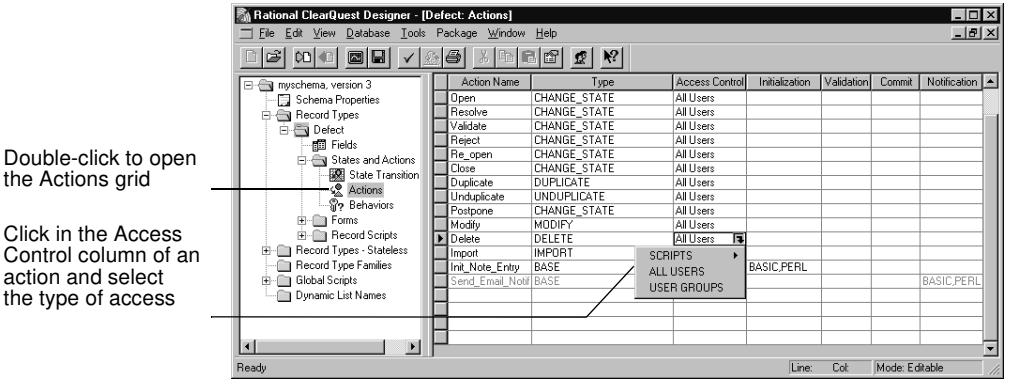
See also, *Subscribing Users and Groups to Databases* on page 167.

Restricting User Access to Actions

By default, all ClearQuest client users can perform all actions on records. You can narrow 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 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 *Using Hooks to Customize Your Workflow* on page 191.
- 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.

Note: To execute a primary action (modify, submit, delete, import), the current user must be in the access control list for the primary action as well as for all the base actions.

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, click **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 export this text file to another schema repository.

To import users and user groups:

1 In ClearQuest Designer, click **Tools > User Administration** to open the User Administration dialog box.

2 In the User Administration dialog box, click **Utilities** and select **Import**.

Select the file containing the user and group information you want to import and click **Open**.

Note: If you import a profile with users or user groups that are subscribed to a specific database, and the schema into which the profile is being imported does not include that specific database, then the import will fail. One way to avoid this problem is to make sure that all users and groups are subscribed to the **All existing and future databases** option.

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
- Establishing 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. For example, if a user has SQL Editor privileges on the local replica, that user has SQL Editor privileges on all replicas, as long as 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 in the following list 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 the following 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 that 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.

Establishing 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 mastered, look at the **Mastership** column in the User Administration dialog box.

Changing the Mastership of a User

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

Changing the mastership of a user involves three tasks:

1. Changing Mastership. Do this at the site that currently has mastership of the user.
2. Synchronizing the Replicas After Mastership Changes.
3. Upgrading the User Database with the Mastership Change. Do this at the new mastering site and all other sites.

Changing Mastership

To change the mastership of a user:

1. In the User Administration dialog box, select a user and select **User Action > Edit User** to open the User Properties dialog box.
2. Select a replica from the Mastership list and click **OK**.
3. Notify the user administrator at the new site that mastership of a user has changed. The user administrator at the new site must receive and apply the syncreplica packet and then, upgrade the user database with the new user information.

Synchronizing the Replicas After Mastership Changes

See *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 169.

Using Security in Rational ClearQuest

9

Rational ClearQuest security features enable you to protect your data and ensure that only authorized users view or change records. You can hide specific records from some users while allowing other users to view or change the same records.

These security features are particularly valuable when more than one user group has access to the same database; for example when 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.

The topics covered include:

- *Hiding Records in ClearQuest* on page 179.
- *Security Example* on page 184.
- *Using Other ClearQuest Security Features* on page 188.

To implement ClearQuest security features, you must 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. For more information, see Chapter 4, *Working with ClearQuest Schemas* and Chapter 5, *Customizing a Schema*.
- Administer users and user groups. For more information, see *Working with Users* on page 164.
- Add fields to record forms. For more information, see *Adding Controls to a Form* on page 148.

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 done by placing a *security context field* in the record type of the records to which you want to restrict access. The

security context field references a security context record containing data that determines which users can see or change the controlled record.

For example, 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 are able to see the controlled record.

Hiding records in ClearQuest involves the following:

- 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. 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** check box 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.

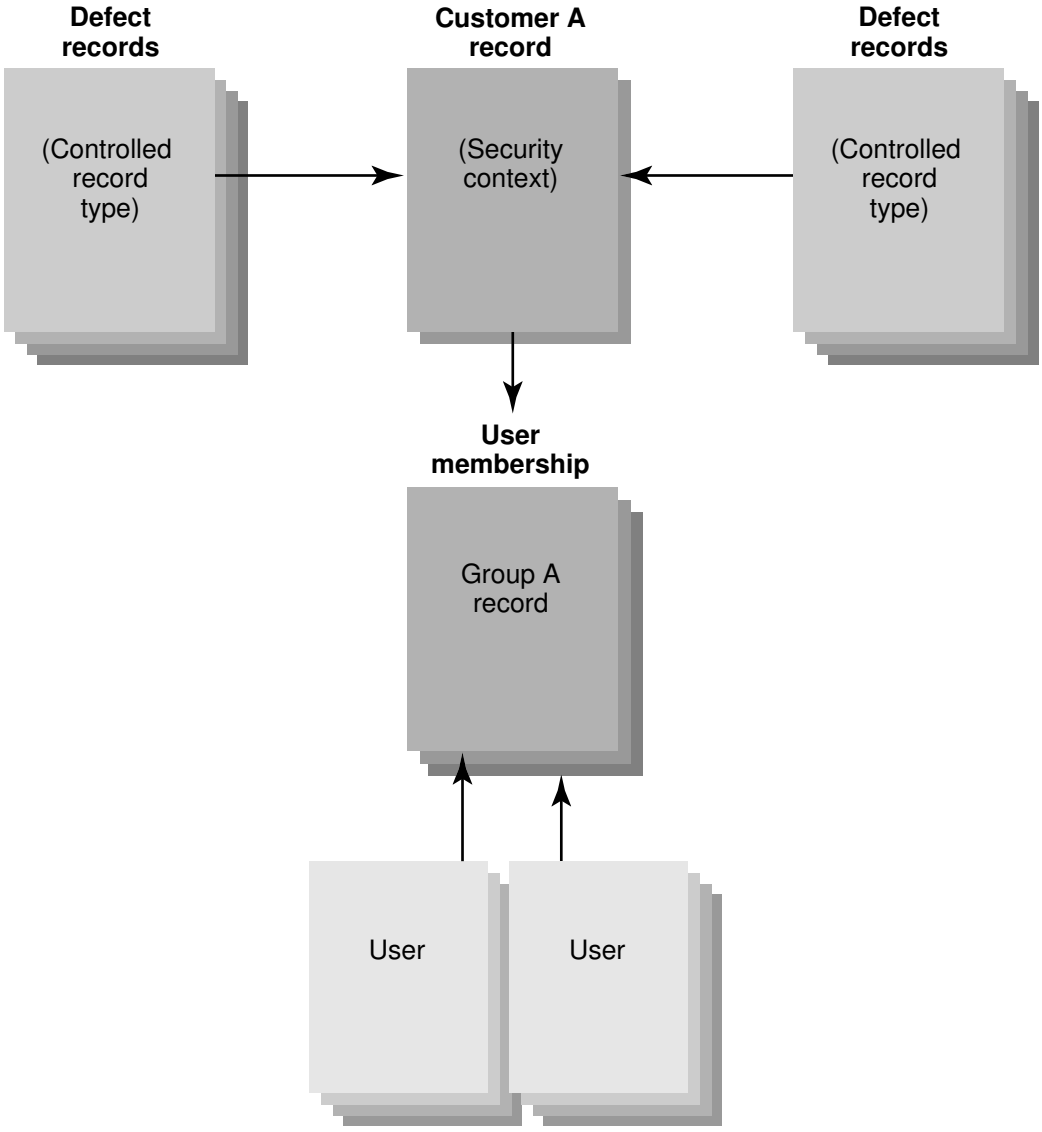
Editing Records to Allow User Access

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

Figure 6 shows 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.

Figure 6 Elements of a Security System



In ClearQuest, if you implement the design shown in Figure 6, this is what the end user will experience: User A logs into ClearQuest. User A is a member of Group A, which is associated with Customer A. When User A runs a query or requests a record, ClearQuest automatically filters all defects based on the security context of Customer A. Only defects belonging to Customer A are 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.

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 following example. You must have Super User privileges to complete 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

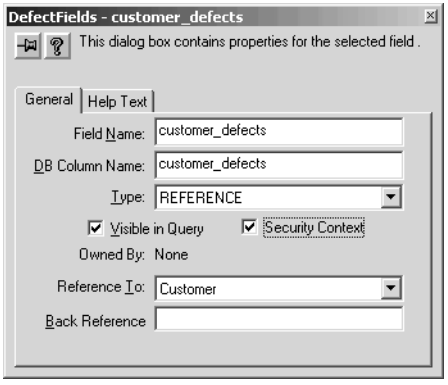
Note: You must have Schema Designer privileges to create a new field and add the field to the record form.

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**.
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.)
3. Right-click the new customer_defects field and click **Field Properties**.
4. In the Field Properties dialog box, select the **Customer** record from the **Reference To** list.



Clicking the Customer record type from the Reference To list, enables the **Security Context** check box.

5. Check the **Security Context** check box.

When you check **Security Context**, ClearQuest Designer adds a tab called **RatI_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 **RatI_Security** tab. For more information, see *Changing the Name of a Tab on page 157*.)

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 Other ClearQuest Security Features* on page 188.

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, click **Record Types > Defect > Forms** and double-click **Defect_Base**.
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. After you perform these steps, these changes cannot be undone. For more information, see *Procedures for Modifying a Schema* on page 80.

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 172 and *Adding Users to a Group* on page 174.

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. Click **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. Click **Action > Modify**.
3. On the **Ratl_Security** tab, select the Widgets Inc. group and click **Add**. Click **Apply**.
4. Repeat Step 2 and Step 3 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 gives 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 on to the database containing the defect records you want to control.
2. Run a query on All Defects to see all 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 Other ClearQuest 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*
- *Restrict 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

10

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

- *Understanding Hooks* on page 192.
- *Working with Field Hooks* on page 194.
- *Working with Action Hooks* on page 204.
- *Execution Order of Field and Action Hooks* on page 208.
- *Working with Record Scripts* on page 210.
- *Working with Global Scripts* on page 214.
- *Writing External Applications* on page 217.
- *Using the ClearQuest API* on page 217.
- *Common API Calls* on page 218.
- *Finding Text in Hook Scripts* on page 220.

See also

- For more information on the ClearQuest API, click **Start > Programs > Rational Software > Rational ClearQuest > Rational ClearQuest API Reference**. The *API Reference* for Rational ClearQuest is an online reference guide for all ClearQuest API calls.
- To learn about using hooks in the Web environment, see Chapter 11, *Administering ClearQuest Web*.
- The ClearQuest Sample Hooks Repository provides a place for users to trade hook scripts with one another. The Repository is located at http://clearquest.rational.com/. To gain access to the database, enter:

  username: hooks
  password: password

  Select the link for the Sample Hooks Database. From there you can browse the database of existing hooks scripts.

Understanding Hooks

Hooks are entry points, like triggers, for pieces of code that ClearQuest runs 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 lifecycle 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 85.
- 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, be sure to test them in the Web environment. See *Using Hooks to Detect a Web Session* on page 232.
- 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.

  Note: To obtain verbose debugging output when an exception occurs during hook execution, create the Windows registry key HKEY_CURRENT_USER\SOFTWARE**RationalSoftware\ClearQuest\2.0\Common\HookDebug** with a DWORD value of 1. To revert to the default (less verbose) hook exception debug message, change this value to 0.

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 85).

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 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 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 run 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
idName = GetFieldValue("id").GetValue
set currentObj = curSession.GetEntity("defect", idName)
' Mark the entity with ID="SAMPL00000031" as a duplicate of this entity.
' Use the action named "duplicate".
set dupEntityObj = curSession.GetEntity("defect", "SAMPL00000031")
curSession.MarkEntityAsDuplicate dupEntityObj, currentObj, "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	Description
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 199.
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 *API Reference* for Rational ClearQuest 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).

See also

- *Customizing Fields by Adding Hooks* on page 112
- *Adding a Field Hook* on page 195
- *Execution Order of Field and Action Hooks* on page 208
- *Field Hook Examples* on page 316

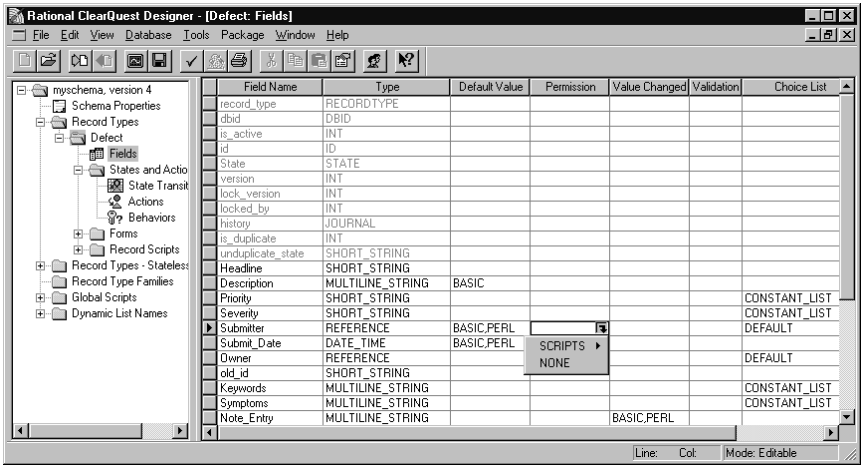
Adding a Field Hook

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 runs your hooks based on the rules specified in the section *Execution Order of Field and Action Hooks* on page 208.

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 box.

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 Click **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 your script compiles correctly. ClearQuest Designer saves your changes.

Deleting a Field Hook

To delete 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 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.

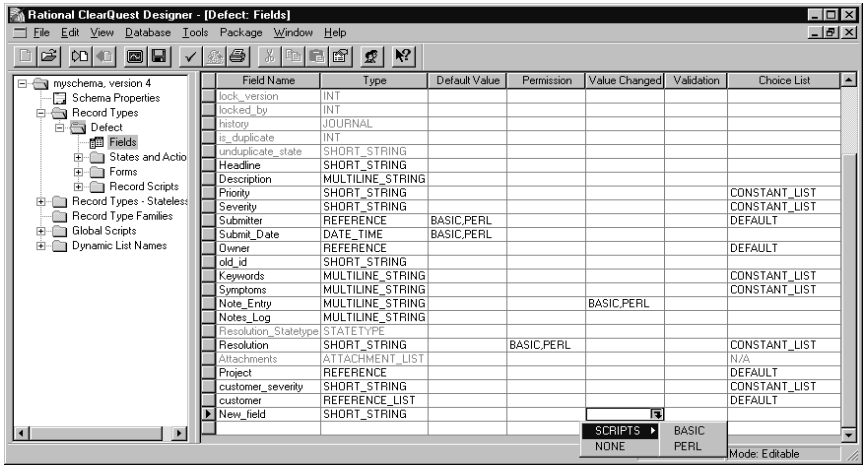
Creating a Dependency Between Fields

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 and click **SCRIPTS > BASIC** or **SCRIPTS > Perl**.

If Instant Edit Mode is enabled, ClearQuest Designer automatically opens the Script editor. To enable or disable instant editing, click **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 327.)

5 When you finish editing your script, click **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 about controls, see *Working with Form Controls* on page 146.

2 After adding the controls to the record form, right-click the control for the parent field and click **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 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 317.
- 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 199.
- 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

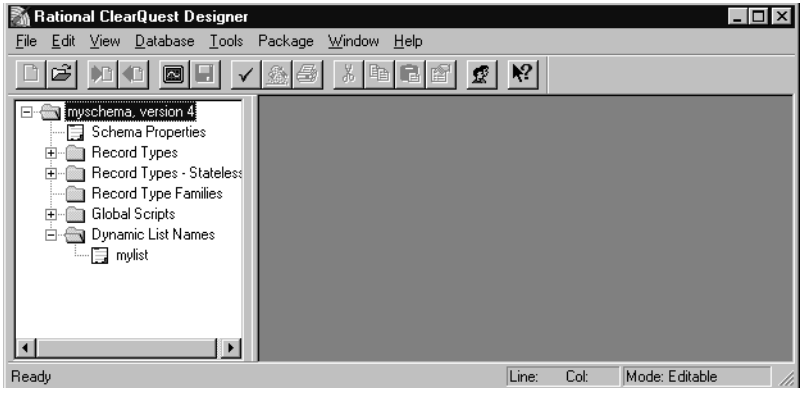
A dynamic list can be a drop-down list, such as a list of states. Dynamic lists are helpful when you want to add values to a list without editing the schema. Users with Schema Designer, Dynamic List Administrator, or Super User privileges can add values to a dynamic choice list from any ClearQuest client.

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.

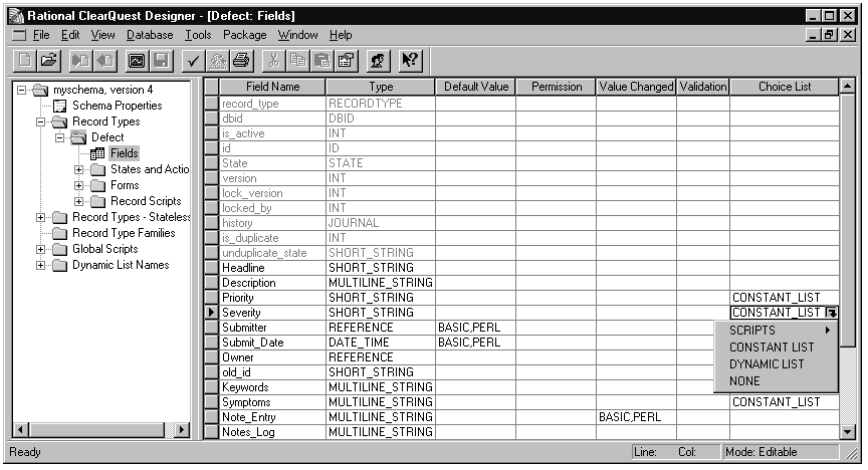
To create a dynamic list for a field you first create the list and associate it with a field in ClearQuest Designer, then define the values for the list in ClearQuest client:

1 In the Workspace, right-click **Dynamic List Names** and click **Add** from the shortcut menu.

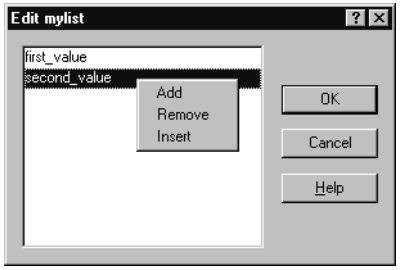
2 Type a name for the dynamic list.



3 In the Fields grid, click the Choice List cell for the desired field, click the drop-down arrow 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 202.

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 *Procedures for Modifying a Schema* on page 80.

7 To define the values for a dynamic list, use ClearQuest client. See *Editing a Dynamic List* on page 201.

Editing a Dynamic List

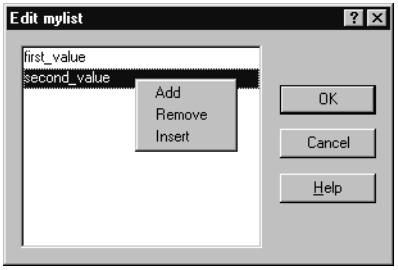
You first add a dynamic list in ClearQuest Designer (see *Creating a Dynamic Choice List* on page 199), then you define the values of the dynamic list in ClearQuest client.

You must have Schema Designer, Dynamic List Administrator, or Super User privileges to edit a dynamic list. (See *ClearQuest User Privileges* on page 163.)

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 Click **OK**.

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 Examples* on page 320.

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 (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 run.

Action hook	Use	When run
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. Note: To run a primary action (modify, submit, delete, import), the current user must be in the access control list for the primary action as well as for all the base actions. See *Restricting User Access to Actions* on page 174.	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 runs this hook before committing any changes to the database. Validation hooks must use a script. See *Validating User Input in a Field* on page 203.	When the user commits the action.

Action hook	Use	When run
Commit	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). 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. You can write a commit hook as a VBScript or Perl subroutine.	Just before ClearQuest commits the transaction to the database.
Notification	Starts a post-commit action that notifies users when an action is performed. See Chapter 12, *Administering ClearQuest E-Mail*. Notification hooks must use a script.	After ClearQuest commits the transaction.

See also

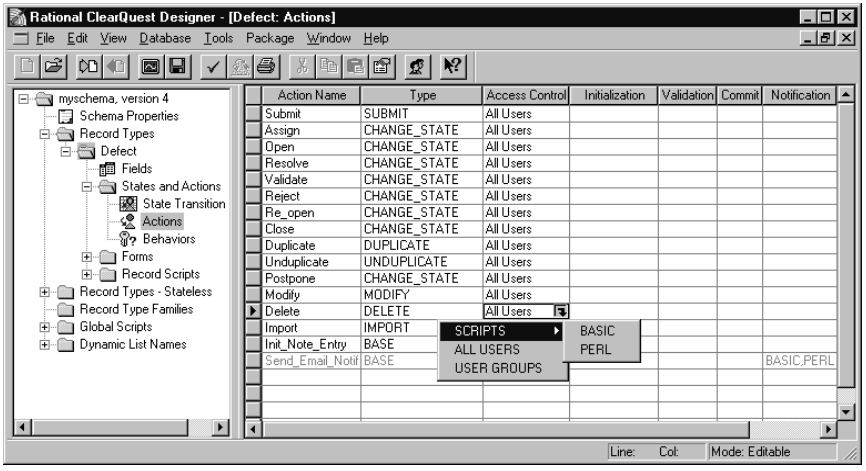
- *Execution Order of Field and Action Hooks* on page 208
- *Action Hook Examples* on page 325

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 you choose **User Groups**, make sure groups are subscribed to all databases the schema uses.

   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 edit 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: After 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 runs 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 runs 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 nonexistent 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 runs 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 runs 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 to Detect a Web Session* on page 232.

When the Record Is Validated

Before committing changes to the database, ClearQuest validates the record by running 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 running 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 ClearQuest 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 *API Reference* for Rational ClearQuest.

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 use 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 the appropriate data. If the 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 212.) 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 run. 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 run 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 193
- *Operating Context for Using Scripts* on page 194
- *Record Script Example* on page 340

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 do not 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 ran successfully.
- To enable record hooks for the Web, check **Enable for web** on the **Extended** tab of the control's property sheet.
- In the Web, record hooks run on the ClearQuest Web server. Because they execute on the server, you should not call Windows routines that cause a window to display, as they will 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-enabled.

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	More information
Push button	AD_BUTTON_CLICK	Button name
Combo box	AD_CONTEXMENU_ITEM_SELECTION	Null string
Drop-down list box	AD_CONTEXMENU_ITEM_SELECTION	Null string
List box	AD_CONTEXMENU_ITEM_SELECTION	Current field value selection
List view	AD_CONTEXMENU_ITEM_SELECTION	Current field value selection

Control type	Supported event type	More information
Text box	AD_CONTEXMENU_ITEM_SELECTION	Current field value selection
Drop-down combo box	AD_CONTEXMENU_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 121.

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 click **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, click **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 212.

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:

   ```
   REM added your hook code here
   ```

5. When you are finished editing your code, click **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.

Note: If actions are using a record script that you are trying to delete, you get a warning. The delete fails if actions are associated with 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 the ClearQuest 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.
- You should only use functions in a global hook, since all code in a global script is run unless it is in a function.
- See *Writing Scripts* on page 193 and *Operating Context for Using Scripts* on page 194.

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 click **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:

```
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, click **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:

```
REM TODO -- put your script code here
```

4 When you are finished editing, click **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.

Note: You do not get a warning if the script you are deleting is still being used in hook code. Deleting the script and not removing the calls from hooks causes errors when a user performs an action or change that calls the deleted 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, run 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 217.

Note: ClearQuest includes a 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 *API Reference* for Rational ClearQuest.

Using the ClearQuest API

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

See the online *API Reference* for Rational ClearQuest 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 *API Reference* for Rational ClearQuest.

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.
- Run 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 *API Reference* for Rational ClearQuest.

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 *API Reference* for Rational ClearQuest.

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 *API Reference* for Rational ClearQuest.

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.

API Call (VBScript/Perl)	Function
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.
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.

API Call (VBScript/Perl)	Function
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.

Note: If you have a form open, close it before perfoming the following steps.

To find text in hook scripts:

1. Open the schema in which you want to search for text.
2. Click **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.

Administering ClearQuest Web

11

Rational ClearQuest Web allows users to access ClearQuest data from a Web browser. The topics covered include:

- *ClearQuest Web Considerations* on page 223.
- *Customizing ClearQuest Web* on page 224.
- *Limiting Access to ClearQuest Web* on page 230.
- *Using Hooks in ClearQuest Web* on page 231.

For detailed procedures about how set up the ClearQuest Web client, see the *Installation Guide* for Rational Server Products.

ClearQuest Web Considerations

It is important to note that the ClearQuest integrations are not applied to the Web client; therefore, fields, forms, reports, scripts and other functionality added to the ClearQuest Client on either Windows or UNIX by an integration package will not be available through the Web client.

You should advise your ClearQuest Web users of the following:

- 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.

Note: 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. When you design a form, be sure to test it in ClearQuest Web.

ClearQuest Data Code Page and ClearQuest Web

The Rational ClearQuest Web, UNIX, and Windows clients offer different levels of protection against data corruption and divergence. Because the Web client cannot determine the operating system code page used by the Web browser, there is limited protection against invalid code page characters when you use the Web interface. When you use the Web client instead of a native client, there is no guarantee that invalid characters will not enter your database and cause data corruption.

Therefore, when you use the ClearQuest Web interface, you should input only characters that are supported by the ClearQuest data code page value of the ClearQuest user database. If you type unsupported characters, then those characters may be rejected or corrupted.

Customizing ClearQuest Web

Note: To edit Web settings, you must have Super User or Schema Designer privileges.

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.

Web settings include:

- *ClearQuest Web Settings* on page 224.
- *Enable E-mail Notification Settings* on page 227.
- *Presentation Scheme Settings* on page 227.

To edit the ClearQuest Web settings:

1. Log on to the ClearQuest client with Super User or Schema Designer privileges and click **Operations > Edit Web Settings**.
2. Edit the Web settings as needed.
3. When you are done, click **Apply** to apply your changes and keep the Web settings window open, or click **OK** to save the changes and close the edit Web settings window.

ClearQuest Web Settings

ClearQuest Web settings include miscellaneous settings such as access privileges, dynamic HTML menu toggles, time-out settings, and more. Each Web setting is described in the following table.

Setting	Description
Restricted Access WebSite	Select this check box to restrict access for all users.
Restricted Access UserNames	Indicate which users you want to give only restricted access. User names must be separated by commas. If you enter spaces, the names are ignored. These users can only submit defects and run the query specified in Restricted Access Query Name.
Restricted Access User Groups	Indicate which groups of users you want to give only restricted access. User group names must be separated by commas. If you enter spaces, the names are ignored. These users can only submit defects and run the query specified in Restricted Access Query Name.
Restricted Access Query Name	Indicate which query is available to users with restricted access. The query listed here is the only query users with restricted access can run. Often this query uses a Submitter = CurrentUser filter so that every user can see only the records he or she has submitted, but it can be any query. ClearQuest is automatically restricted when: 1) the ClearQuest site, ClearQuest users, or ClearQuest groups are specified in Web Settings as Restricted, or 2) ClearQuest is run unlicensed.
Allow Find Record When Restricted	Set to False if you do not want to allow users with restricted access to search the database.
Allow Advanced Query Editor	Set to False to prevent users from creating advanced queries, (see *Working with Queries* on page 218).
Use DHTML Menus	Set to False to turn off menu ability in the main toolbar. When set to false, a list of menu options is available. You may need to set this to false if your Web browser does not support dynamic HTML.
Show Read Only Tabs When Update	Set to False if you want to decrease the information displayed in each record being modified and allow faster scrolling. Read only tabs will not appear.
Splash Image	Enter the name of the file containing the Web site's splash image. This is the main product page that appears after logging in.
Client Side Cursors for Oracle	Set to True if you are using an Oracle database and reports will not generate.
CCIntegration Sessions persist across calls	Do not change unless directed by ClearQuest technical support.

Setting	Description
Default Field Script Display Mode	Change to set the default script mode for all users the first time they open a form with a script. Users can change this setting for their own sessions. See *Running Field Scripts* on page 226 for more information about this setting.
Login Locking Scheme	Do not change unless directed by ClearQuest technical support.
Login Timeout	This login timeout setting specifies the time to wait (in milliseconds) on a global lock before giving up. The default value is 15000 (or fifteen seconds).
Debug Trace Level	Set to 0 unless you need to debug a problem in the Active Server Pages (ASP).

Running Field Scripts

ClearQuest uses ActiveX field controls on forms to display information from other databases. ActiveX controls are implemented in ClearQuest through the Microsoft Script Control and a user-defined Active Server Pages (ASP) script stored in the ClearQuest Server application. The ClearQuest administrator is responsible for writing the script that implements the ActiveX control on the Web.

When you open a form with an ActiveX control, the field's script generates and displays information about the data associated with the field. The information generated and displayed depends on the script. For example, if a defect has an attached file associated with it, the script could display the file.

Because running a script can be time-consuming and you do not always need to see the information generated, TestExpert lets you choose whether or not to run the scripts.

You make your choice at the end of the first field on a form generated by an external ASP script. Your selection for this field applies to all such fields on the form. The choices are:

- **All** if you want fields with scripts to run and generate updated information on read-only forms and modifiable forms.
- **Read-Only** if you do not want the script to run and provide updated information. This is the default setting.
- **No** if you do not want fields with scripts to run on either read-only forms or modifiable forms. This option is useful when you are viewing forms and do not need the latest information.

Click the button next to the option you want. For example, if you click **All**, the message at the end of the first field reads:

> You have opted to compute this field on All forms. You may compute it on Read-Only or No forms by selecting the appropriate button here.

Note: Contact Rational technical support for more information about using ActiveX controls with ClearQuest.

Enable E-mail Notification Settings

ClearQuest Web Enable E-mail Notification settings are described in the following table.

Setting	Description
Send Active	Enter 1 to enable e-mail notification. Enter 0 to disable it.
Mail Transport	Specify SMTP or MAPI depending on your mail transport type.
MAPI Server	If you are using MAPI, enter the mail server name.
MAPI Profile	If you are using MAPI, enter the profile alias name.
SMTP Host	If you are using SMTP, enter the host name.
Real Names	Enter your Web server machine name. If you enter a Web server name, e-mail messages display the server name in the **From** field. If you do not want the server name to display in all e-mail messages, do not enter anything here.
Reply To	Enter a generic, reply-to address for this Web server. This address automatically displays in the Reply To field of each e-mail message. If you leave it blank, no address appears.

Note: For automatic e-mail notification to work, the e-mail package must be installed in your schema and each ClearQuest client must be enabled to send e-mail messages.

Presentation Scheme Settings

The Presentation Scheme settings include general display options such as colors, borders, field widths, fonts, and more. These settings can be individually changed or you can select a predefined presentation scheme to apply, see *Working with Predefined Presentation Schemes* on page 229.

Each presentation scheme setting is described in the following table.

Setting	Description
Use Single Column Layout	Set to True if you want to display records as displayed in legacy versions of the product.
Single Column for Netscape 6	Set to True if you are using Netscape 6 and if you want to display records as displayed in legacy versions of the product.
Table Border Width	Enter the width of the record tables.
Text Field Width	If using single column layout, enter the width of a text field in pixels.
Text Field Height	If using single column layout, enter the height of a text field in pixels.
Vertical Grid Tolerance	If using a multicolumn layout, enter the spacing tolerance between items on the same line in a form.
Null Field String	Enter the string to be used for null fields.
Menu Frame Width	Enter the width for the Workspace on the main page.
Title Frame Height	Enter the height for the Rational toolbar at the top of the main page.
Table Heading Background Color	Select a color from the color table below the Settings Table.
Table Background Color	Select a color from the color table below the Settings Table.
Page Background Color	Select a color from the color table below the Settings Table.
Page Text Color	Select a color from the color table below the Settings Table.
Page Link Color	Select a color from the color table below the Settings Table.
Page Visited Link Color	Select a color from the color table below the Settings Table.
Menu Title Font Color	Select a color from the color table below the Settings Table.
Menu Title Font Face	Enter a list of fonts that can be used for menu titles.
Title Text Font Color	Select a color from the color table below the Settings Table.
Title Text Font Face	Enter a list of fonts that can be used for field titles.

Setting	Description
Dialog Heading Background Color	Select a color from the color table below the Settings Table.
Dialog Heading Text Color	Select a color from the color table below the Settings Table.
Dialog Background Color	Select a color from the color table below the Settings Table.
Dialog Text Color	Select a color from the color table below the Settings Table.

Working with Predefined Presentation Schemes

ClearQuest includes several predefined presentation schemes (name sets). You can use the buttons at the bottom of the Presentation Scheme settings to modify the predefined schemes in the following ways:

- *Editing a Presentation Scheme* on page 229.
- *Applying a Different Presentation Scheme* on page 230.
- *Deleting a Presentation Scheme* on page 230.
- *Renaming a Presentation Scheme* on page 230.
- *Creating a New Presentation Scheme* on page 230.

Using a Predefined Presentation Scheme

To use a predefined presentation scheme:

1 Click a presentation scheme from the list and click **Load**.

2 Click **Apply**.

Editing a Presentation Scheme

To edit a presentation scheme:

1 Click a presentation scheme from the list and click **Load**.

2 Edit the presentation settings as wanted.

Note: For options involving colors, select the type of color palette you want to view. The ClearQuest Color Palette window appears. Copy a color number and paste it into the Presentation Scheme option table.

3 Click **Apply**.

Applying a Different Presentation Scheme

To apply a different presentation scheme:

1. Click a presentation scheme from the list and click **Load**.
2. Click **Apply**.

Deleting a Presentation Scheme

To delete a presentation scheme:

- Click a presentation scheme from the list and click **Delete**.

Renaming a Presentation Scheme

To rename a presentation scheme:

1. Click a presentation scheme from the list.
2. Type a new name in the text box and click **Rename**.

Creating a New Presentation Scheme

To create a new presentation scheme:

1. Click an existing presentation scheme from the list on which to base the new scheme.
2. Type a new name in the text box and click **Save As**.

Limiting Access to ClearQuest Web

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

- Submitting records.
- Running 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 172) 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, in ClearQuest Web click **Operations > Edit Web Settings**. You must have Super User privileges to configure Web entries.

For more information, click **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:

- You must enable dependent fields for ClearQuest Web. See *Enabling Dependent Fields for ClearQuest Web* on page 231.
- 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 232.

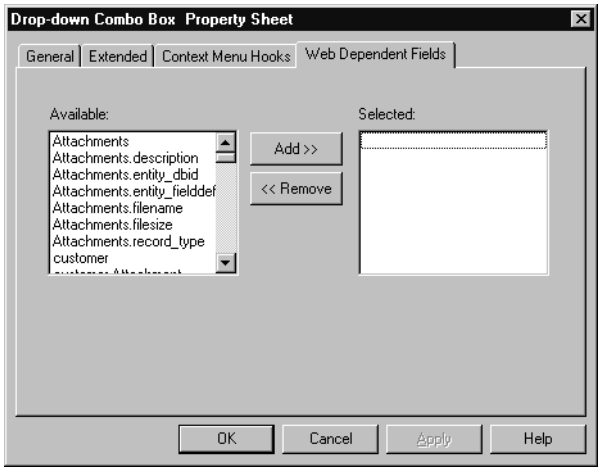
Enabling Dependent Fields for ClearQuest Web

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

   See *Working with Form Controls* on page 146.
2. After adding the controls to the record form, right-click the control for the parent field and click **Properties** from the shortcut menu.
3. In the Web Dependent Fields tab, click the appropriate child fields from the **Available** list and add them to the **Selected** list.
4. Right-click the control on the form and click **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, click **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. Because 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
}
```

Administering ClearQuest E-Mail

12

This material describes how to enable Rational ClearQuest e-mail capabilities.

The topics covered include:

- *Overview of ClearQuest E-Mail Features* on page 235.
- *Enabling Automatic E-Mail Notification* on page 236.
- *Using an Action Hook to Send E-Mail* on page 246.
- *Enabling E-Mail Submission Through the Rational E-Mail Reader* on page 247.
- *Formatting E-Mail for Submission* on page 254.

Overview of ClearQuest E-Mail Features

ClearQuest supports the following e-mail capabilities:

- **Automatic e-mail notification:** ClearQuest can automatically send e-mail notification to users or user groups when a record meets the criteria you define. For example, ClearQuest can send an e-mail message to your testing team whenever a defect is fixed. See *Enabling Automatic E-Mail Notification* on page 236.
- **Round-trip e-mail:** You can combine e-mail notification with e-mail submission to update records by using *round-trip* e-mail. For example, when a user receives an automatic e-mail notification that a new record has been submitted, that user can respond with an e-mail message that appends notes to the record in the database. See *Using Round-Trip E-Mail* on page 245.
- **E-mail notification action hook:** You can add hook code that creates and sends an e-mail message when a user completes an action. See *Using an Action Hook to Send E-Mail* on page 246.
- **Submitting and modifying defects by e-mail:** ClearQuest users can submit and modify records by using e-mail without having to log in to the ClearQuest database. See *Enabling E-Mail Submission Through the Rational E-Mail Reader* on page 247.

Enabling Automatic E-Mail Notification

ClearQuest can automatically send e-mail notification to users or user groups when a record meets the criteria you define. Enabling ClearQuest automatic e-mail notification requires:

- Creating E-Mail Rules—The ClearQuest administrator creates e-mail rules in the ClearQuest client.
- Enabling ClearQuest Clients to Send E-Mail Notification—ClearQuest client users configure ClearQuest to take advantage of e-mail rules.

Creating E-Mail Rules

As the ClearQuest administrator, it is your job to create the e-mail rules that define the criteria for automatic e-mail notification. You create e-mail rules by submitting Email_Rule records from the ClearQuest client. You can create a rule that sends e-mail notification when:

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

You must specify at least one of the already mentioned criteria for the e-mail rule to work. For example, you can create an e-mail rule that sends an e-mail message to the quality assurance team whenever a defect is resolved. The e-mail message can include any of the fields of the resolved defect.

E-Mail rules are implemented as a stateless record type in a schema. Most ClearQuest predefined schemas, except the Common and the Blank schemas, include the Email_Rule record type. If your schema does not include the Email_Rule record type, you can add it by installing the Email package.

Only users with Super User or Schema Designer privileges can create or modify an e-mail rule. For more information, see *ClearQuest User Privileges* on page 163.

Note: If you change the names of record types or actions in a schema that has existing e-mail rules, you must update the e-mail rules to reflect the new names.

To create an e-mail rule:

1 Log on to the ClearQuest client as a user with Schema Designer or Super User permission.

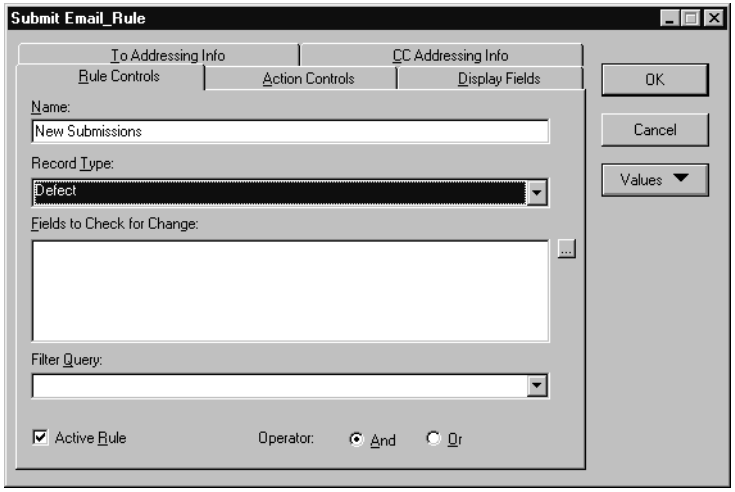
2 In ClearQuest client, click **Actions > New** and select the **Email_Rule** record type.

This displays the Submit Email_Rule dialog box. Use this dialog box to set up the parameters that determine when the e-mail message is sent, the users or user groups to receive the e-mail message, and the content of the e-mail message.

The following instructions provide an example of how to set up an e-mail rule to automatically send an e-mail message to specific users or user groups when a defect is submitted, and when a defect is assigned to a new owner.

Specifying Rule Controls for an E-Mail Rule

Use the **Rule Controls** tab of the Submit Email_Rule dialog box to specify the fields or queries on which to base the e-mail rule.



1 Specify a **Name** for the e-mail rule — for example, New Submissions.

2 Select the **Record Type** on which to base the e-mail rule — for example, Defect.

3 Select the **Fields to Check for Change**. When the selected fields change, an e-mail message is automatically sent. To send an e-mail message whenever a new defect is submitted, you do not need to select any fields. For other types of e-mail rules, you might select fields such as Headline, State, or Owner.

4 For **Filter Query**, you can select a public query to use as an additional trigger for sending an e-mail message. An e-mail message is only sent when a user modifies a

record that meets the query criteria. For example, if you select a query that finds all records associated with a beta release, an e-mail message is sent when a user modifies a record that included that query.

For this example e-mail rule, you do not need to select a **Filter Query**.

5 Make sure **Active Rule** is checked. To temporarily disable an e-mail rule clear **Active Rule**.

6 For **Operator**, you can select **And** or **Or**. The **Operator** works on the following fields on the **Rule Controls** tab:

- Fields to Check for Change
- Filter Query

The **Operator** also works on the following fields on the **Action Controls** tab (see *Specifying Action Controls for an E-Mail Rule* on page 238):

- Actions
- Action Types
- Source States
- Destination State

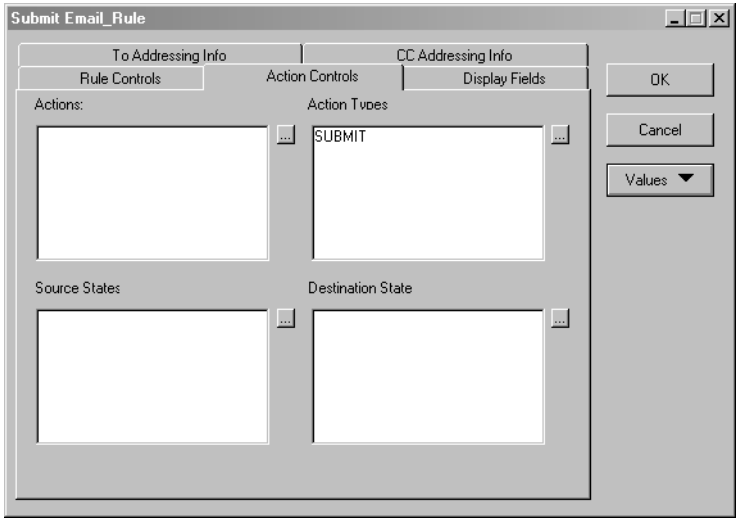
If you select **And**, an e-mail message is sent only if all of the conditions specified in the previously mentioned fields are met.

If you select **Or**, an e-mail message is sent if any of the conditions specified in the previously mentioned fields is met.

Specifying Action Controls for an E-Mail Rule

Use the **Action Controls** tab of the Submit Email_Rule dialog box to specify the actions, action types, or states on which to base the e-mail rule.

Note: Specifications made on the **Action Controls** tab are affected by the **And**/**Or** operator. See Step 6 on page 238.



1 For **Actions**, select the action that will be the basis of the e-mail rule. For example, select Assigned to have an e-mail message sent when an action is assigned.

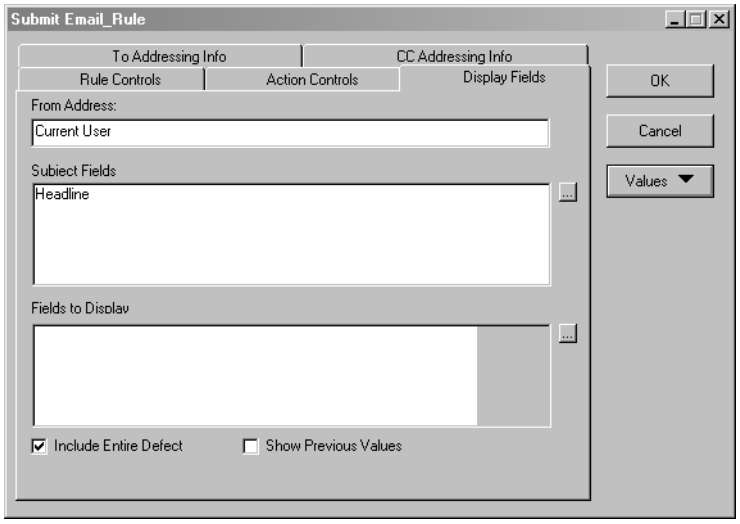
2 You can also base an e-mail rule on **Action Type** (one of the common actions predefined by ClearQuest Designer in the Actions grid.). For example, select **Submit** to have an e-mail message sent whenever a record is submitted.

3 You can also specify the **Source States**, and **Destination State** on which to base the e-mail rule.

For this example, leave **Source States**, and **Destination State** blank.

Specifying Display Fields for an E-Mail Rule

Use the **Display Fields** tab of the Submit Email_Rule dialog box to indicate the fields to include in the e-mail message.



1 For **From Address**, specify the return address for the e-mail message — for example, name@company.com.

   For this example, you can leave **From Address** as Current User.

   Note: The **From Address** is not applicable when using the MAPI protocol.

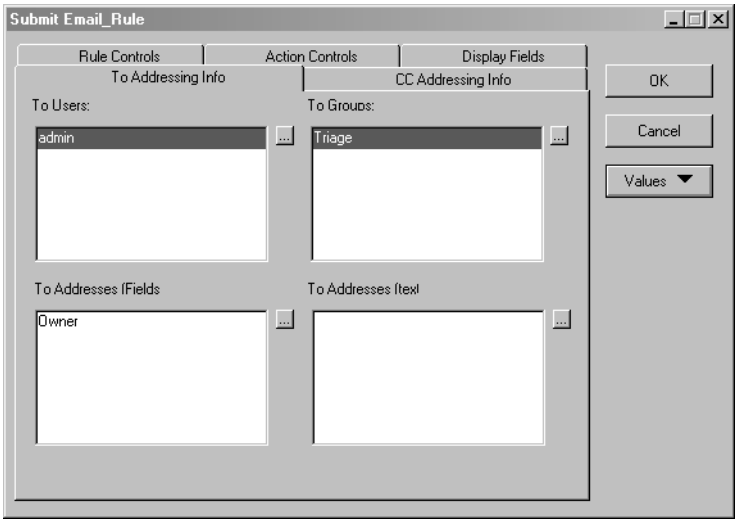
2 For **Subject Fields**, select the fields to display in the subject line of the e-mail message — for example, Headline.

3 For **Fields to Display**, select the fields to display in the body of the e-mail message.

   - To include the previous field values in the e-mail message, check **Show Previous Values**.
   - To include all of the fields in the e-mail message, check **Include Entire Defect**.

Specifying E-Mail Addresses for an E-Mail Rule

To specify the users and user groups to receive an e-mail message use the **To Addressing Info** tab of the Submit Email_Rule dialog box.



1 For **To Users**, you can select the users to receive the e-mail notification — for example, admin.

2 For **To Groups**, you can select a user group to receive e-mail — for example, the Triage group.

3 For **To Addresses (Fields)**, you can infer an e-mail address based on a field value.

For example, to send an e-mail message when a defect is assigned to a new owner: On the **Action Controls** tab, select the **Assign** action for **Actions**; then on the **To Addressing Info** tab, select the **Owner** field for **To Addresses (Fields)**.

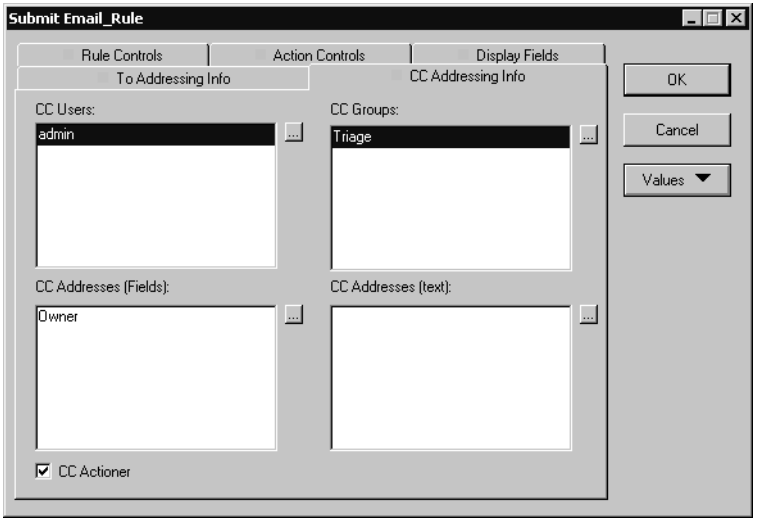
Note: Unless a specific e-mail address is entered in the To Addresses (Fields) box, all users must have an e-mail address defined as part of their user profile in order to receive e-mail messages. See *Adding a New User* on page 164.

4 For **To Addresses (Text)**, you can enter a specific e-mail address.

Specifying cc Addresses for an E-Mail Rule

Use the **CC Addressing Info** tab of the Submit Email_Rule dialog box to specify the users and user groups to receive copies of e-mail messages.

Note: To be able to receive copies of e-mail messages, all individual users, and users who are members of groups, must have their e-mail addresses defined as part of their user profile information. See *Adding a New User* on page 164.



1 For **CC Users**, select a specific user to receive a copy of the e-mail notification — for example, admin.

2 For **CC Groups**, you can select a user group to receive a copy of the e-mail message — for example, the Triage group.

3 For **CC Addresses (Fields)**, you can specify an e-mail address based on a field value.

Note: Unless a specific e-mail address is entered in the CC Addresses (Fields) box, all users must have an e-mail address defined as part of their user profile in order to receive e-mail. See *Adding a New User* on page 164.

4 For **CC Addresses (Text)**, you can enter a specific e-mail address.

5 Select **CC Actioner** to send a copy of the e-mail message to yourself.

6 Click **OK** to apply the e-mail rule.

Sample E-Mail Rules

If you installed the evaluation user database of ClearQuest, also referred to as the Sample Database, you can access examples of several e-mail rules. These examples are stored in the Public Queries folder of the Workspace and include:

- New Submissions
- Modifications
- Resolutions
- Closures

Finding Existing E-Mail Rules

You can create a query to find all of the existing e-mail rules on an existing ClearQuest system and to create an archive listing of your e-mail rules.

To create a query that finds existing e-mail rules, do the following in the ClearQuest client:

1. Create a new folder in Public Queries called Email Rules by right-clicking the Public Queries folder, selecting **New Folder** from the list, and name the new folder, **Email Rules**.
2. Click **Query > New Query**.
3. Select **Email_Rule** as the record type and click **OK**.
4. In the Choose Query dialog box, click **Next** without specifying a query.
5. In the Define how the query displays dialog box, select a few fields such as **Name**, **record_type**, and **Is_Active_Rule**. When the query is run, these fields are displayed to help you distinguish between the existing e-mail rules. Click **Next**.
6. In the Select fields to use as query filters dialog box, click **Next** without filtering the query.
7. In the Define query filters dialog box, click **Run**. A list of existing e-mail rules will be displayed.
8. Select **File > Save** and name the new query **All Email Rules**. Click **OK**.
9. The name of the new query will appear in your **Personal Queries** folder. Drag the new query into the **Email Rules** folder. When you want to find all of the e-mail rules in your system, double-click the **All Email Rules** query in your **Email Rules** folder.

Modifying an Existing E-Mail Rule

To modify an existing e-mail rule:

1. If you have not done so already, create a query that finds all existing e-mail rules. For more information, see *Finding Existing E-Mail Rules* on page 243.
2. In the Email Rules folder you created, double-click the query that lists the existing e-mail rules
3. In the existing e-mail rules table, select the rule that you want to modify.
4. Click **Actions > Modify**.
5. Use the E-mail Rules form to change the Rule Controls, Action Controls, Display Fields, To Addressing Info and CC Addressing Info selections.

6 Click **OK**.

Administering E-Mail Addresses

It is important to keep user e-mail addresses up-to-date. For example, an e-mail rule may not produce the results you want if the To Addressing Info or CC Addressing Info section lists a group in which the e-mail address of one or more users is missing or invalid.

For more information, see *Working with User Groups* on page 172.

Hidden Users

The Rational ClearQuest E-Mail Reader will not send e-mail messages to hidden users whose ClearQuest user name is used in an e-mail rule.

When this situation arises, use an explicit e-mail address, such as name@company.com, instead of the ClearQuest user name.

Enabling ClearQuest Clients to Send E-Mail

To take advantage of automatic e-mail notification, each ClearQuest client on Windows and UNIX must enable e-mail notification. To learn how to enable e-mail notification, refer users to the instructions in the Help for the ClearQuest client. Keep in mind that you need to provide users with the e-mail provider parameters, such as the SMTP host address of the mail server or the MAPI profile name.

Note: To use MAPI notification for Windows platforms, you must have Collaboration Data Objects installed. See the *Installation Guide* for Rational Server Products for more information.

Enabling E-Mail Notification for ClearQuest Web

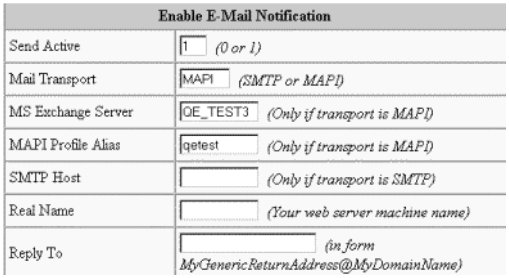
To enable e-mail notification in ClearQuest Web, you must have Super User or Schema Designer privileges.

After installing ClearQuest Web (either for an initial or an upgrade install), you can set up your ClearQuest Web clients to have e-mail notification according to the business rules of your schema.

1 Log on to ClearQuest Web client with Super User or Schema Designer privileges.

2 Click **Operations > Edit Web Settings**.

3 Scroll to the **Enable E-Mail Notification** section.



4 Enter your e-mail settings.

Setting	Description
Send Active	Enter 1 to enable e-mail notification. Enter 0 to disable it.
Mail Transport	Specify SMTP or MAPI depending on your mail transport type.
MAPI Server	If you are using MAPI, enter the mail server name.
MAPI Profile	If you are using MAPI, enter the profile alias name.
SMTP Host	If you are using SMTP, enter the host name.
Real Names	Enter your Web server machine name. If you enter a Web server name, e-mail messages display the server name in the **From** field. If you do not want the server name to display in all e-mail messages, do not enter anything.
Reply To	Enter a generic, reply-to address for this Web server. This address automatically appears in the Reply To field of each e-mail message. If you leave it blank, no address appears.

5 At the top of the Web page click **OK** to save your changes.

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.

Note: To use round-trip e-mail, you must specify the same e-mail address when you configure the Rational E-mail Reader that you specify for the **From Address** on the **Display Fields** tab of the Submit E-mail Rule dialog box. See *Creating E-Mail Rules* on page 236, and *Configuring the Rational E-Mail Reader* on page 248.

The following is an example of how you can use round-trip e-mail:

1. A user submits a record by using e-mail, using the required format. (See *Formatting E-Mail for Submission* on page 254.) 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 *Creating E-Mail Rules* on page 236.)
3. The lead engineer receives the e-mail message entitled "PROD0000137," then responds by e-mail, including remarks in the description field.
4. The E-mail Reader processes the e-mail message and submits the modified record to the ClearQuest database.

Using an Action Hook to Send E-Mail

You can use an action notification hook to have ClearQuest automatically send an e-mail message when a user completes a specification. In your hook code, you use the ClearQuest API to create the message and send it.

ClearQuest 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.

Note: If you change the names of record types or actions in the schema, you must update your action hook to reflect the new names.

To create an action notification hook:

1. Log on to ClearQuest Designer as a user with Schema Designer or Super User privilege.
2. In ClearQuest Designer, expand **Record Types** or **Record Types – Stateless** until you see the **Actions** folder.
3. Double-click **Actions** to display the **Actions** grid.
4. In the **Actions** grid, click the **Notification** column of the action you want to add the notification hook to, then click the arrow icon and click **SCRIPTS > BASIC** or **SCRIPTS > PERL**.

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 using the correct editor before adding or editing your code.

5 Add your hook code, then click **Hooks > Compile** to compile your code and verify that there are no syntax errors.

6 Close the script editor window.

For more information, see *Working with Action Hooks* on page 204 and the *Action Notification Hook Example* on page 334.

Note: You will need to stop and restart the Rational ClearQuest Mail Service every time you make changes either to schema or an e-mail notification hook.

Enabling E-Mail Submission Through the Rational E-Mail Reader

The Rational E-Mail Reader is a tool that takes specially formatted e-mail messages, converts the information in them into a form understood by ClearQuest, and submits the information to a ClearQuest database. This allows users to submit and modify records by e-mail, without using a ClearQuest client.

In the current release, the Rational E-Mail Reader uses a Windows Service called the Rational ClearQuest Mail Service to perform the functions already described. You can start, stop, pause, and resume the execution of the Rational ClearQuest Mail Service, configure startup and recovery options, and enable or disable its use.

The Rational E-Mail Reader is used for two Rational products, ClearQuest and RequisitePro. During the configuration process, you must specify which product is being implemented.

This section describes:

- *Creating Mailboxes for the Rational E-Mail Reader* on page 248.
- *Configuring the Rational E-Mail Reader* on page 248.
- *Configuring Additional Options for the Rational E-Mail Reader* on page 253.
- *Formatting E-Mail for Submission* on page 254.

Creating Mailboxes for the Rational E-Mail Reader

The Rational E-Mail Reader requires a dedicated e-mail account for each ClearQuest database and RequisitePro project. The account provides an e-mail mailbox to which change request submissions and modifications can be sent. The Rational ClearQuest Mail Service takes the messages sent to this mailbox, processes them, and submits the request to the ClearQuest database or to the RequisitePro project.

Before configuring the Rational E-Mail Reader, you and the e-mail administrator for your organization must:

- Determine if your e-mail system uses the SMTP or MAPI protocol.
- Create a dedicated e-mail account for each ClearQuest database and RequisitePro project.
- For each e-mail account, identify the server names or IP addresses of the SMTP and POP3 servers (if you are using the SMTP protocol) or MAPI mail servers (if you are using the MAPI protocol).
- For each e-mail account, identify the login and password of the POP3 server (if you are using the SMTP protocol). If you are using the MAPI protocol, the mailbox login must be configured to use the same operating system authentication on the mail server machine.
- For each e-mail account using MAPI, the mailbox login and the Rational ClearQuest Mail Service must be configured to use the same operating system authentication.

Configuring the Rational E-Mail Reader

The process for configuring the Rational E-Mail Reader differs slightly depending on whether you want to use the SMTP protocol or the MAPI protocol for e-mail transmissions. If you are using the SMTP protocol, omit steps 8 and 9. If you are using the MAPI protocol, omit steps 5 to 7.

To configure the Rational E-Mail Reader for the SMTP protocol, do the following:

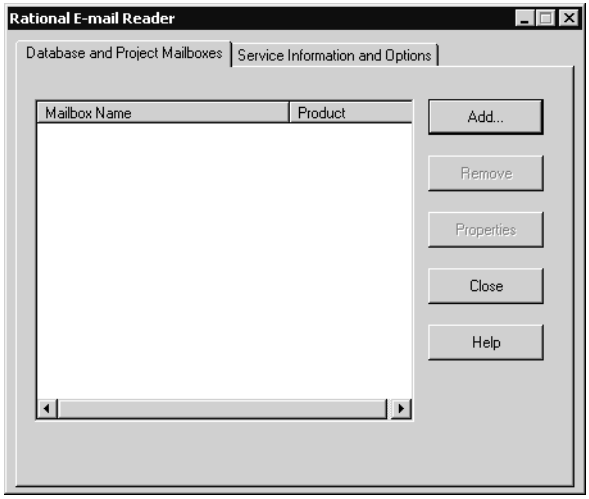
1. From the Start menu, click **Run**.
2. In the Run dialog box, enter the path to the E-Mail Reader. For example, if C:\Program Files\Rational\ClearQuest is the default directory to store all ClearQuest files during installation, enter:

   C:\Program Files\Rational\ClearQuest\mailreader.exe

   Click **OK**.

3 The Rational E-Mail Reader dialog box appears. The dialog box consists of two tabs:

- Database and Project Mailboxes



  This tab lists the ClearQuest databases and RequisitePro Projects that have e-mail accounts. Use this tab to add the e-mail accounts you created for new ClearQuest databases and RequisitePro projects; see *Creating Mailboxes for the Rational E-Mail Reader* on page 248.

- Service Information and Options

  This tab includes additional options and status information for the Rational E-Mail Reader.

4 In the Database and Project Mailboxes tab, click **Add** to add the new e-mail account for the new ClearQuest database or RequisitePro project.

- If you are using the SMTP protocol, follow Step 5 to Step 7 then skip to Step 10.
- If you are using the MAPI protocol, skip to Step 8 now.

5 In the Choose E-Mail Handler dialog box select **SMTP** as the **E-Mail Protocol**. Click **Next**.

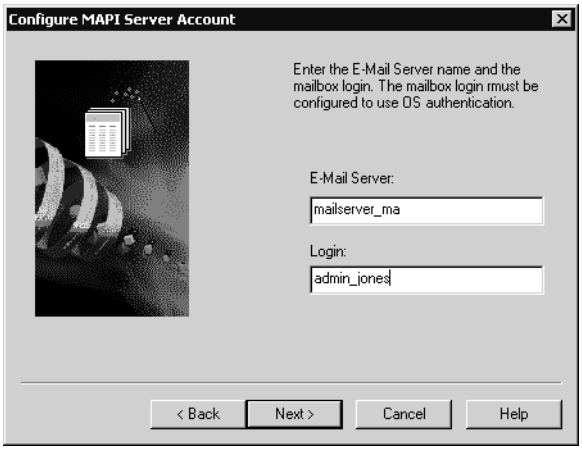
6 In the Select Mail Server dialog box, specify the **SMTP Server** name for outgoing mail and the **POP3 Server** name for incoming mail. The server names should be the same. Click **Next**.

7 The Configure Mail Server Account dialog appears. Enter the information of the dedicated e-mail account for the ClearQuest database, including the **POP3 E-Mail Address**, **Login** and **Password**. This e-mail account must be created exclusively for use by the Rational E-Mail Reader, and there should be no other users using this account. Click **Next**.

- If you are using the SMTP protocol, skip to Step 10 and continue through Step 13.
- If you are using the MAPI protocol, continue with Step 8 to Step 13.

8 In the Choose E-Mail Handler dialog box, select **MAPI**. Click **Next**.

9 In the Configure MAPI Server Account dialog box, enter values for the **E-Mail Server** and the **Mailbox Login**. Click **Next**.

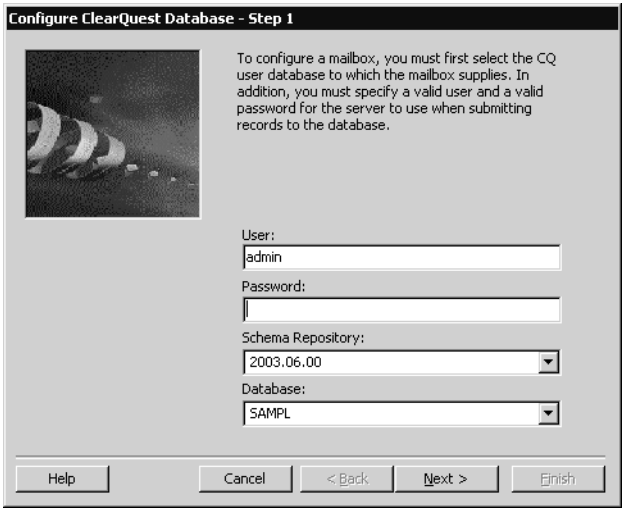


Note: The mailbox login must be configured to use the same operating system authentication on the server machine.

10 The Configure Rational E-Mail Handler dialog box appears.

The Rational E-Mail Handler supports both ClearQuest and Requisite Pro. However, for a single e-mail account, the Reader can be set up only to read e-mail for either ClearQuest or RequisitePro. In the Choose Rational E-Mail Handler dialog box, select **ClearQuest** and click **Configure**.

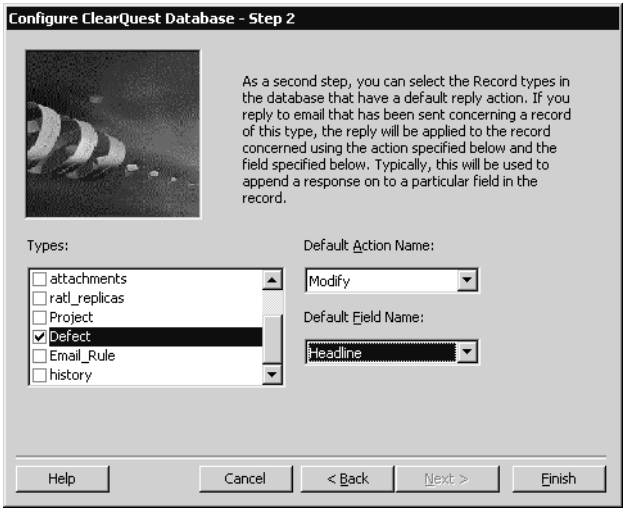
11 In the Configure ClearQuest Database - Step 1 dialog box, enter a valid **username** and **password** that the Rational E-Mail Reader can use to login to the ClearQuest system. Inbound e-mail activity will be traced to this user account.



Then select the **schema repository** and **user database** names. Click **Next**.

The Rational E-Mail Handler verifies the user account with the ClearQuest database. The Configure ClearQuest Database Step 2 dialog box appears.

12 The Configure ClearQuest Database - Step 2 dialog box is used to set default actions for sending and replying to an email message.



An e-mail message sent with the Rational E-Mail Reader may specify an action to be taken in response, and a field in the record where the response will be appended in the message. However, if no action or field are specified in the email message, then the defaults you supply here are used automatically.

For example, you might configure the Rational E-Mail Reader to use the SAMPL user database, then select Defect as the record type, Modify as the default action, and Description as the default field. Then when an email is sent about a Defect record in the SAMPL database, and that email does not specify an action or field for the response, a message will be generated automatically using the Modify action, and that message will be placed in the Description field.

Note: When you format the e-mail message, multiline text fields such as the **Description** field must be surrounded by { }.

Select the **Record Types**, **Default Actions**, and **Default Field Names** and click **Finish**.

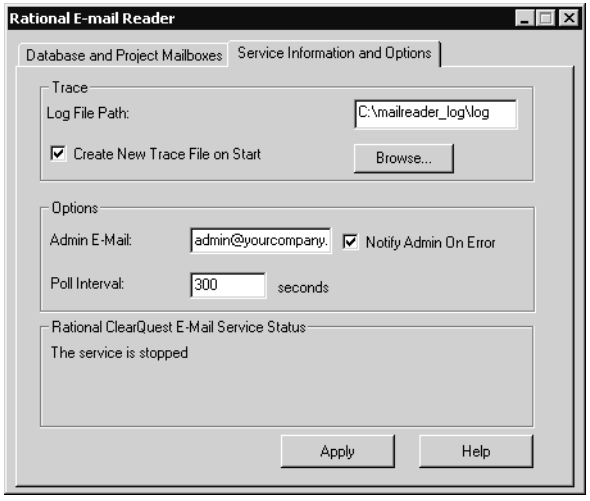
13 The Configure Rational E-Mail Handler dialog box displays the name of the connection and user database in the Configure box. Click **Finish**.

The ClearQuest e-mail account is listed in the **Databases and Projects** tab of the Rational E-Mail Reader.

Configuring Additional Options for the Rational E-Mail Reader

To configure additional options for the Rational E-Mail Reader, click the **Service Information and Options** tab. You can do the following:

- Create a log file to trace each e-mail.
- Specify the poll interval of the Reader. The poll interval determines how often the Reader checks for e-mail messages. The default value is 300 seconds.
- Provide the e-mail address of the ClearQuest administrator and ask the Rational E-Mail Reader to notify the administrator when an error occurs.



Note: Any changes you make on this tab will only take effect after you have stopped and restarted the Rational ClearQuest Mail Service.

To configure these options, do the following:

1. Enter the path to where you want the log file to be created and stored in the **Log File Path** box.
2. Click **Create New Trace File on Start** if you want to delete the current log file and create a new log file each time the service is restarted.
3. Enter an e-mail address and click **Notify Admin On Error** to receive messages from the Rational E-Mail Reader when an error occurs. Also, specify another time in seconds if the default value of 300 seconds is not satisfactory.
4. Click **Apply**.

- Select the **Database and Project Mailboxes** tab and click **Close** to exit.

Configuring the Rational ClearQuest Mail Service Properties

Before starting the service, you need to enter additional logon information in the Rational ClearQuest Mail Service Properties dialog box.

To enter logon information for the Rational ClearQuest Mail Service:

1 Click **Start > Settings > Control Panel > Administrative Tools > Services > Rational ClearQuest Mail Service**. Right-click and select **Properties**.

2 The Rational ClearQuest Mail Service Properties dialog box appears. Click the **Log On** tab.

3 The default user account setting is **Local System account**. If the Rational ClearQuest Mail Service needs to access network resources, then you need to specify a user account for the service that has privileges to these network resources.

   If that is the case, change the default and click **This account**. Specify the user account and then enter the **Password** and confirm the **Password**.

4 Click the **General** tab and verify the **Startup type**.

   In most cases you want the Startup type to be **Automatic**, so that the service will restart automatically every time the system is restarted.

5 Click **OK** and close the Services window.

6 Restart the computer where you are setting the Rational ClearQuest Mail Service.

   This will update all your changes into the system. The Rational ClearQuest Mail Service will automatically restart. Then you can use the service to stop, disable or restart the e-mail processing.

Formatting E-Mail for Submission

When ClearQuest users submit or modify records by e-mail, all of your schema rules apply, including required fields and legal values. For example, if a user specifies the Submit action, but fails to include all required fields for the Submit state, the E-mail Reader returns the e-mail message to the sender with an error message.

When submitting or modifying ClearQuest records by using e-mail, you must use the following format. If the e-mail message does not conform to this format, the e-mail submission fails and the e-mail message is deleted.

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

Body: [<fieldname>:]< legal value>

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

[<fieldname>:]< legal value>

{[<multiline_fieldname>:] This is an example of a field value for a field whose data type is a multiline text field. It requires curly braces.
}

Note: The right curly brace (}) must be on a separate line from the multiline text it encloses.

Guidelines for Formatting E-Mail

Use the following guidelines when submitting or modifying ClearQuest records by e-mail:

- The <record type> is 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* on page 256). Users can override the defaults by specifying a different action.

Note: If you do not specify an <action>, and your schema does not have default actions, the e-mail submission will be rejected.

- Specify 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...
  }

- Enter legal values for each field you are modifying. 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.
- Enter values for all 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 enter values for read-only fields such as the defect ID field. ClearQuest does not change such fields.
- ClearQuest E-mail submission does not support attachments.

- Regarding (RE:) is not required at the beginning of the subject line.
- ClearQuest e-mail notification mail uses the required format automatically. You can use this mail as a template for sending additional records or modifications by e-mail.
- You can configure the E-mail Reader to keep a log and to send e-mail messages 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 Notes_Log. The Subject line requires the record type and record ID, but because of the defaults, the action name is optional.

Subject: RE: defect SAMPL00000017

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

Format Example for Submitting a New Record

This example e-mail message submits a defect record and fills in the specified fields.

Subject: defect submit

Body: Headline: inventory report is not running correctly
Severity: 1-Critical
Project: ClassicProject
Priority: 2-Give High Attention

{Description: When running an inventory report, application crashes if more than 50 items are included in the report.
}

Format example for changing the state of a record

Subject: defect closed SAMPL00001234

Body: {Note_Entry: Followed the steps outlined in the description, and confirmed that the error no longer occurs.

}

Test Method: Manual

Format Example for Modifying a Record Using the Default Action

In this example, the ClearQuest administrator has designed the schema so that the Defect record type defines the default action as MODIFY, and the default field as the Description field. These defaults are used when no action is specified in the subject line. This e-mail message uses the MODIFY action to update the SAMPL00000017 defect and appends the value in the Description field.

Subject:	defect SAMPL00000017
Body:	{Have tried to reproduce the problem with the inventory reports not allowing more than 50 items and it works just fine. Please retry and provide more information. }

Importing and Exporting Data

13

This material describes how to export data from one Rational ClearQuest database and import it into another ClearQuest database, and how to import data into ClearQuest from another defect tracking system.

The topics covered include:

- *Overview of Importing and Exporting Data* on page 259.
- *Creating an Import Schema* on page 260.
- *Creating a Database for Imported Data* on page 262.
- *Creating a ClearQuest Import File* on page 262.
- *Exporting Data from ClearQuest* on page 267.
- *Importing Data into ClearQuest* on page 271.

See also

- *Copying a Schema with cqload* on page 376.
- *Reusing Record Forms* on page 159.

Overview of Importing and Exporting Data

The Import and Export tools allow you to move data such as records, history, and attachments from other change request systems and between ClearQuest databases.

Starting the Import Process

The procedure for exporting and importing data is:

- Create a new import schema, or modify an existing schema to support the data you want to import. See *Creating an Import Schema* on page 260.
- Upgrade an existing user database with the modified import schema, or create a new user database and associate it with the import schema. See *Creating a Database for Imported Data* on page 262.
- Export your data to text files that use the ClearQuest import-file format.

- If you are exporting data from a system outside of ClearQuest, see *Creating a ClearQuest Import File* on page 262.
- If you are exporting data from an existing ClearQuest database, see *Exporting Data from ClearQuest* on page 267. You can use the ClearQuest Export Tool to create an import file.

- Use the ClearQuest Import Tool to import the data from the import files into your user database. See *Importing Data into ClearQuest* on page 271.

Testing the Import Process

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

- Create a test database.
- 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 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. When creating an import schema remember that:

- To prevent data corruption, the schema repository of the import schema must have the same data code page value as the database set that exports the data. For more information, see Chapter 2, *ClearQuest and Code Pages*.
- The import schema must contain the appropriate record types, fields, field 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 in your import file must match those 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.

- You must 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.
- If you are updating existing records, the import schema must have an action of the Modify type.
- If you are importing duplicate records, the import schema must have an action of the Duplicate type. It must also include a Duplicate state and the original state that the record was in before the state changed to Duplicate.
- If you are importing duplicate records, your record form must contain 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 IDs of the duplicate records. For an example, see the Duplicates tab of the Defect record type in any ClearQuest predefined schema.
- The import schema must contain a record form that users can use to view and modify the imported data.
- If you are importing history, attachments, or duplicate records, or if you are upgrading existing records, the import schema must contain a field to store the old ID values. See *Including the Original Record ID* on page 261.
- For records that contain reference lists, the import 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 records before importing the defect records and then populate the project record type with project names.

Note: ClearQuest validates 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. See *Data Types Supported in ClearQuest* on page 265.

For more information, see Chapter 4, *Working with ClearQuest Schemas* and Chapter 5, *Customizing a Schema* in the *Administrator's Guide* for Rational ClearQuest.

Including the Original Record ID

ClearQuest assigns a new record ID to each imported record, so be sure that your import schema contains a field for the original record ID. ClearQuest uses the original record ID to maintain data integrity when importing duplicate records, history, and attachments, and when updating existing records. The original record ID is also used

when re-creating parent-child links. It is also useful for finding records based on the original record ID.

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

Creating a Database for Imported Data

Before importing data into ClearQuest, you must create a user database to hold the imported data and associate this database with the import schema. For more information, see *Creating User Databases* on page 52.

You can use an existing user database that is already associated with the schema you are using as 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 *Overview of Customizing a Schema* on page 93.

Creating a ClearQuest Import File

If you are exporting data from an existing ClearQuest database, you can use the ClearQuest Export Tool to create the import file.

To import data into ClearQuest from another defect-tracking system, you must first 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. If a record type includes history and attachments, you must create at least three import files, one for records, one for history, and one for attachments.

This section describes the format requirements for record type, history, and attachments import files.

- Formatting Record Type Import Files
- Formatting History Import Files
- Formatting Attachment 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. See *Testing the Import Process* on page 260.

Formatting Record Type 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, semicolons, 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 log in 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 data types of 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 and user groups. To learn more about creating users and users groups, see *Working with Users* on page 164.
- 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."
  ```

A carriage return can be used to import values into a multiline field.

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. The following table lists the data types that ClearQuest supports.

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
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

Formatting History Import Files

Each history import file should contain history information for only one record type. If you are importing multiple record types, you must create a history import file for each record type that contains 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 history 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 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 2003"
- "April 6, 2003 8:30:00"
- "8:30:00 Apr 6 2003"
- "4/6/2003 8:30:00PM"

Formatting Attachment Import Files

Each attachment import file should contain attachment information for only one record type. If you are importing multiple record types, you must create an attachment import file for each record type containing attachments.

Keep in mind the following when formatting an attachment import file:

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

Note: If you use the ClearQuest Export Tool to export attachment 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.

Exporting Data from ClearQuest

Note: To export data from a defect tracking system other than ClearQuest, you must use your own database vendor tool. See *Creating a ClearQuest Import File* on page 262.

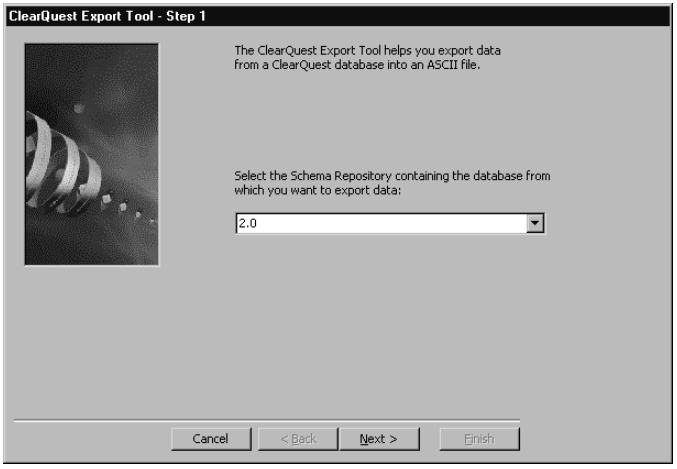
To export data from an existing ClearQuest database, use the ClearQuest Export Tool. The ClearQuest Export Tool exports data from a ClearQuest database into an ASCII format optimized for the ClearQuest Import Tool.

Note: To prevent data corruption, you must only export and import data between databases that have the same data code page value. Import and export server machines must also have the same operating system code page. For more information, see Chapter 2, *ClearQuest and Code Pages*.

To run the ClearQuest Export Tool:

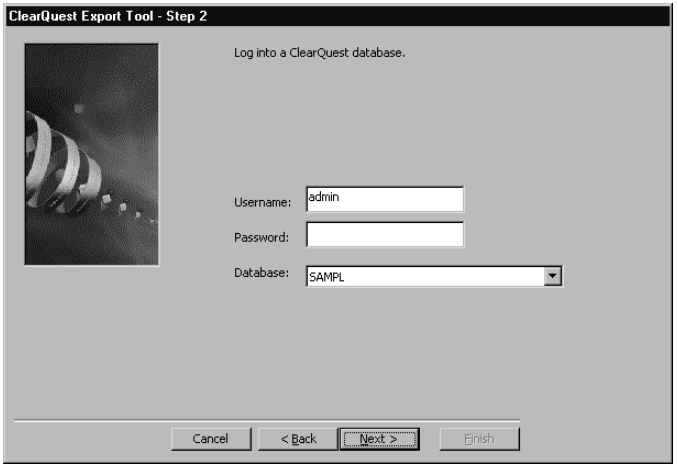
1 Click **Start > Programs > Rational Software > Rational ClearQuest > Rational ClearQuest Export Tool**.

2 In the ClearQuest 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 Click **Next**.

4 Enter your user name and password, select the database you want to export and click **Next**.

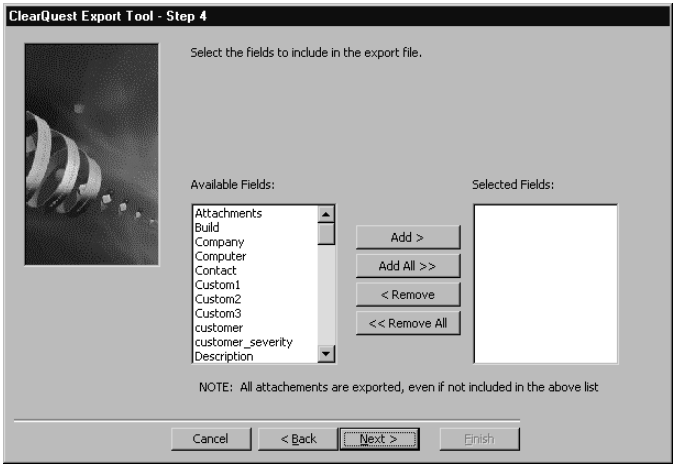


5 Select a record type to export.

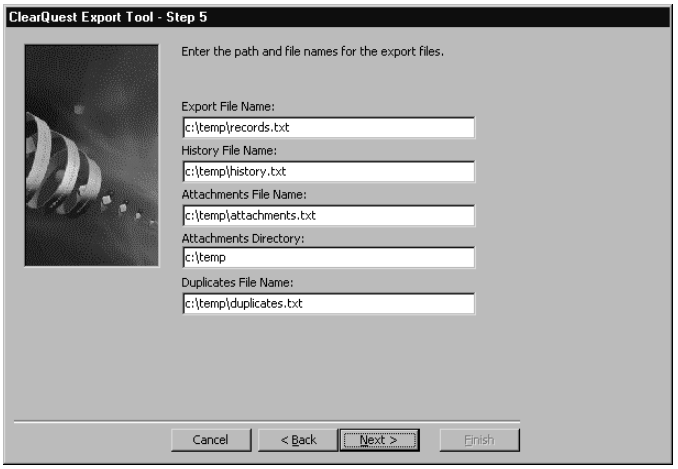
The Export Tool automatically exports history, attachments, and duplicates for the selected record type.

6 Click **Export all records** or click **Export records based on an existing query** and select a query and click **Next**.

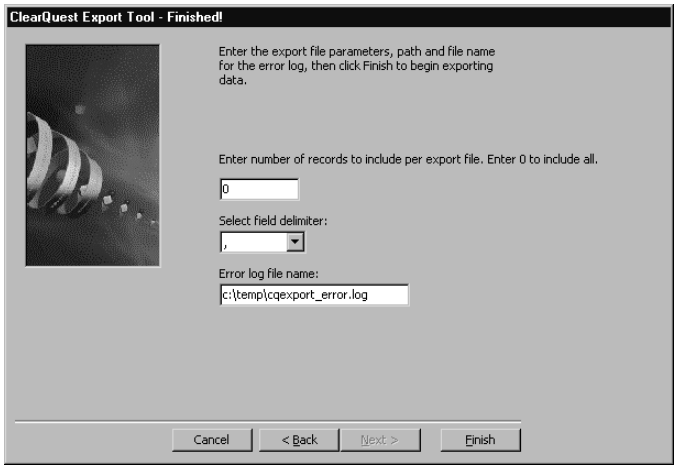
7 Select the fields to export.



8 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.



9 Enter the final export parameters.



- Specify **0** to include all records in one file or enter the number of records you want for each file.
- Enter the path and name for the Error log.

10 Click **Finish** to begin the export.

ClearQuest displays a dialog box when the export is complete and lists any detected errors.

Note: When you export data from ClearQuest using the Export Tool, the records.txt files have an ID field but the values in the Old_id field are blank. When you import history files, you must designate the field that has the old ID. In order for the import

process to work, you must duplicate the values in the ID field to the Old_id field. This will allow you to import history files.

Importing Data into ClearQuest

Before importing data into ClearQuest, you must have completed:

- *Creating an Import Schema* on page 260.
- *Creating a Database for Imported Data* on page 262.
- *Creating a ClearQuest Import File* on page 262 or *Exporting Data from ClearQuest* on page 267.

Determining the Order of Record Importing

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.

The ClearQuest Import Tool allows you to import one record type at a time, including the history and attachment information for that record type. 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.

If you have duplicate records, you must import them separately. See *Importing Duplicate Records* on page 275.

Using the ClearQuest Import Tool

Note: To prevent data corruption, you must only export and import data between databases that have the same data code page value. Import and export server machines must also have the same operating system code page. For more information, see Chapter 2, *ClearQuest and Code Pages*.

Use the ClearQuest Import Tool to import records from an import file into ClearQuest.

1. Click **Start > Programs > Rational Software > Rational ClearQuest > Rational ClearQuest Import Tool**.
2. Log in with your user name and password, select the schema repository to import into and click **OK**.
3. In the ClearQuest Import Tool - Step 1 of 5 dialog box:
   a. In Step 1, select the name of the record type to which you will be importing records.

b In Step 2, select the type of data you are 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 In Step 3, if applicable, indicate whether or not you are updating existing records. If this is your first import, this step is not necessary.

You must select **Yes**, if you are importing history information, attachments, or duplicate information.

d Click **Next**.

4 In the ClearQuest Import Tool - Step 2 of 5 dialog box:

a For **Record Data**, enter the location and name of the import file that contains the data to be imported.

Enter a location and name for the discarded data log. This must be a new file; you cannot use an existing file.

If a record does not convert successfully, the Import Tool saves it to the discarded data file you specify. You can use this file to fix problems and to reimport the record. See *Recovering from Import Errors* on page 275.

Select the field delimiter used in the import file.

b If you are importing history information, enter the **History Data**:

Enter the location and name of the import file that contains the data to be imported.

Enter a location and name for the discarded data log. This must be a new file; you cannot use an existing file.

If the history data does not convert successfully, the Import Tool saves it to the discarded data file you specify. You can use this file to fix problems and to reimport the record. See *Recovering from Import Errors* on page 275.

Select the field delimiter used in the import file.

c If you are importing attachment information, enter the **Attachment Data**:

Enter the location and name of the import file that contains the data to be imported.

Enter a location and name for the discarded data log. This must be a new file; you cannot use an existing file.

If the attachment data does not convert successfully, the Import Tool saves it to the discarded data file you specify. You can use this file to fix problems and to reimport the record. See *Recovering from Import Errors* on page 275.

Select the field delimiter used in the import file.

d Click **Next**.

5 In the ClearQuest Import Tool - Step 3 of 5 dialog box:

a For 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 For importing duplicates, mark whether the import file contains information about duplicate records.

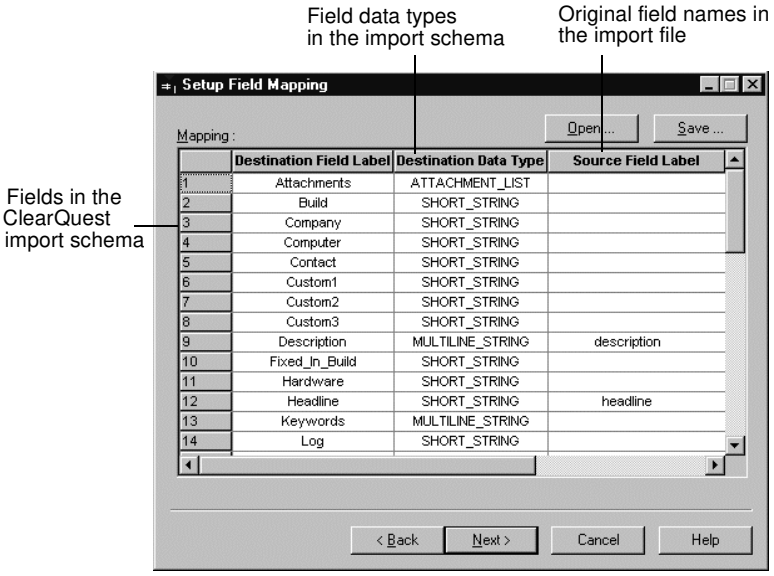
If the import file contains information about duplicates, select the exported field name that contains the duplicate information. For more information on importing duplicates, see *Importing Duplicate Records* on page 275.

c For importing record updates, duplicates, history, or attachments, select the field in the import schema that will contain the original record ID (or other unique identifier) of the exported data.

When importing history, attachments, and duplicate records, ClearQuest must track the past unique identifier (typically the old ID number) of the imported record. ClearQuest must know the record to which the attachment, history information or duplicate record belongs.

d Click **Next**.

6 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.
- Be sure to map the import file's original ID to the Destination field that will store it.

Note: If you used the ClearQuest Export Tool to create the import file, the original ID is stored in the "ID Column" of the exported file. You should map this "ID" field to the original ID field.

To save the mapping:

a Click **Save**.

b Browse to a location where you want to save the mapping profile and type a name, using a *.txt extension, then click **Save**.

To reuse an existing mapping:

a Click **Load**.

b Select an existing mapping profile and click **OK**.

7 In the ClearQuest Import Tool - Step 5 of 5 dialog box,

a Review the import summary to verify that the settings 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 at the same you import 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. Import the parent record before importing child records.
- Within ClearQuest, duplicate records are handled using a Duplicate state and action. Be sure your import schema includes both.
- The import schema should also include the original state that the record was in before it changed to the Duplicate state.

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.

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.

When importing records separately, in the ClearQuest Import Tool - Step 1 of 5 dialog box, select **Yes** for updating existing records.

Recovering from Import Errors

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 the previously mentioned Step 3.
- errlog.txt: An error file saved to your TEMP directory containing detailed information about all failures. 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. The name of this file can be:
  - The discarded data log file name with ".log" appended.
  - errlog.txt
- A file saved into your TEMP directory that contains a copy of everything printed to the ClearQuest Import Tool status log. It is a text file named using the following convention:

```
import_status_<year><month><day>_<hour><minute>_<second>.txt
```

For example, import_status_20021012_1355_01.txt indicates a status file created on October 12, 2002 at 1:55:01 p.m.

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 Record Type Import Files* on page 263.
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 *Importing Data into ClearQuest* on page 271. In Step 3, select **Update existing records**. All other steps stay the same.

ClearQuest Schemas and Packages

A

Rational ClearQuest includes several predefined schemas. There is a schema for each Rational Suite product for Windows, and a *Blank* schema, containing only required record types, which you can use to build a schema.

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

The topics covered include:

- *ClearQuest Predefined Schemas* on page 278.
- *ClearQuest Packages* on page 279.
- *State Model for the Defect Record Type* on page 288.
- *State Model for the EnhancementRequest Record Type* on page 289.
- *State-Type Models for Packages* on page 291.

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 Rational ClearQuest for a list of upgrade packages that you may need to apply.

For instructions on how to add packages to a schema, see Chapter 6, *Applying Packages*. For information on how to customize a schema, see Chapter 5, *Customizing a Schema*.

Warning: Be sure to plan carefully before adding packages. After 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.

ClearQuest Predefined Schemas

The following table lists the predefined schemas in ClearQuest.

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
Development Studio	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

Schema	Description	Packages included
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

This section describes the latest version of each ClearQuest package. Earlier package versions may create different record types and fields. Some packages are read-only; their functionality cannot be changed.

Package	Description	Added or modified	Fields
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: ▪ 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: ▪ am_statetype

Package	Description	Added or modified	Fields
AMWorkActivity Schedule	Provides scheduling attributes needed to integrate Rational ClearQuest and Microsoft Project 2000 using Rational ClearQuest Project Tracker. With the AMWorkActivity Schedule 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 are added to this record type family. Adds the **Schedule** tab to the enabled record type.	Fields added to the enabled AMSchedule record type: ▪ 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)	Lets you add and remove attachments related to a record.	Adds an Attachments tab to the enabled record type.	Fields added to enabled record type: ▪ Attachments
BaseCMActivity	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. For more information, see *Managing Software Projects* with Rational ClearCase.	Adds the BaseCMActivity record type.	Fields included in BaseCMActivity record type: ▪ Owner ▪ Description ▪ Headline

Package	Description	Added or modified	Fields
CharacterSet Validation	Prevents data corruption by validating that all data entered into a record from the native client is from the same code page as the ClearQuest data code page of the schema repository. You can apply the CharacterSetValidation package with the Package Wizard and with the **apply_character_set_validation_package**. For more information, see *About the CharacterSetValidation Package* on page 38.	Adds two Perl hooks for all non-stateless record types: ▪ The access control hook CheckCodePageMismatch ▪ The base validation hook CharacterSetValidation Adds a new stateless record type called _ratl_data_code_page	
ClearCase (read-only)	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.	Adds the cc_change_set and cc_vob_object record types. Adds the ClearCase tab to the enabled record type.	Fields included in cc_change_set ▪ record type: ▪ objects Fields included in cc_vob_object ▪ record type: ▪ name ▪ object_oid ▪ vob_family_uuid Fields added to enabled record type: ▪ cc_change_set

Package	Description	Added or modified	Fields
ContentStudio	Provides support for integration with Rational Suite ContentStudio.	Adds the ContentChangeRequest record type.	Fields included in ContentChangeRequest record type: ▪ Description ▪ Headline ▪ Note_Entry ▪ Notes_Log ▪ Owner ▪ vgnAssignee ▪ vgnCMS ▪ vgnDueDate ▪ vgnManagementID ▪ vgnProjectPath ▪ vgnTaskName
Customer	Supports the integration of customer data with your defect and change tracking system.	Adds a Customer stateless record type. Also adds reference fields for customer information to the record type you select.	Fields included in the Customer record type: ▪ Attachment ▪ CallTrackingID ▪ Company ▪ Description ▪ Email ▪ Fax ▪ Name ▪ Phone Fields added to enabled record type: ▪ Customer ▪ Customer_Severity

Package	Description	Added or modified	Fields
Email (read-only)	Supports automatic e-mail notification when records are modified.	Creates an Email_Rule stateless record type. Adds a base action to the enabled record type called Send_Email_Notif. This base action runs the e-mail rule whenever any action is invoked.	Fields included in Email_rule record type: ▪ 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
Enhancement Request	Supports an additional record type for product enhancement requests.	Adds the Enhancement Request record type.	Fields included in EnhancementRequest record type: ▪ 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

Package	Description	Added or modified	Fields
History (read-only)	Lets you keep a historical account of all actions taken on a record.	Adds a **History** tab to the enabled record type.	No fields added.
Notes	Lets you keep a historical account of all notes that were 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: ▪ 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: ▪ PQC_DiagnosticTool ▪ PQC_Executable ▪ PQC_TestCommand ▪ PQC_TestTool ▪ PQC_Stack ▪ PQC_StackID
Project	Lets you 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: ▪ Name ▪ Description Fields added to enabled record type: ▪ Project
Repository (read-only)	Provides support needed for both Rational RequisitePro, Rational Administrator, and Rational TeamTest.	Creates an RAProject stateless record type.	Fields included in RAProject record type: ▪ Name ▪ TT_Repo (Refers to the TeamTest Repository) ▪ RA_Project_Path Fields added to enabled record type: ▪ RAProject

Package	Description	Added or modified	Fields
RequisitePro (read-only)	Provides support for integration with Rational RequisitePro.	Adds the Requirement and RequirementMap stateless record types. Adds the **Requirements** tab to the enabled record type. Also adds the ASCQIBase base action to the enabled record type.	Fields included in Requirement record type: ▪ Name ▪ Project_Name ▪ Req_GUID ▪ Req_ID ▪ Requirement ▪ Tag Fields included in RequirementMap record type: ▪ CQBackReqListAttName ▪ CQDatabase ▪ CQDatabasePath ▪ CQDialogTitle ▪ CQEntityDefName ▪ CQHelpContextID ▪ CQModifyAction ▪ CQReqListAttName ▪ CQRepoProjectAttName ▪ HelpFileName ▪ RPAttrGUID ▪ RPHelpContextID ▪ RPProjectName ▪ RPProjectPath ▪ RPReqTypeGUID Fields added to enabled record type: ▪ Requirements_List
Resolution	Adds support for tracking how a record was resolved. Requires you to map schema states to the following state types: Not_Resolved; Resolved.	Adds a **Resolution** tab to the enabled record type.	Fields added to enabled record type: ▪ Resolution ▪ Resolution_Statetype (read-only)

Package	Description	Added or modified	Fields
TeamTest (read-only)	Provides support for integration with Rational TeamTest.	Adds **Test Data** and **Environment** tabs to the record type you specify. Also adds a TestInput stateless record type.	Fields added to enabled record type: ▪ 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 Fields added to the TestInput record type: ▪ Test_Input_Name ▪ Test_Input_ID ▪ Source_UID
UCMPolicy Scripts	Provides support for the UnifiedChange Management (UCM) package by adding three global scripts.	Does not add any record types.	

Package	Description	Added or modified	Fields
UnifiedChange Management (UCM) (read-only)	Provides support for the UCM process by enabling integration with Rational ClearCase 4.0 and higher; links a ClearCase Project VOB with a ClearQuest user database. Requires the UCMPolicyScripts package. Can be used with the BaseCMActivity package. Requires you to map schema states to the following state types: Waiting, Active, Ready, Complete.	Adds the UCMUtilityActivity record type. Adds UCM_Project stateless record type. Adds UCM queries to the client workspace in the Public Folders. Adds the ucm_base_ synchronize action to the enabled record type.	Fields included in UCMUtilityActivity record type: ▪ am_statetype ▪ Description ▪ Owner ▪ Headline ▪ ucm_vob_object ▪ ucm_stream_object ▪ ucm_stream ▪ ucm_view ▪ ucm_project Fields included in UCM_Project record type: ▪ name ▪ ucm_vob_object ▪ ucm_chk_before_deliver ▪ ucm_chk_before_work_on ▪ ucm_chk_mstr_before_dlvr ▪ ucm_cq_act_after_deliver Fields added to enabled record type: ▪ am_statetype ▪ ucm_vob_object ▪ ucm_stream_object ▪ ucm_stream ▪ ucm_view ▪ ucm_project

Package	Description	Added or modified	Fields
Visual SourceSafe (read-only)	Provides support for integration with Microsoft Visual SourceSafe.	Adds SSObject and SCSnapObject stateless record types. Adds the **SourceSafe** tab to the form of the enabled record type.	Fields added to enabled record type: ▪ VSSChangeSet Fields added in SCObject record type: ▪ CQDefects ▪ VSSCheckOutState ▪ VSSFileName ▪ VSSSpec ▪ VSSUser ▪ VSSVersion Fields added in SCSnapObject record type: ▪ CreatedBy ▪ CreatedOn ▪ Label ▪ SnapElements

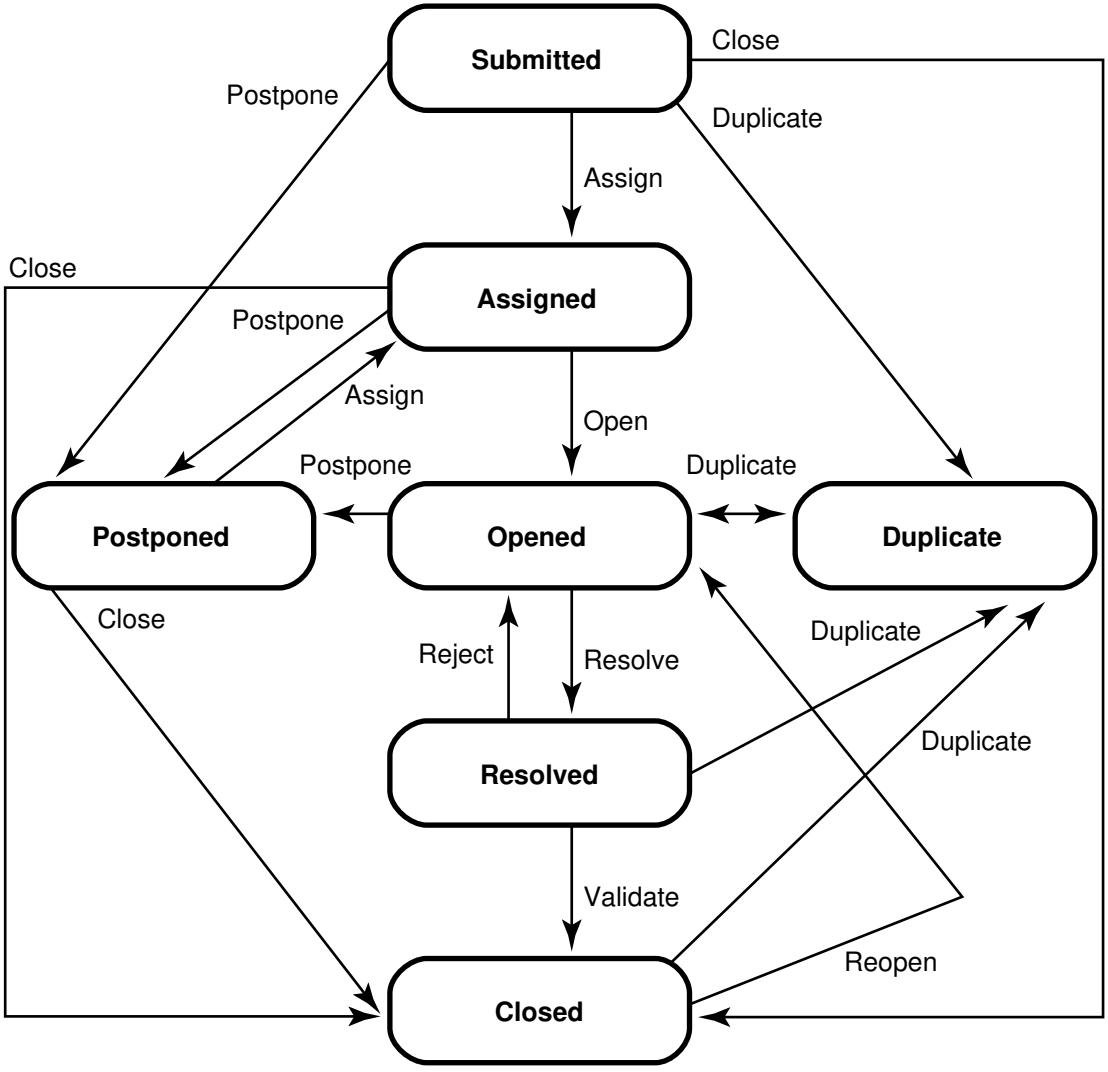
State Model for the Defect Record Type

The following table lists the states of the defect record type.

State	Description
Submitted	First state of a new defect.
Assigned	Assigned to a team member.
Opened	Being worked on.
Resolved	Was fixed.
Closed	Was 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.

Figure 7 State Model for Defect Record Type



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

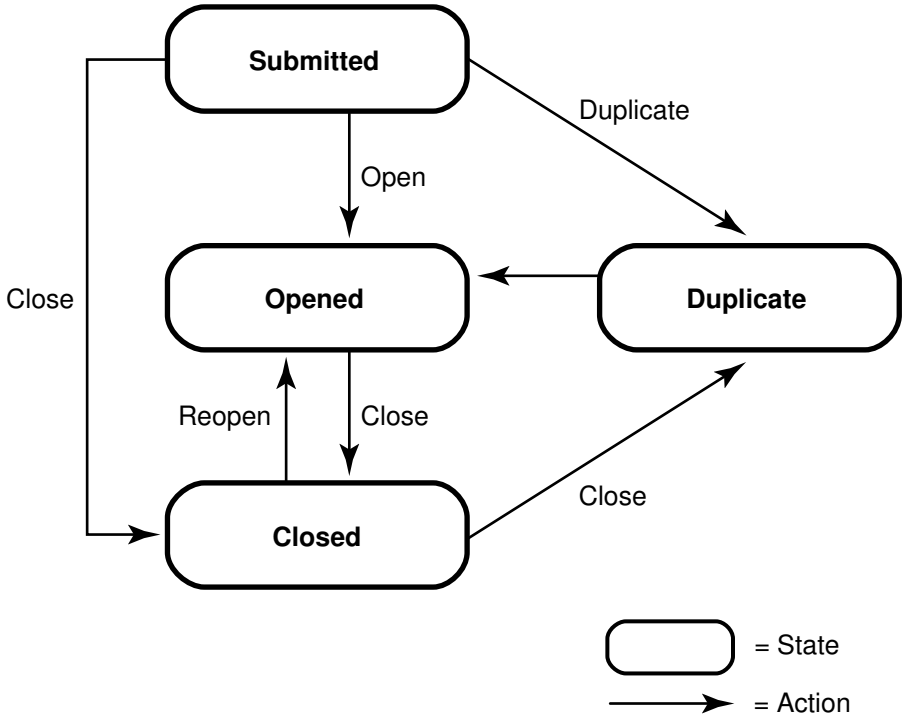
State Model for the EnhancementRequest Record Type

The following table lists the states of the EnhancementRequest record type.

State	Description
Submitted	First state of a new enhancement request.
Opened	Being worked on.
Closed	Was 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.

Figure 8 State Model for Enhancement Record Type



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. The ClearQuest Resolution package uses state types to invoke certain hooks when a record moves to a state mapped to the Resolved state type.

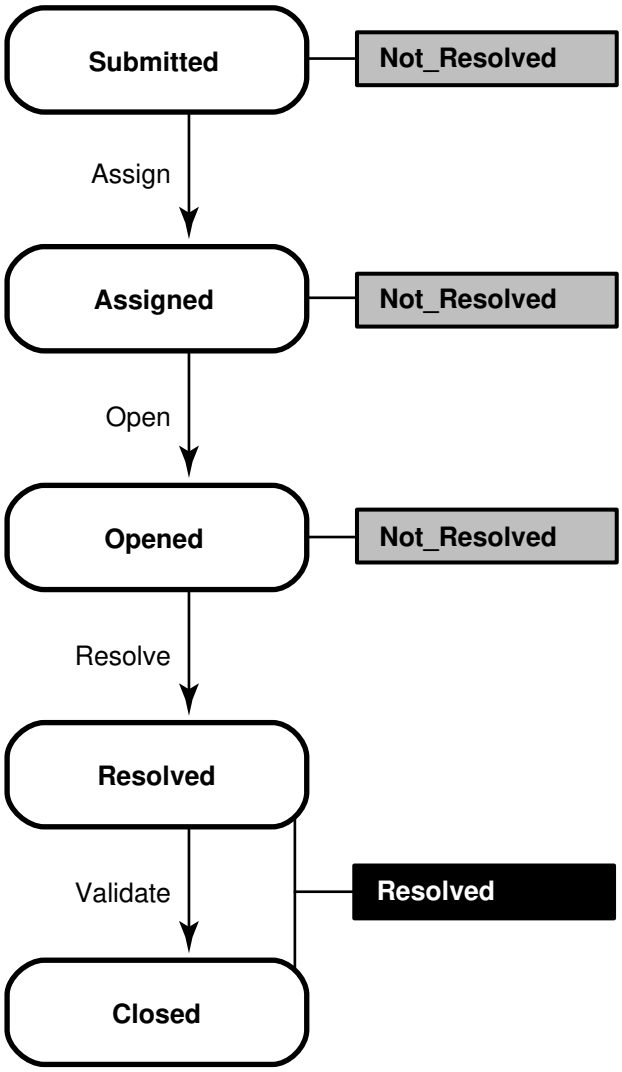
Note: When you add a new state to a schema that uses state types, you must map the new state to a state type. See *Mapping State Types* on page 116.

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.

Figure 9 State Type Model for Resolution Package



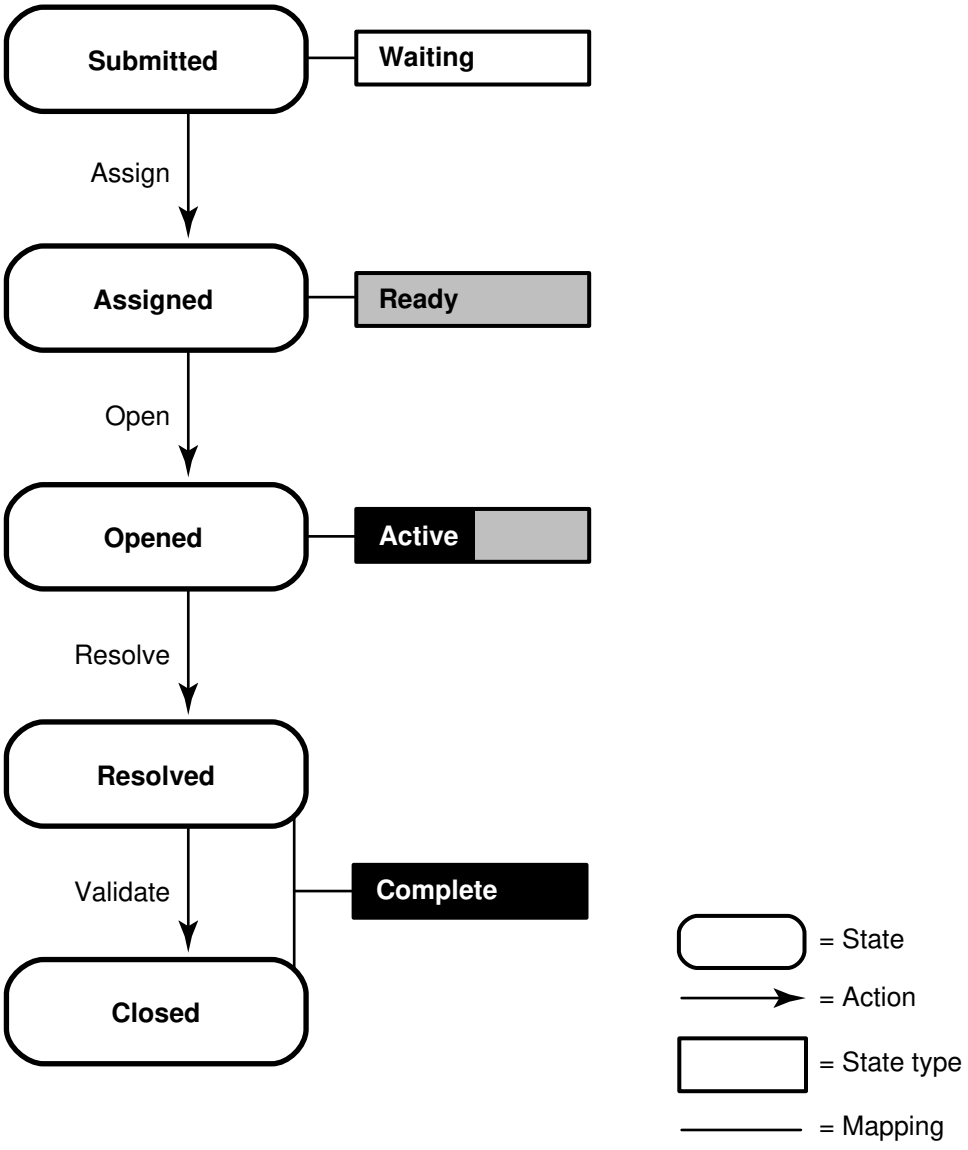
UnifiedChangeManagement Package State Type Model

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.

See Appendix B, *Adding ClearQuest Integrations* for more information.

State type	Description	Mapped state in UCM schema
Waiting	Record not yet assigned, triaged, or scheduled.	Submitted Postponed
Ready	Record was 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

Figure 10 State Type Model for Enhancement Record Type



Adding ClearQuest Integrations

B

You can integrate Rational ClearQuest with software configuration management tools to create a complete change management system. This material describes the integrations available with ClearQuest.

The topics covered include:

- *Overview of ClearQuest Integrations* on page 295.
- *Viewing the Packages in Your Schema* on page 299.
- *Testing Integrations* on page 300.
- *Adding Independent Integrations* on page 300.
- *Adding Dependent Integrations* on page 301.
- *Applying Package Upgrades* on page 312.

Note: The instructions provided in these topics assume that you know how to work with ClearQuest schemas. See Chapter 4, *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.

It is important to note that the ClearQuest integrations are not applied to the Web Client, therefore, fields, forms, reports, scripts and other functionality added to the ClearQuest Client for Windows or UNIX by an integration package will not be available through the Web Client.

Note: These instructions 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 *Applying Package Upgrades* on page 312.

Warning: Be sure to plan carefully before adding packages. After 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 following integrations are independent integrations. You can use the same installation process for all independent integrations. See *Adding Independent Integrations* on page 300.

- Rational ClearCase and ClearQuest

  Associates one or more ClearQuest change requests with one or more ClearCase versions.

- Rational Suite ContentStudio and ClearQuest

  Allows you to submit and track content changes with ClearQuest.

- Rational PureCoverage and ClearQuest

  Allows you to submit code coverage data to a ClearQuest database and track it.

- Rational Purify and ClearQuest

  Allows you to submit data to a ClearQuest database and track it.

- Rational Quantify and ClearQuest

  Allows you to submit performance data to a ClearQuest database and track it.

- Your e-mail system and 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 12, *Administering ClearQuest E-Mail*.

Dependent Integrations

The following integrations are dependent integrations. For more information about dependent integrations, see *Adding Independent Integrations* on page 300.

- Rational Administrator and ClearQuest

  Associates Rational projects with ClearQuest databases. See *Adding a Rational Administrator Integration* on page 302.

- Rational ClearQuest Project Tracker and ClearQuest

  Allows you to exchange project data between the two systems. See *Adding a Rational ClearQuest Project Tracker Integration* on page 303.

- Rational RequisitePro and ClearQuest

  Associates RequisitePro requirements with ClearQuest records.

  For information on how to add a RequisitePro/ClearQuest integration, see the *Administrator's Guide* for Rational Suite.

- Rational TeamTest and ClearQuest

  Allows you to submit defects found through TeamTest to ClearQuest databases, and to track them. See *Adding a Rational TeamTest Integration* on page 304.

- Rational Unified Change Management (UCM) and ClearQuest

  Links ClearCase UCM projects and activities to ClearQuest records. See *Adding a Rational UCM Integration* on page 305.

- Microsoft Visual SourceSafe and ClearQuest

  Associates Visual SourceSafe information with ClearQuest records. See *Adding a Microsoft Visual SourceSafe Integration* on page 310.

ClearQuest Integrations and Code Pages

Because all data in a ClearQuest database must be from the same code page, there are several things you need to keep in mind when using ClearQuest integrations:

- To enter data into a ClearQuest database through an integration, the integrated product must be running on a machine with the same operating system code page as the ClearQuest data code page of the database set. ClearQuest blocks any data from an integrated product that uses a different code page.
- If your ClearQuest data code page value is set to ASCII, you may not be able to integrate ClearQuest with other products. The products that ClearQuest integrates with do not limit data entry to only ASCII characters. A ClearQuest database set with an ASCII data code page value will reject all non-ASCII characters coming from the integrated product.

Example

The following example shows how a code page mismatch can prevent an integration between ClearQuest and Rational Administrator from functioning correctly.

Rational ClearQuest is installed on a machine with the **1252** operating system code page and the ClearQuest data code page value for the database set is also **1252**.

The Rational Administrator client is installed on a machine with the **936** (Japanese) operating system code page.

A user creates a Rational Administrator project, and because she is working on a machine with a **936** code page, the project name includes Japanese characters. She then attempts to associate the new Rational Administrator project with a ClearQuest database. The attempt fails because the Japanese characters in the Rational Administrator project name are not from the **1252** code page, which is the data code page value of the ClearQuest database set.

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 279.

To enable package functionality for a new record type:

1 In ClearQuest Designer, click **File > Open Schema** and select a schema to open.

2 With the schema open, click **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 packages that you want to enable the record type.

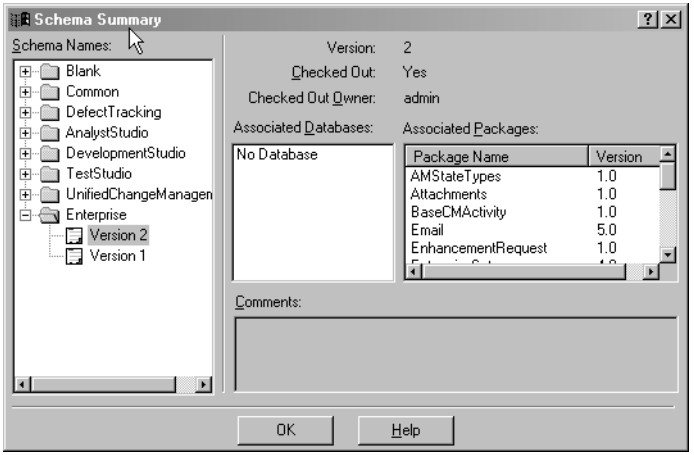
Viewing the Packages in Your Schema

Many packages are used by multiple integrations. Before you add any integrations to your schema, you should find out which packages are already in the schema.

To view a list of packages already in your schema:

1 In ClearQuest Designer, click **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 *Creating a Test Database* on page 80.

Adding Independent Integrations

Independent integrations require you to add the appropriate package as described in the following table.

Integration	Package to add	Additional documentation
Rational ClearCase (Base ClearCase)	ClearCase	ClearCase Help. In the table of contents, click **ClearCase User Interface > Integrations with other products > Base ClearCase integration with ClearQuest**. **Note:** The ClearCase integration described 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 305, automatically sets up predefined ClearCase policies.
Rational Purify	PQC	Rational Purify Help. Look up ClearQuest.
Rational Quantify	PQC	Rational Quantify Help. Look up ClearQuest.
Rational PureCoverage	PQC	Rational PureCoverage Help. Look up ClearQuest.

Integration	Package to add	Additional documentation
Rational Suite ContentStudio	ContentStudio	*Installing Rational Suite ContentStudio*

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 299.

If possible, add all integrations first in a test environment before adding the integration to your production database. See *Testing Integrations* on page 300.

Warning: Be sure to plan carefully before adding packages. After 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, click **File > Check In**.
2. Click **Package > Package Wizard** and install the desired package. See *Applying Packages* on page 129 for detailed instructions.
3. Validate the schema changes. See *Validating Schema Changes* on page 86 for detailed instructions.
4. Click **File > Check In** to save the schema changes. See *Checking In a Schema* on page 88 for detailed instructions.
5. Apply the schema changes to the user database by clicking **Database > Upgrade Database**. See *Upgrading a User Database* on page 89 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, 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 300.

Warning: Be sure to plan carefully before adding packages. After 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:

- Adding a Rational Administrator Integration.
- Adding a Rational ClearQuest Project Tracker Integration.
- Adding a Rational TeamTest Integration.
- Adding a Rational UCM Integration.
- Adding a Microsoft Visual SourceSafe Integration.
- Rational RequisitePro, see the *Administrator's Guide* for Rational Suite.

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 299. 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 298.

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, click **File > Check In**.
2. Click **Package > Package Wizard** and add the latest **Repository** package. See *Applying Packages* on page 129 for detailed instructions.

Saving the Schema Changes

After you add the Repository package:

1. Validate the schema changes. See *Validating Schema Changes* on page 86 for detailed instructions.
2. Click **File > Check In** to save the schema changes. See *Checking In a Schema* on page 88 for detailed instructions.
3. Apply schema changes to the user database by clicking **Database > Upgrade Database**. See *Upgrading a User Database* on page 89 for detailed instructions.

Configuring Rational Administrator

Configure the Rational Administrator application as needed. See the Rational Administrator Help for additional configuration information.

Adding a Rational ClearQuest Project Tracker Integration

Integrating with Project Tracker allows you to exchange project data between Project Tracker and ClearQuest. The Project Tracker integration requires that the following be done in the order described:

1. Adding the AMBaseActivity Package
2. Adding the AMWorkActivitySchedule Package
3. Saving the Schema Changes
4. Configuring Rational ClearQuest Project Tracker

Warning: To avoid errors, you must install packages in the order described.

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 299. 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 298.

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, click **File > Check In**.
2. Click **Package > Package Wizard** and select the latest **AMBaseActivity** package to add. See *Applying Packages* on page 129 for detailed instructions.
3. Check the schema back into the schema repository by clicking **File > Check In**.

Note: The AMBaseActivity package cannot be applied to a BaseCmActivity record type when setting up Project Tracker. BaseCmActivity is a UCM record type. The AmBaseActivity package can only be applied to Defect and Enhancement Request record types.

Adding the AMWorkActivitySchedule Package

With the schema checked out, click **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 86 for detailed instructions.
2. Click **File > Check In** to save the schema changes. See *Checking In a Schema* on page 88 for detailed instructions.
3. Apply schema changes to the user database by clicking **Database > Upgrade Database**. See *Upgrading a User Database* on page 89 for detailed instructions.

Configuring Rational ClearQuest Project Tracker

Configure Project Tracker as needed. See the manual *Using Project Tracker* for Rational ClearQuest 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 that the following be done in the order described:

1. Adding the Repository Package.
2. Adding the TeamTest Package.
3. Saving the Schema Changes.
4. Configuring Rational TeamTest.

Warning: To avoid errors, you must install packages in the order described.

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 299. 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 298.

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, click **File > Check In**.
2. Click **Package > Package Wizard** and add the latest **Repository** package. See *Applying Packages* on page 129 for detailed instructions.
3. Check the schema back into the schema repository by clicking **File > Check In**.

Adding the TeamTest Package

With the schema checked out, click **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 86 for detailed instructions.
2. Click **File > Check In** to save the schema changes. See *Checking In a Schema* on page 88 for detailed instructions.
3. Apply schema changes to the user database by clicking **Database > Upgrade Database**. See *Upgrading a User Database* on page 89 for detailed instructions.

Configuring Rational TeamTest

Configure the TeamTest application as needed. See the Rational TeamTest Help for more 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 Chapter 4, *Working with ClearQuest Schemas*.

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 300 for instructions on adding a Base ClearCase integration.

Integrating Rational UCM with packages requires that the following be done in the order described:

1. Adding the AMStateTypes Package.
2. Setting the Default Actions for UCM.
3. Adding the UCMPolicyScripts Package.
4. Adding the UnifiedChangeManagement Package.
5. Adding the BaseCMActivity Package (Optional).
6. Saving the Schema Changes.
7. Configuring Rational UCM.

Warning: To avoid errors, you must install packages in the order described.

Note: To find out if the AMStateType, UCMPolicyScripts, UnifiedChangeManagement, and BaseCMActivity packages are already installed in your schema, see *Viewing the Packages in Your Schema* on page 299. 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 298.

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

Adding the AMStateTypes Package

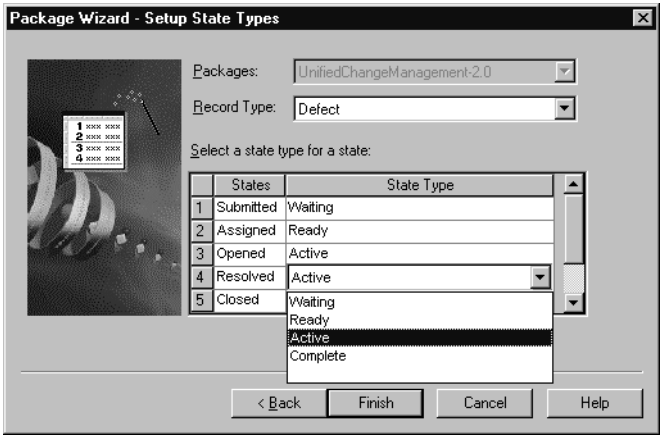
1. In ClearQuest Designer, make sure the schema you want to add the package to is checked in. To check in a schema, click **File > Check In**.
2. Click **Package > Package Wizard** and add the latest **AMStateType** package. See *Applying Packages* on page 129 for detailed instructions.

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 116.



5 Repeat the state type mapping for each record type you enabled. Click **Finish**.

ClearQuest automatically validates your schema. The validation window will indicate that you need to set default actions.

Setting the Default Actions for UCM

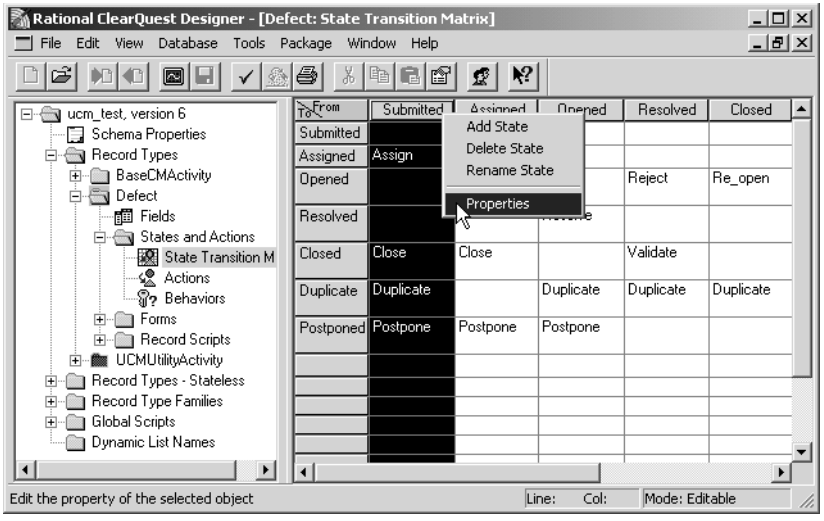
The State Transition Matrix in your schema must provide at least one path through the state type model for the UnifiedChangeManagement package, from the Waiting state type, to Ready, to Active, to Complete. See *Displaying a State Transition Matrix* on page 115, and *UnifiedChangeManagement Package State Type Model* on page 292 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 124 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 click **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 Click **File > Check In** to check in the schema. See *Checking In a Schema* on page 88 for detailed instructions.

For more information, see *Using Default Actions* on page 124 and *UnifiedChangeManagement Package State Type Model* on page 292.

Adding the UCMPolicyScripts Package

1 With the schema checked in, click **Package > Package Wizard** and add the latest **UCMPolicyScripts**. See *Applying Packages* on page 129 for detailed instructions.

2 Select **File > Check In** to check in the schema.

Adding the UnifiedChangeManagement Package

1 With the schema checked in, click **Package > Package Wizard** and add the latest **UnifiedChangeManagement** package.

2 Select **File > Check In** to check in the schema.

Adding the BaseCMActivity Package (Optional)

The BaseCMActivity 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 the manual *Using Project Tracker* for Rational ClearQuest.

1 With the schema checked in, click **Package > Package Wizard** and add the latest **BaseCMActivity** package.

2 Edit the state types for the **BaseCMActivity** package. See *Applying Packages* on page 129 for detailed instructions.

3 Click **File > Check In** to check in the schema.

Saving the Schema Changes

After installing the last UCM package, do the following:

1 Validate the schema changes. See *Validating Schema Changes* on page 86 for detailed instructions.

2 Click **File > Check In** to save the schema changes. See *Checking In a Schema* on page 88 for detailed instructions.

3 Apply schema changes to the user database by clicking **Database > Upgrade Database**. See *Upgrading a User Database* on page 89.

Configuring Rational UCM

Configure the UCM application as needed. See *Managing Software Projects with ClearCase* for more 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 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 299. 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 298 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, click **File > Check In**.

2 Click **Package > Package Wizard** and add the latest **Visual SourceSafe** package. See *Applying Packages* on page 129.

Saving the Schema Changes

After installing the Visual SourceSafe package:

1 Validate the schema changes. See *Validating Schema Changes* on page 86 for detailed instructions.

2 Click **File > Check In** to save the schema changes. See *Checking In a Schema* on page 88.

3 Apply schema changes to the user database by clicking **Database > Upgrade Database**. See *Upgrading a User Database* on page 89.

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 Help for detailed instructions.

Setting Up Each Client Machine

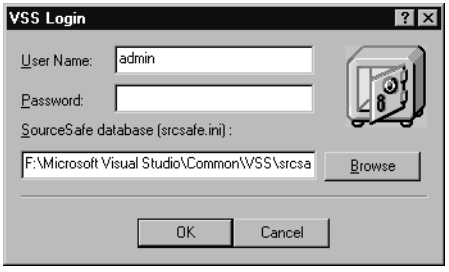
Note: When you run the ClearQuest Visual Source Safe tool, cqvss.exe, the information you enter is stored in a .ini file, in the following location:

```
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, click **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 311 and click **OK**.

Applying Package Upgrades

Note: If you are upgrading a RequisitePro integration with ClearQuest, see "Upgrading ClearQuest" in the *Upgrade Guide* for Rational Suite.

To upgrade all your existing packages:

1. Log in to ClearQuest Designer with Super User or Schema Designer privileges.
2. If the Open Schema dialog box is displayed, click **Cancel**.
3. In ClearQuest Designer, click **Package > Upgrade Installed Packages**.
4. Select the schema name from the list of schema repositories and click **Next**.
5. A list of all the packages that need to be applied appears. To continue, click **Upgrade**.
6. After the upgrade is complete, click **Exit**.

   If you are upgrading a UCM integration from release 2000.02.10, map the UCM state types prior to testing the upgrade in Step 7. To do so:

   a. In ClearQuest Designer, click **Package > Setup State Types**.

   b. In the Package Wizard, click **AMStateTypes** from the list of **Packages** and accept the default value for the **Record Type**.

   c. Map each of the ClearQuest states (shown in the right column) to the appropriate State Type from the pull-down menus in the **State Type** or left column.

   d. If you have UCM integrations with additional record types, they will be listed in the **Record Type** scroll-down menu. Select the next record type and repeat the mapping process for each of the record types.

   Note: For more information on mapping state types, see *Mapping State Types* on page 116.
7. Verify your work by clicking **File > Test Work** from the ClearQuest Designer menu. Select the test database associated with the chosen schema and click **OK** to continue. For more information on testing schema changes, see *Creating a Test Database* on page 80.
8. After the testing is complete and you verify that there are no errors, click **Next**.

   Note: If the upgrade fails, the errors are displayed in the status window.
9. Check in your changes by clicking **File > Check in** and click **OK**.

Viewing the Package Upgrade Status

The package upgrade displays the status as it completes the process. A typical status looks similar to the following example.

The following packages will be upgraded for DefectTracking:

```
Email 1.0 will be upgraded to revision 6.0.

Notes 1.0 will be upgraded to revision 3.0.

Customer 1.0 will be upgraded to revision 2.0.

UnifiedChangeManagement 3.0 will be upgraded to revision 4.0.

Package upgrade was successful.  Please click on Exit to close the
wizard.  The schema will be opened for you to view the changes.
```

Hook Examples

C

Hook examples provide a general idea of how you might add hooks to your schema. For readability, the examples to 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.

Because hooks run with administrator privileges, these examples work even if the behavior of the fields is read-only.

This section provides the following hook examples:

Field Hook Examples

- *Field Choice List Hook Example* on page 316
- *Hook for Creating a Dependent List* on page 317
- *Field Choice List Hook to Display User Information* on page 318
- *Field Default Value Hook Examples* on page 320
- *Field Permission Hook Example* on page 322
- *Field Validation Hook Example* on page 323
- *Field Value Changed Hook Example* on page 324

Action Hook Examples

- *Action Initialization Hook Example* on page 325
- *Action Initialization Hook for a Field Value* on page 326
- *Action Hook for Setting the Value of a Parent Record* on page 327
- *Action Access Control Hook Example* on page 331
- *Action Commit Hook Example* on page 332
- *Action Notification Hook Example* on page 334
- *Action Validation Hook Example* on page 337

Record Script Example on page 340

Global Script Example on page 341

Using CAL Methods in ClearQuest Hook Scripts Example on page 342

Advanced Reporting and Automation with cqperl on page 345

For more information, see the *API Reference* for Rational ClearQuest.

Field Hook Examples

Field Choice List Hook Example

Choice list hooks allow you to build a list of choices for the user. When ClearQuest calls a choice list hook, it provides an instance of the HookChoices class in the choices parameter. You use this object in your hook to add items to the list or to sort the existing list items. If you do not sort the items in the choice list, they appear in the order in which they were added to the list.

The following example builds a choice list whose contents are the names of various operating systems.

VBScript

```
Sub OS_type_ChoiceList(fieldname, choices)
    ' fieldname As String
    ' choices As Object
    ' entityDef = defect
    choices.AddItem("Solaris")
    choices.AddItem("Windows")
    choices.AddItem("HP/UX")
End Sub
```

Perl

```
sub OS_type_ChoiceList {
    my($fieldname) = @_;
    my @choices;
    # $fieldname as string scalar
    # @choices as string array
    # entityDef is Defect
    # use array operation to add items.  Example:
    # push(@choices, "red", "green", "blue");
    push(@choices, "Solaris", "Windows", "HP/UX");
    return @choices;
}
```

Hook for Creating a Dependent List

The following example 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;
}
```

Field Default Value Hook Examples

When the user creates a new record, ClearQuest creates the record and initializes its fields to appropriate default values. If a particular field has a default-value hook associated with it, ClearQuest runs the hook to set the field's value. If no hook is associated with the field, ClearQuest assigns a default value that is appropriate for the field's type. For example, integer fields are set to 0, string fields are set to an empty string, and list fields are set to an empty list. ClearQuest does not initialize date fields to a default value; you should always provide a default-value hook for date fields.

Note: Because hooks run with administrator privileges, these examples work even if the behavior of the fields is read-only.

Example 1

The following example shows you how to initialize a date field to the current date and time:

VBScript

```
Sub submit_date_DefaultValue(fieldname)
    ' fieldname As String
    ' entityDef = defect
    call SetFieldValue(fieldname, now)
End Sub
```

Perl

```
# Define a function for the current timestamp
```

```
sub GetCurrentDate {
   my ($sec, $min, $hour, $mday, $mon, $year, $wday, $yday,
       $time)=localtime();
   if ($year < 100) {
    $year = 19 . $year;
   } else {
     $year = 2000 + $year - 100;
   }
   return sprintf("%2.2d/%2.2d/%4s %2.2d:%2.2d:%2.2d", $mon + 1,
$mday,
     $year, $hour, $min, $sec);
}
# Define a routine to call the timestamp function
sub Submit_Date_DefaultValue {
   my($fieldname) = @_;
   # $fieldname as a string scalar and entityDef is Defect
   $entity->SetFieldValue($fieldname, GetCurrentDate());
}
```

Example 2

The following example initializes the 'submitter' field to the login name of the person who is submitting the record:

VBScript

```
Sub submitter_DefaultValue(fieldname)
    ' fieldname As String
    ' entityDef = swbug
    SetFieldValue fieldname, GetSession().GetUserLoginName()
End Sub
```

Perl

```
sub Submitter_DefaultValue {
    my($fieldname) = @_;
    # $fieldname as a string scalar
    # entityDef is Defect
    my $session;
    my $username;
```

```
    $session = $entity->GetSession();
    $username = $session->GetUserLoginName();
    $entity->SetFieldValue($fieldname, $username);
}
```

Field Permission Hook Example

Permission hooks allow you to define the behavior of a field at runtime. Normally, you define the behavior of the field using the Behaviors grid in ClearQuest Designer. The values you enter in the Behaviors grid apply equally to all members of a user group. Using a permission hook, you can specify the behavior of a field with more precision. In the following example, if the current user belongs to the *managers* group and does *not* belong to the *engineering* group, this hook makes the field optional. If the user is not in at least one group, the hook fails.

Note: Because hooks run with administrator privileges, this example works even if the behavior of the fields is read-only.

VBScript

```
Function field1_Permission(fieldname, username)
    ' fieldname As String
    ' username As String
    ' field_Permission As Long
    ' entityDef = defect

    ' Assign the default return value
    set curSession = GetSession
    userGroups = curSession.GetUserGroups()
    for each group in userGroups
      if group = "managers" And group <> "engineers" Then
        field1_Permission = AD_OPTIONAL
      End If
    Next
End Function
```

Perl

```
sub field1_Permission  {
    my($fieldname, $username) = @_;
    my $result;
```

```
    # $fieldname as string scalar
    # $username as string scalar
    # $result as long scalar
    # entityDef is Defect

    # Assign the default return value
    $curSession = $entity->GetSession();
    $userGroups = $curSession->GetUserGroups();
    foreach $group (@$userGroups)  {
      if ($group eq "managers" && $group ne "engineers") {
         $result = $CQPerlExt::CQ_OPTIONAL;
      }
    }
    return $result;
}
```

Field Validation Hook Example

Validation hooks allow you to verify that a field contains an appropriate value. ClearQuest calls your validation hooks at predefined times to make sure the field's contents are valid. If a record contains any fields with invalid values, ClearQuest prevents the record from being committed to the database until the error has been corrected.

The advantage of using hooks to validate individual fields (as opposed to using an action validation hook to validate the entire record) is that the user is notified immediately if the given field value is invalid.

Field validation hooks are written as functions that return a string value. The function's return value is considered to be an error message. If the return value is an empty string, ClearQuest considers the field value to be valid.

In the following example, if fewer than 10 characters were typed into the field, the validation hook rejects the entry, forcing the user to type at least 10 characters.

VBScript

```
Function word_Validation(fieldname)
    ' fieldname As String
    ' word_Validation As String
    ' entityDef = puzzle_words
```

```
    Dim val
    val = GetFieldValue(fieldname).GetValue()
    If Len(val) < 10 Then
      word_Validation = "All words must be at least 10 letters long"
    End If
End Function
```

Perl

```
sub word_Validation  {
    my($fieldname) = @_;
    # $fieldname as string scalar
    # $result as string scalar
    # $entityDef = puzzle_words
    my($value);

    $value = $entity->GetFieldValue($fieldname)->GetValue();
    if (length ($value) < 10)  {
      $result = "All words must be at least 10 letters long";
    }
    return $result;
}
```

Field Value Changed Hook Example

Value-changed hooks allow you to synchronize fields or perform other tasks after the value in a field has changed.

In the following example, the hook checks the name of the operating system stored in the current field. Depending on the operating system, the hook then assigns a version number to the OS_version field. If the current field has not yet been set, and thus does not contain the name of the operating system, this hook does not set the corresponding version number.

VBScript

```
Sub OS_type_ValueChanged(fieldname)
    ' fieldname As String
    value = GetFieldValue(fieldname).GetValue()

     If value = "solaris" Then
```

```
        SetFieldValue "OS_version", "7.x"
      ElseIf value = "windows" Then
        SetFieldValue "OS_version", "95"
      ElseIf value = "hpux" Then
        SetFieldValue "OS_version", "10.x"
      End If
End Sub
```

Perl

```
sub OS_type_ValueChanged  {
    my($fieldname) = @_;
    my($value);
    $value = $entity->GetFieldValue($fieldname)->GetValue();

    if ($value eq "solaris")  {
      $entity->SetFieldValue("OS_version", "7.x”);
    } elsif ($value eq "windows")  {
      $entity->SetFieldValue("OS_version", "95");
    } elsif ($value eq "hpux")  {
      $entity->SetFieldValue("OS_version",  "10.x”);
    }
}
```

Action Hook Examples

Action Initialization Hook Example

Initialization hooks allow you to perform complex initialization at the beginning of an action. You can use this hook to reset fields or assign different values to fields based on the type of action.

The following example demonstrates a hook that is run when the user attempts to reassign a defect to a different user. The hook clears the contents of the *action_reason* field at the beginning of a *reassign* action. If the behavior of this field is set to *Mandatary,* the user must enter a reason for reassigning the action.

VBScript

```
Sub swbug_Initialization(actionname, actiontype)
```

```
    ' actionname As String
    ' actiontype As Long
    ' action = reassign
    ' Empty the string at the beginning of the action
    SetFieldValue "action_reason", ""
End Sub
```

Perl

```
sub swsub_Initialization {
    my($actionname, $actiontype) = @_;
    # $actionname as string scalar
    # $actiontype as long scalar
    # action is reassign
    # do any setup for the action here
    # Empty the string at the beginning of the action
    $entity->SetFieldValue("action_reason", "");
}
```

Note: If your schema requires the user to enter a reason for initiating each action, you can use a DEFAULT_VALUE hook to clear the *action_reason* field at the beginning of each action. In the preceding example, clearing the field is only required for a *reassign* action, so the action initialization hook is more appropriate.

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 along 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 109).

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

VBScript

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

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

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

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

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

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


    SessionObj.OutputDebugString "StateStatus is: " & StateStatus &
vbCrLf
    SessionObj.OutputDebugString "CurrentState is: " & CurrentState
&vbCrLf
    SessionObj.OutputDebugString "SameState is: " & SameState & vbCrLf
    if StateStatus = "" then
      StateStatus = CurrentState
      SameState = 1
      SessionObj.OutputDebugString "coming to statestatus is null" &
vbCrLf


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


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


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

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

Perl

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

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

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

foreach $ChildID (@ChildArray) {

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

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

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

    if ($StateStatus eq "") {
        $StateStatus = $CurrentState;
        $SameState = 1;
        $SessionObj->OutputDebugString("coming to statestatus is
null\n");

    } elsif ($StateStatus eq $CurrentState)  {
        SessionObj->OutputDebugString ("coming to same state\n");
        $SameState = 1;

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

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

    if ($status ne "")  {
        $SessionObj->OutputDebugString ("error when updating parent
state:
           $status\n");
    $ParentObj->Revert();
    return -1;  # Exit
    }
    $ParentObj->Commit();
}    #end if for ($SameState == 1)
}    #end if for ($ParentID ne "")
```

Action Access Control Hook Example

Access-control hooks let you restrict access to particular actions based on a specific set of criteria. In ClearQuest Designer, you can limit actions to specific groups of users by choosing a hook type of **User Group**, or you can choose **All Users** to give everyone access to the action. You can also choose the **Scripts** option and write a VBScript or Perl hook to determine access.

The following example shows you how to limit access to a single user named "Pat."

VBScript

```
Function swbug_AccessControl(actionname, actiontype, username)
    ' actionname As String
    ' actiontype As Long
    ' username As String
    ' swbug_AccessControl As Boolean
    ' action = close

    Dim is_ok
    ' Test whether the current user has the privilege to close this bug
    If username = "Pat" Then
      is_ok = TRUE
    Else
      is_ok = FALSE
    End If
    swbug_AccessControl = is_ok

End Function
```

Perl

```
sub swbug_AccessControl {
    my($actioname, $actiontype, $username) = @_;
    my $result;
    # $actionname string scalar, $actiontype as long scalar
    # $username as string scalar, # action is Close
    # return TRUE if the user has permission to perform this action

    if ($username eq "Pat") {
      $result = 1;
```

```
    } else {
      $result = 0;
    }
    return $result;
}
```

Action Commit Hook Example

Commit hooks allow you to perform additional actions before a record is committed to the database.

The following example checks whether a defect has duplicates (*dups*). If the original defect was marked as *tested*, the hook marks the duplicates as *dupdone*, indicating that they should be evaluated again to verify that they are fixed. If there is a failure to commit one of these updates, all of the database transactions will be rolled back, including the one to which this hook belongs.

VB Script

```
Sub swbug_Commit(actionname, actiontype)
    ' actionname As String
    ' actiontype As Long
    ' action = tested
    Dim dups  ' Array of all direct duplicates of this defect
    Dim dupsvar ' Variant containing link to a duplicate
    Dim dupsobj ' The same link, but as an object rather than a variant
    Dim record  ' The record extracted from the link
    Dim session
    Dim parent_id ' The display name of this defect

    ' Make an API to call to see if this record has duplicates
    If HasDuplicates() Then
      Set session = GetSession
      dups = GetDuplicates
      parent_id = GetDisplayName
      For Each dupvar In dups
        Set dupobj = dupvar
        Set entity = dupobj.GetChildEntity
        session.EditEntity entity, "dupdone"
      entity.SetFieldValue "action_reason", "Original " & parent_id &
```

```
            " is tested"
          entity.Validate
          entity.Commit
        Next
      End If
End Sub
```

Perl

```
sub swbug_Commit {
    my($actionname, $actiontype) = @_;
    # $actionname As string scalar
    # $actiontype as long scalar
    # action is Submit
    # This hook is fired during the "commit" step of an
    # entity update. It is the appropriate place to put an
    # activity which should be bundled into the same
    # transaction as the commit, such as subactions
    # or updates of external data storage.

    my ($dups, # Array of all direct duplicates of this defect
    $record,   # The record extracted from the link
    $parent_id, # The display name of this defect
    $session,
    $locEntity,
    $dupobj
    );
    # Make an API to call to see if this record has duplicates
    if ($entity->HasDuplicates()) {
      $session = $entity->GetSession();
      $dups = $entity->GetDuplicates();
      $parent_id = $entity->GetDisplayName();
       my $count = $dups->Count();
       my $i = 0;
       for (i=0;$i<$count;$i++){
           $dupobj = $dups->Item($i);
           $locEntity = $dupobj->GetChildEntity();
           $session->EditEntity($locEntity, "dupdone");
```

```
            $locEntity->SetFieldValue("action_reason", "Original "
              . $parent_id . " is tested");
            $locEntity->Validate();
            $locEntity->Commit();
        }
    }
}
```

Action Notification Hook Example

Notification hooks allow you to trigger additional actions after a set of changes are committed to the database. You can send an e-mail notification to one or more users, or modify other related records. (You can also create an e-mail rule to send e-mail messages. See *Creating E-Mail Rules* on page 236.)

Note: You will need to stop and restart the Rational ClearQuest Mail Service every time you make changes to an e-mail notification hook.

The following example displays a dialog box for each field in the defect that was modified. You might also use a notification hook to generate an e-mail message and send it to an appropriate distribution list.

Note: An action that is itself initiated from a hook will not fire a notification unless you set the session variable **CQHookExecute** to a value of **1** in the hook script. This variable has a type of **Long** in VBScript and **long scalar** in Perl.

VBScript

```
Sub swbug_Notification(actionname, actiontype)
    ' actionname As String
    ' actiontype As Long
    ' action = modify
    ' Note: don't use MsgBox for web-based databases
    MsgBox "Modify action completed, notification hook started"

    fieldnames = GetFieldNames
    If IsArray(fieldnames) Then
      I = LBound(fieldnames)
      limit = UBound(fieldnames) + 1

      ' Get three kinds of values
      Do While I < limit
```

```
        onename = fieldnames(I)
        Set oldinfo = GetFieldOriginalValue(onename)
        Set newinfo = GetFieldValue(onename)
        oldstat = oldinfo.GetValueStatus
        If oldstat = AD_HAS_NO_VALUE Then
            oldempty = True
        Else
            oldempty = False
            oldval = oldinfo.GetValue
        End If

        newstat = newinfo.GetValueStatus
        If newstat = AD_HAS_NO_VALUE Then
          newempty = True
        Else
          newempty = False
          newval = newinfo.GetValue
        End If

      ' Compare the values
      If oldstat = AD_VALUE_UNAVAILABLE Then
        MsgBox "Field " & onename & ": original value unknown"
      Else
        If newempty And Not oldempty Then
          MsgBox "Field " & onename & " had its value deleted"
        ElseIf oldempty And Not newempty Then
          MsgBox "Field " & onename & " now= " & newval
        ElseIf oldval <> newval Then
          MsgBox "Field " & onename & " was= " & oldval
          MsgBox "Field " & onename & " now= " & newval
        Else
          MsgBox "Field " & onename & " is unchanged"
        End If
      End If
      I = I + 1
    Loop
End If
```

```
MsgBox "Modify action and notification hook completed"
End Sub
```

Perl

```
sub swsub_Notification {
    my($actionname, $actiontype) = @_;
    # $actionname as string scalar
    # $actiontype as long scalar
    # action is Submit
    # Post-commit notifications about actions may be handled here

    my ($fieldnames,
      $session,
      $fieldname,
      $oldinfo,
      $newinfo,
      $newstat,
      $oldstat,
      $oldval,
      $oldempty,
    );

    $session = $entity->GetSession();
    $fieldnames = $entity->GetFieldNames();

    # Get three kinds of values
    foreach $fieldname (@$fieldnames) {
      $oldinfo = $entity->GetFieldOriginalValue($fieldname);
      $newinfo = $entity->GetFieldValue($fieldname);
      $oldstat = $oldinfo->GetValueStatus();
      if ($oldstat eq CQPerlExt::CQ_HAS_NO_VALUE) {
        $oldempty = 1;
      } else {
        $oldempty = 0;
        $oldval = $oldinfo->GetValue();
      }
      $newstat = $newinfo->GetValueStatus();
```

```
      if ($newstat eq CQPerlExt::CQ_HAS_NO_VALUE) {
        $newempty = 1;
      } else {
           $newempty = 0;
        $newval = $oldinfo->GetValue();
      }

    # Compare the values
      if ($oldstat eq CQPerlExt::CQ_VALUE_UNAVAILABLE) {
        $session->OutputDebugString("Field " . $fieldname . ":
          original value unknown\n");
      } else {
        if ($newempty && !$oldempty) {
          $session->OutputDebugString ("Field " & $fieldname .
            " had its value deleted\n");
        } elsif ($oldempty && !$newempty) {
          $session->OutputDebugString ("Field " . $fieldname . " now =
" . $newval. "\n");
        } elsif ($oldval != $newval) {
          $session->OutputDebugString ("Field " . $fieldname . "
was = " .  $oldval. "\n");
          $session->OutputDebugString ("Field " . $fieldname . " now =
" . $newval. "\n");
        } else {
          $session->OutputDebugString ("Field " . $fieldname . " is
unchanged\n”);
          }
      }
    }
    $session->OutputDebugString ("Modify action & notification hook
completed\n");
}
```

Action Validation Hook Example

Action validation hooks let you check conditions that would be difficult to verify at the field level, such as the validity of a set of related field values. ClearQuest runs field-level validation hooks immediately after the field is modified; ClearQuest does

not run an action validation hook until the user is done editing the record and ready to commit it to the database.

The following example checks to see if the user entered a correct client operating system (OS) for the given project. If the specified OS is not supported for the given project, this method generates and returns a validation error message.

VBScript

```
Function defect_Validation(actionname, actiontype)
    ' actionname As String
    ' actiontype As Long
    ' defect_Validation As String
    ' action = teststate
    set sessionObj = GetSession
    ' Get the client OS platform the user indicated.
    platform = GetFieldValue("client_os").GetValue()
    ' Get the project name the user indicated. This information
    ' is stored on a referenced, stateless record.
    projectName = GetFieldValue("project.name").GetValue()
    ' Check the project name against the OS type. If the given project
    ' is not targeted for that platform, return a validation error.
    If projectName = "Gemini" Then
      If platform <> "NT" Then
         defect_Validation = "That project only supports NT."
      End If
    ElseIf projectName = "Aquarius" Then
      If platform <> "Unix" Then
         defect_Validation = "That project only supports Unix."
      End If
    End If
End Function
```

Perl

```
sub defect_Validation {
    my($actionname, $actiontype) = @_;
    my $result;
    # $actionname as string scalar
    # $actiontype as long scalar
```

```
    # $result as string scalar
    # action = teststate
    # Returns a non-empty string explaining why the action
    # can not commit with the current values.
    # Or, if it is valid, returns an empty string value.

    my ($session,
      $platform,
      $projectRecordID,
      $projectRecord,
      $projectName,
    );

    $session = $entity->GetSession();

    # Get the client OS indicated by the user.
    $platform = $entity->GetFieldValue("client_os")->GetValue();

    # Get the project name the user indicated. This information
    # is stored on a referenced, stateless record.
    $projectName = $entity->GetFieldValue("project.name")->GetValue();

    # Check the project name against the OS type. If the
    #  given project is not targeted for that platform,
    #  return a validation error.
    if ($projectName eq "Gemini") {
      if ($platform != "NT") {
        $result = "That project only supports NT.";
      }
    } elsif ($projectName eq "Aquarius") {
      if ($platform != "Unix"){
        $result = "That project only supports Unix.";
      }
    }
    return $result;
}
```

Record Script Example

When you use VBScript, record scripts, field hook, and action hooks are implicitly associated with an Entity object; unless you specifically name another Entity object, all calls to the methods of the Entity class refer to this implicit object. When you use Perl, reference this association with the predefined variable, $entity.

The following example shows a Record script capable of responding to both button clicks and context-menu item selections. When the button is clicked, this hook puts the name of the component's lead engineer in the *component_ref* field, which displays the person assigned to work on the defect.

Note: This example provides a general idea of how you might add a record script to your schema. For readability, the example does 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.

VBScript

```
Function Request_AssignEngineer(param)
    ' param As Variant
    ' This hook responds to changes in the current component and
    ' assigns the request to the lead engineer for that component.
    Dim eventType, componentObj, leadname
    eventType = param.EventType
    If eventType = AD_BUTTON_CLICK Then
      ' Get the lead person for the given component
      leadName = GetFieldValue("component_lead").GetValue
      If leadName = "" Then
        Request_AssignEngineer = "Couldn't get Component Lead value"
        Exit function
      End if
      ' Put that person's name in the Assigned To: field
      SetFieldValue "component_ref", leadName
      Request_AssignEngineer = SetFieldValue "component_ref", leadName
    Elseif eventType = AD_CONTEXMENU_ITEM_SELECTION Then
       SetFieldValue "component_ref", GetSession.GetUserFullname
      Request_AssignEngineer =  SetFieldValue "component_ref",
GetSession.GetUserFullname
    End if
```

```
End Function
```

Global Script Example

The following example is a global script that checks to see if the current user is a member of the specified group. If the user belongs to the group, the hook returns true.

Note: This example provides a general idea of how you might create a global script. For readability, the example does 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.

VBScript

```
Function IsInGroup(groupname)
    ' groupName As String
    ' IsInGroup As Bool
    Set curSession = GetSession
    groupList = curSession.GetUserGroups
    IsInGroup = False
    For Each group in groupList
      If group = groupname Then
        IsInGroup = True
        Exit For
      End If
    Next
End Function
```

Perl

```
sub IsInGroup {
    my ($groupName) = @_;
    my ($curSession,
    $groupList,
    $isInGroup,
    $group,
    );

    $curSession = $entity->GetSession();
    $groupList = $curSession->GetUserGroups();
```

```
    $isInGroup = 0;

    foreach $group (@$groupList) {
      if ($group eq $groupName) {
         $isInGroup = 1;
         last;
      }
    }
    return $isInGroup;
}
```

Using CAL Methods in ClearQuest Hook Scripts Example

This section shows how to include CAL (ClearCase Automation Library) methods in ClearQuest hook scripts to customize the behavior of the UCM-ClearQuest integration. The following example is a modified version of the **UCM_CQActBeforeChact** Visual Basic script, which implements the Perform ClearQuest Action Before Changing Activity policy. When this policy is enabled, the integration executes the script when a developer initiates a Finish Activity operation, either from the GUI or by entering the **cleartool chactivity –cqact** command.

The modified script uses CAL methods to determine whether the developer is working in a single-stream project or a multiple-stream project. If the developer is working in a single-stream project, the script allows the Finish Activity operation. Otherwise, the script returns an error message and cancels the Finish Activity operation.

VBScript

```
REM Start of Global Script UCM_CQActBeforeChact
Function UCM_CQActBeforeChact (entity_type, entity_id, project_info,
stream_info)
' This is the script that implements the "Perform Action Before
' Changing Activity" policy. When initially installed, it invokes a
' default script. If users want to customize this policy, they should
' edit this script to use their code rather than invoking the default
' script. The default script code can be used as an example.
'
' INPUT:
' - entity_type: type of entity on which action will be executed
' - entity_id: id (e.g. "SAMPL0000001") of entity on which action will
```

```
' be executed
' OUTPUT:
' - If the action was successfully executed, this must return an empty
' string
' - If the action was not successfully executed, this must return a
' string to be displayed as an error message.
' Allow chact only if activity is in a single stream project
proj_model = "DEFAULT"
' Get hook's session context
Set session = GetSession()
' Get the entity
Set entity = session.GetEntity(entity_def_name, entity_id)
' Get the entity's ucm_vob_object field value. This value is a string
' that includes the UCM project's object ID and the PVOB's UUID.
ucm_vob_object = entity.GetFieldValue("ucm_vob_object").GetValue()
Dim pvob_uuid
' Strip the project's object ID from the string and return the PVOB's
' UUID.
pvob_uuid = Right(ucm_vob_object, 40)
' Initialize ClearCase.Application COM object
' Create a ClearCase application object. A ClearCase application
' object must exist before you can use CAL methods. The remaining
' steps in the script use CAL methods.
On Error Resume Next
Set CC = CreateObject("ClearCase.Application")
If Err.Number <> 0 Then
MsgBox "ClearCase.Application Error 1:" & Err.Description
End if
On Error Resume Next
' Using the PVOB's UUID, get a ClearCase PVOB object of the activity.
Set PVOB = CC.ProjectVOB(pvob_uuid)
If Err.Number <> 0 Then
MsgBox "ClearCase.Application Error 2:" & Err.Description
End if
On Error Resume Next
' Create a ClearCase activity object (CCActivity) based on the PVOB and
' entity ID (equals UCM activity name).
```

```
Set Act = PVOB.Activity(entity_id)
If Err.Number <> 0 Then
MsgBox "ClearCase.Application Error 3:" & Err.Description
End if
On Error Resume Next
' Return the stream in which the activity was created.
set stream = Act.Stream
If Err.Number <> 0 Then
MsgBox "ClearCase.Application Error 4:" & Err.Description
End if
On Error Resume Next
' Return the project that contains the stream.
set project = stream.Project
If Err.Number <> 0 Then
MsgBox "ClearCase.Application Error 5:" & Err.Description
End if
On Error Resume Next
' Return the project model.
proj_model = project.Model
If Err.Number <> 0 Then
MsgBox "ClearCase.Application Error 6:" & Err.Description
End if
' Test the value of the project model.
' model = SIMPLE: single stream project
' If it is SIMPLE, meaning single-stream, the script returns an
' empty string and the developer is allowed to complete the Finish
' Activity operation.
' model = DEFAULT: hierarchical project
' If it is DEFAULT, meaning multiple-stream, the script returns an
' error message and cancels the Finish Activity operation.
If proj_model = "SIMPLE" Then
' single stream model, allow change act
UCM_CQActBeforeChact = ""
Else
' hierarchical project, fail
UCM_CQActBeforeChact = "Must be in a single stream project."
End if
```

```
End Function
REM End of Global Script UCM_CQActBeforeChact
```

Advanced Reporting and Automation with cqperl

In addition to the command line and batch support provided via the **cqtool** command, ClearQuest for UNIX has full support for external Perl scripting via **cqperl**. **cqperl** must be used for Perl scripting with ClearQuest for UNIX. Other versions of Perl will not function properly.

There are several considerations when using **cqperl** on a UNIX client:

Each Perl script must include code for loading the **CQPerlExt** Perl module. For example:
use CQPerlExt;

See the *API Reference* for Rational ClearQuest for detailed information about the elements of the ClearQuest Perl API.

The following example **cqperl** code generates a report similar to the one described in *Executing Nightly Reports Using cqtool on UNIX* on page 385.

```
# nightlysubmits.pl – A Perl script to list all of the
defects currently in the submit state.
use CQPerlExt;
# All ClearQuest work is done via a session object. Cqperl
# obtains a session object with the CQSession Build method
# accessible from the CQPerlExt Perl module.
#   API Reference: Session Object->Build
$session = CQSession::Build();
# Once we've obtained the session, we need to logon. This is
# done with the UserLogon method. You need to specify the
# username, the password, and the database name. The fourth
# parameter, dbset, is usually left blank.
#   API Reference: Session object->UserLogon method
$session->UserLogon("admin","","SAMPL","");
# Generating a query involves creation of a QueryDef object.
# This is done via a method of the session object called
# BuildQuery. It's only parameter is the entitydef
# (also known as Record Type) that you wish to query on.
# In this case, we'll use "Defect"
```

```
#   API Reference: Session Object->Build Query method
#                  QueryDef Object
#                  EntityDef Object->Name property
$querydef = $session->BuildQuery("Defect");
# The next step (like creating a query through the ClearQuest
# UNIX GUI) is to decide which fields will be in the Query
# Result Set. This is done with the BuildField method of
# the QueryDef object. We'd like to see ID, headline, and
# submitter.
#   API Reference: QueryDef Object->BuildField method
$querydef->BuildField("id");
$querydef->BuildField("headline");
$querydef->BuildField("submitter");
# Next, we need to build the filters for this query.
# This is done by constructing a tree of FilterOperator
# objects. Creating the top level FilterOperator object for
# any subtree is done with the BuildFilterOperator method
# of the QueryDef object. The BuildFilterOperator method
# takes one parameter, the boolean operator that will
# determine how each of the subtrees behaves. If there is
# only one filter, either AND or OR will work. To specify
# the correct boolean operator, select the proper BoolOp
# constant and Perl prefix. In this case, we'll use and, so
# therefore, our constant will be $CQPerlExt::CQ_BOOL_OP_AND.
#   API Reference: QueryDef Object->BuildFilterOperator Method
#                  BoolOp constants
#                  Notation conventions for Perl
$rootfilternode =
   $querydef->BuildFilterOperator($CQPerlExt::CQ_BOOL_OP_AND);
# Once we have the root FilterOperatorNode, we'll assign a
# filter to it. In this case, state equals submitted. We'll
# use the BuildFilter method of the QueryFilterNode object
# for this. Note that the third parameter to BuildFilter must
# be a Perl reference to an array.
#   API Reference: QueryFilterNode object->BuildFilter method
#                  BoolOp constants
#                  Notation conventions for Perl
```

```
@statetest = "Submitted";
$rootfilternode->BuildFilter("State",
                             $CQPerlExt::CQ_COMP_OP_EQ,
                             \@statetest);
# Okay, the Query definition has been created, now it's time
# to execute it. We go back to the session object for this
# and use the BuildResultSet method. It's only parameter
# is the QueryDef object we'd previously created. After
# the result set object is ready, we then execute the query.
#   API Reference: Session object->BuildResultSet method
#                  ResultSet object->Execute method
$resultset = $session->BuildResultSet($querydef);
$resultset->Execute();
# Let's prepare by printing a header for our output.
printf("%13.13s %50.50s %9.9s\n","id","headline","submitter");
printf("%13.13s %50.50s %9.9s\n",
       "-------------",
       "--------------------------------------------------",
       "---------");
# Now, traverse the resultset and print out the output.
# This is done via the MoveNext method of the result set
# object. It will return $CQPerlExt::CQ_SUCCESS as long as
# there are rows to view. GetColumnValue is used to get the
# data from that row of the resultset.
#   API Reference: ResultSet object->MoveNext method
#                  ResultSet object->GetColumnValue method
while ($resultset->MoveNext() == $CQPerlExt::CQ_SUCCESS) {
    printf("%-13.13s %-50.50s %-9.9s\n",
           $resultset->GetColumnValue(1),
           $resultset->GetColumnValue(2),
           $resultset->GetColumnValue(3));
# And we're done, so let's release the session
CQSession::Unbuild($session);
```

Form Controls

D

This section describes each of the Rational ClearQuest form controls:

- *ActiveX Control* on page 349.
- *Attachment Control* on page 350.
- *Check-Box Control* on page 351.
- *Combo-Box Control* on page 352.
- *Drop-Down List Box Control* on page 353.
- *Drop-Down Combo Box Control* on page 355.
- *Duplicate Base Control* on page 356.
- *Duplicate Dependents Control* on page 357.
- *Group Box Control* on page 358.
- *History Control* on page 358.
- *List Box Control* on page 359.
- *List View Control* on page 360.
- *Option Button Control* on page 361.
- *Parent/Child Control* on page 362.
- *Picture Control* on page 364.
- *Push Button Control* on page 364.
- *Static Text Box Control* on page 366.
- *Text Box Control* on page 366.

For information about how to add controls to record forms, see Chapter 7, *Working with Forms*.

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: **General** and **ActiveX Control**.

General Tab

The following table lists the properties of the **General** tab.

Property	Description
Field Name	Select the field to associate with the control.
Label	Enter a label for the control. By default, ClearQuest sets the label to the name of the selected field.
X	The horizontal starting point of the control in pixels.
Y	The vertical starting point of the control in pixels.
Width	The width of the control in pixels.
Height	The height of the control in pixels.

ActiveX Control Tab

The following table lists the properties of the **ActiveX Control** tab.

Property	Description
Registered Name	Type the registered name of the ActiveX control you want to use.
Initialization Record	Select a record script from the drop-down list. The record script is used to initialize the control, and fires when the control's form opens.
Action Record Script	Select a record script from the drop-down list. The record script is called by the ClearQuest run time whenever the user initiates an action, such as submit or modify, or when the action is completed or reverted.

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

The following table lists the properties of the **General** tab.

Property	Description
Field Name	Select the field to associate with the control.
Label	Enter a label for the control. By default, ClearQuest sets the label to the name of the selected field.
X	The horizontal starting point of the control in pixels.
Y	The vertical starting point of the control in pixels.
Width	The width of the control in pixels.
Height	The height of the control in pixels.

Check-Box Control

A check-box control is a two-value control that you can use for Boolean values or any field that has only two 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 two strings for the checked and unchecked values of the control.

You should attach a default-value hook to the field for 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.

General Tab

The following table lists the properties of the **General** tab.

Property	Description
Field Name	Select the field to associate with the control.
Label	Enter a label for the control. By default, ClearQuest sets the label to the name of the selected field.
X	The horizontal starting point of the control in pixels.
Y	The vertical starting point of the control in pixels.

Property	Description
Width	The width of the control in pixels.
Height	The height of the control in pixels.

Extended Tab

The following table lists the properties of the **Extended** tab.

Property	Description
Check Value	Enter the text to display when there is a mark in the check box.
Uncheck Value	Enter the text to display when there is no mark in the check box.

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 only assign multiline and reference list 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: **General**, **Extended**, and **Context Menu Hooks**.

General Tab

The following table lists the properties of the **General** tab.

Property	Description
Field Name	Select the field to associate with the control.

Property	Description
Label	Enter a label for the control. By default, ClearQuest sets the label to the name of the selected field.
X	The horizontal starting point of the control in pixels.
Y	The vertical starting point of the control in pixels.
Width	The width of the control in pixels.
Height	The height of the control in pixels.

Extended Tab

The **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

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 in the ClearQuest client application. To run 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 only assign multiline and reference list 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 four tabs: **General**, **Extended**, **Context Menu Hooks**, and **Web Dependent Fields**.

General Tab

The following table lists the properties of the **General** tab.

Property	Description
Field Name	Select the field to associate with the control.
Label	Enter a label for the control. By default, ClearQuest sets the label to the name of the selected field.
X	The horizontal starting point of the control in pixels.
Y	The vertical starting point of the control in pixels.
Width	The width of the control in pixels.
Height	The height of the control in pixels.

Extended Tab

The **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

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 in the ClearQuest client application. To run one of these hooks, the user must right-click the control and select the name of the hook.

Web Dependent Fields Tab

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 four tabs: **General**, **Extended**, **Context Menu Hooks**, and **Web Dependent Fields**.

General Tab

The following table lists the properties of the **General** tab.

Property	Description
Field Name	Select the field to associate with the control.
Label	Enter a label for the control. By default, ClearQuest sets the label to the name of the selected field.
X	The horizontal starting point of the control in pixels.
Y	The vertical starting point of the control in pixels.
Width	The width of the control in pixels.
Height	The height of the control in pixels.

Extended Tab

The **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

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 in the ClearQuest client application. To run one of these hooks, the user must right-click the control and select the name of the hook.

Web Dependent Fields Tab

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.

The property sheet for a duplicate base control has one tab: **General**.

General Tab

The following tables lists the properties for the **General** tab.

Property	Description
Field Name	Select the field to associate with the control.
Label	Enter a label for the control. By default, ClearQuest sets the label to the name of the selected field.
X	The horizontal starting point of the control in pixels.
Y	The vertical starting point of the control in pixels.
Width	The width of the control in pixels.
Height	The height of the control in pixels.

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.

The property sheet for a duplicate dependents control has one tab: **General**.

General Tab

The following table lists the properties for the **General** tab.

Property	Description
Field Name	Select the field to associate with the control.
Label	Enter a label for the control. By default, ClearQuest sets the label to the name of the selected field.
X	The horizontal starting point of the control in pixels.
Y	The vertical starting point of the control in pixels.
Width	The width of the control in pixels.

Property	Description
Height	The height of the control in pixels.

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.

The property sheet for a group box control has one tab: **General**.

General Tab

The following table lists the properties of the **General** tab.

Property	Description
Field Name	Select the field to associate with the control.
Label	Enter a label for the control. By default, ClearQuest sets the label to the name of the selected field.
X	The horizontal starting point of the control in pixels.
Y	The vertical starting point of the control in pixels.
Width	The width of the control in pixels.
Height	The height of the control in pixels.

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.

The property sheet for a history control has one tab: **General**.

General Tab

The following table lists the properties of the **General** tab.

Property	Description
Field Name	Select the field to associate with the control.
Label	Enter a label for the control. By default, ClearQuest sets the label to the name of the selected field.
X	The horizontal starting point of the control in pixels.
Y	The vertical starting point of the control in pixels.
Width	The width of the control in pixels.
Height	The height of the control in pixels.

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: **General**, **Extended**, and **Context Menu Hooks**.

General Tab

The following table lists the properties of the **General** tab.

Property	Description
Field Name	Select the field to associate with the control.
Label	Enter a label for the control. By default, ClearQuest sets the label to the name of the selected field.
X	The horizontal starting point of the control in pixels.

Property	Description
Y	The vertical starting point of the control in pixels.
Width	The width of the control in pixels.
Height	The height of the control in pixels.

Extended Tab

The **Extended** tab contains a single check box for Auto Sort. If **Auto Sort** is checked, entries in the list portion of the list box are sorted alphabetically. If the **Auto Sort** check box is cleared, entries are listed in the order in which they were added.

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 **Add**. To disassociate a hook, select the hook in the **Selected** column and click **Remove**. Hooks associated with this control are displayed in the control's shortcut menu in the ClearQuest client application. To run 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 lets you 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: **General**, **Extended**, and **Context Menu Hooks**.

General Tab

The following table lists the properties of the **General** tab.

Property	Description
Field Name	Select the field to associate with the control.
Label	Enter a label for the control. By default, ClearQuest sets the label to the name of the selected field.
X	The horizontal starting point of the control in pixels.
Y	The vertical starting point of the control in pixels.
Width	The width of the control in pixels.
Height	The height of the control in pixels.

Extended Tab

The **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

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 **Add** to disassociate a hook, select the hook in the **Selected** column and click **Remove**. Hooks associated with this control are displayed in the control's shortcut menu in the ClearQuest client. To run 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 can assign a different field to each option button in the group.

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

General Tab

The following table lists the properties of the **General** tab.

Property	Description
Field Name	Select the field to associate with the control.
Label	Enter a label for the control. By default, ClearQuest sets the label to the name of the selected field.
X	The horizontal starting point of the control in pixels.
Y	The vertical starting point of the control in pixels.
Width	The width of the control in pixels.
Height	The height of the control in pixels.

The following table lists the properties of the **Extended** tab.

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

Use a parent/child control 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.

A parent/child control consists of 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 parent/child control has three tabs: **General**, **Extended**, and **Context Menu Hooks.**

General Tab

The following table lists the properties of the **General** tab.

Property	Description
Field Name	Select the field to associate with the control.
Label	Enter a label for the control. By default, ClearQuest sets the label to the name of the selected field.
X	The horizontal starting point of the control in pixels.
Y	The vertical starting point of the control in pixels.
Width	The width of the control in pixels.
Height	The height of the control in pixels.

Extended Tab

The **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

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 **Add**. To disassociate a hook, select the

hook in the **Selected** column and click **Remove**. Hooks associated with this control are displayed in the control's shortcut menu in the ClearQuest client. To run 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 following table lists the properties of a picture control.

Property	Description
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.
X	The horizontal position of the upper-left corner of the picture in pixels.
Y	The vertical position of the upper-left corner of the picture in pixels.
Width	The width of the picture in pixels.
Height	The height of the picture in pixels.

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 runs 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.

The property sheet for a push button control has two tabs: **General** and **Extended**.

General Tab

The following table lists the properties of the **General** tab.

Property	Description
Field Name	Select the field to associate with the control.
Label	Enter a label for the control. By default, ClearQuest sets the label to the name of the selected field.
X	The horizontal starting point of the control in pixels.
Y	The vertical starting point of the control in pixels.
Width	The width of the control in pixels.
Height	The height of the control in pixels.

Extended Tab

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

Property	Description
Click Hook	Select a record script to run when the button is pressed.
Enable for web	Check Click Hook to enable the button for the web.
Associated Component	Select a list view ID to associate with the button.
Add	Select to add items to a list view.
Remove	Select to remove items from a list view.
New	Select to add a new record to the list view.
Other	Select to associate a record script with the list view.

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.

The property sheet for a static text box control has one tab: **General**.

General Tab

The following table lists the properties of the **General** tab.

Property	Description
Field Name	Select the field to associate with the control.
Label	Enter a label for the control. By default, ClearQuest sets the label to the name of the selected field.
X	The horizontal starting point of the control in pixels.
Y	The vertical starting point of the control in pixels.
Width	The width of the control in pixels.
Height	The height of the control in pixels.

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: **General**, **Extended**, **Date/Time**, and **Context Menu Hooks**.

General Tab

The following table lists the properties of the **General** tab.

Property	Description
Field Name	Select the field to associate with the control.
Label	Enter a label for the control. By default, ClearQuest sets the label to the name of the selected field.
X	The horizontal starting point of the control in pixels.
Y	The vertical starting point of the control in pixels.
Width	The width of the control in pixels.
Height	The height of the control in pixels.

Extended Tab

The following table lists the properties of the **Extended** tab.

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.
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 following table lists the properties of the **Date/Time** tab.

Property	Description
Date	If checked, this control enables the Date Formats list, allowing you to chose 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** check box, 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 **Add**. To disassociate a hook, select the hook in the **Selected** column and click **Remove**. Hooks associated with this control are displayed in the control's shortcut menu. To run one of these hooks, the user must right-click the control and select the name of the hook.

Command Line Utilities

E

Rational ClearQuest includes a number of utilities that can be initiated from the command line. These command line utilities are powerful tools for performing certain types of administrative tasks.

The topics covered include:

- *The dbset Parameter in Command Line Utilities* on page 369.
- *Using installutil* on page 369.
- *Copying a Schema with cqload* on page 376.
- *Importing and Exporting Dynamic Lists with importutil* on page 381.
- *Moving Workspace Items Between User Databases with bkt_tool* on page 382.
- *Executing Nightly Reports Using cqtool on UNIX* on page 385.

The dbset Parameter in Command Line Utilities

The **–dbset** parameter is optional with many ClearQuest command line utilities. If you have multiple schema repositories in your ClearQuest system, you must specify **–dbset** and the connection name to run the command. If **–dbset** is omitted, the command line tool will use the default dbset name, which is the product version number (**2003.06.00**, for example).

Using installutil

The **installutil** command line utility includes a number of subcommands that can be useful when setting up or modifying databases. This section describes a few of the more common **installutil** subcommands.

Options and arguments to many **installutil** subcommands are specific to a particular database vendor, as listed in Table 15. All parameters to **installutil** subcommands are required unless stated otherwise in Table 15, and must be entered in the order specified. Options that must be entered with a null value are shown (and must be entered) as **“”**.

Table 15 Sample Data by Vendor Type

Parameter	Access	SQL Anywhere	MS SQL	Oracle	DB2
db_vendor	MS_ ACCESS	SQL_ ANYWHERE	SQL_ SERVER	ORACLE	DB2
server	""	Database service name	Database server host name	SQL*Net alias	""
database	Full path to the schema repository or user database	Full path to the schema repository or user database	Schema repository or user database name	Schema repository or user database name	Schema repository or user database alias
dbo_login	**admin**	**admin**	Database owner user name	Database owner user name	Database owner user name
dbo_password	**admin**	**admin**	Database owner password	Database owner password	Database owner password
rw_login	**admin**	**admin**	Database owner user name	Database owner user name	Database owner user name
rw_password	**admin**	**admin**	Database owner password	Database owner password	Database owner password
ro_login	**admin**	**admin**	Database owner user name	Database owner user name	Database owner user name
ro_password	**admin**	**admin**	Database owner password	Database owner password	Database owner password

Table 15 Sample Data by Vendor Type

Parameter	Access	SQL Anywhere	MS SQL	Oracle	DB2
connect_ options	""	SERVER_ VER=*server_ version_ number*	""	HOST=*host*; SID=*sid*; SERVER_ VER=*server_ version_ number*; CLIENT_ VER=*client_ version_ number*; LOB_ TYPE=*data_ type*	""

Creating Database Copies

To copy a schema repository and create a sample database for testing the upgrade process, run **installutil convertschemarepo**. This subcommand does the following:

- Copies the original database to a new database.
- Connects ClearQuest to the database copy.
- Locks the original database.

Note: Before running this command, use your database vendor tools to back up your databases and create new ones.

To create a copy of your databases so you can test the upgrade process offline:

1. Create new test databases. See the *Installation Guide* for Rational Server Products.
2. Use **installutil convertschemarepo** to copy your existing schema repository and connect it to ClearQuest. See *installutil convertschemarepo* on page 371 for sample input parameters by vendor type.
3. Use **installutil convertuserdb** to copy each of your existing user databases and connect them to ClearQuest. See *installutil convertuserdb* on page 372 for sample input parameters by vendor type.

installutil convertschemarepo

installutil convertschemarepo [**–dbset** *dbset_name*] *cq_login cq_password to_db_vendor to_server to_database to_dbo_login to_dbo_password to_rw_login to_rw_password to_ro_login to_ro_password connect_options* [*tcpip,ipx,netbios,namedpipes*] [*host1,host2,...*]

Parameter	Description
dbset_name	The name of the existing connection.
cq_login	The user login name for ClearQuest databases that you want to copy. The user must have Super User privileges.
cq_password	The login password for the ClearQuest Super User.
to_db_vendor	The new upgrade vendor database name. For vendor-specific values, see the *database* entry in Table 15 on page 370.
to_server	The new upgrade database server name.
to_database	The new upgrade database name.
to_dbo_login	The login name of the empty database owner.
to_dbo_password	The login password of the empty database owner.
to_rw_login	The read/write login name for the empty database.
to_rw_password	The read/write login password for the empty database.
to_ro_login	The read-only login name for the empty database.
to_ro_password	The read-only login password for the empty database.
connect_options	For vendor-specific values, see the *connect_options* entry in Table 15 on page 370.
[*tcpip*,*ipx*,*netbios*, *namedpipes*]	These are optional parameters to specify networks.
[*host1*,*host2*,...]	These are optional parameters to identify multiple host machines.

installutil convertuserdb

installutil convertuserdb [**–dbset** *dbset_name*] *cq_login cq_password user_dbname to_user_db_vendor to_user_server to_user_database to_user_dbo_login to_user_dbo_password to_user_rw_login to_user_rw_password connect_options* [*tcpip*,*ipx*,*netbios*,*namedpipes*] [*host1*,*host2*,...]

Parameter	Description
dbset_name	The name of the existing connection.

Parameter	Description
cq_login	The user login name to ClearQuest databases that you want to copy. The user must have Super User privileges.
cq_password	The login password for the ClearQuest Super User.
user_dbname	The name of the user database that you want to convert.
to_user_db_vendor	The new upgrade vendor database name. For vendor-specific values, see the *database* entry in Table 15 on page 370.
to_user_server	The new upgrade user database server name.
to_user_database	The new upgrade user database name.
to_user_dbo_login	The login name of the owner of the empty user database.
to_user_dbo_password	The login password of the owner of the empty user database.
to_user_rw_login	The read/write login name for the empty user database.
to_user_rw_password	The read/write login password for the empty user database.
connect_options	For vendor-specific values, see the *connect_options* entry in Table 15 on page 370.
[*tcpip*,*ipx*,*netbios*, *namedpipes*]	These are optional parameters to specify networks.
[*host1*,*host2*,...]	These are optional parameters to identify multiple host machines.

Unlocking Databases

When you create database copies using **installutil**, the **convertschemarepo** subcommand locks the production database before copying it. To allow users to continue using the production database while you test the copy, you must unlock it.

Note: After you complete the upgrade, do not unlock the original databases. Keeping the original databases locked prevents users from accessing the wrong databases.

- To unlock the schema repository, enter the following **installutil** command, referencing the database copy with the *database* parameter:

  installutil unlockschemarepo [**–dbset** *dbset_name*] *db_vendor server database dbo_login dbo_password connect_options*

- To unlock the user database, enter the following **installutil** command, referencing the original database (not the schema repository) in the *database* parameter:

  installutil unlockuserdb [**–dbset** *dbset_name*] *db_vendor server database dbo_login dbo_password connect_options*

For vendor-specific options and arguments, see Table 15 on page 370.

Examples

This section provides example **installutil** command lines for creating copies of a schema repository and a user database, then unlocking the originals. Commands like these are typical in procedures that create a copy of a schema repository and user database for testing purposes while the original schema repository and user database remain in use. Example database vendors include Oracle and SQL Server.

Oracle

This example assumes the following:

- An existing schema repository in a DBSet that has the default name (so the **–dbset** option need not be specified.)
- An existing ClearQuest user database named **MyUserDB**. The database owner for this database has Oracle userid **udbo** and password **P**.
- An existing ClearQuest schema repository named **MySR**. The database owner for this schema repository has Oracle userid **srdbo** and password **P**.
- Default ClearQuest superuser user id and password of **admin** and “” for all databases and schema repositories.
- The Oracle TNS name **MYTNS** points to the Oracle server to which the databases will be copied.
- The Oracle userids **srusr** and **dbusr**, each with password **P**, exist on the target Oracle server. These users have been granted CONNECT and RESOURCE roles.
- The Oracle server's hostname is **DBServ** and its SID (Oracle system ID) is **ORC**.
- You are using Oracle 8.1 on the Oracle database server and the client workstation.

To copy the schema repository:

```
installutil convertschemarepo admin "" ORACLE MYTNS srusr srusr P srusr P srusr P\
"HOST=DBServ;SID=ORC;CLIENT_VER=8.1;SERVER_VER=8.1;LOB_TYPE=LONG"
```

To copy the user database:

```
installutil convertuserdb admin "" ORACLE MYDB MYTNS dbusr dbusr P dbusr P \
"HOST=DBServ;SID=ORC;CLIENT_VER=8.1;SERVER_VER=8.1;LOB_TYPE=LONG"
```

To unlock the originals after the copy is complete:

installutil unlockschemarepo ORACLE MYTNS MySR srdbo P\ "HOST=DBServ;SID=ORC;CLIENT_VER=8.1;SERVER_VER=8.1;LOB_TYPE=LONG"

installutil unlockuserdb ORACLE MYTNS MyUserDB udbo P\ "HOST=DBServ;SID=ORC;CLIENT_VER=8.1;SERVER_VER=8.1;LOB_TYPE=LONG"

SQL Server

This example assumes the following:

- An existing schema repository in a DBSet that has the default name (so the **–dbset** option need not be specified).
- An existing ClearQuest user database named **MYDB**.
- Default ClearQuest superuser user id and password of **admin** and “”.
- The target SQL Server host name is **MySQLServer**.
- The name of the SQL Server database that will hold the new copy of the schema repository is **MyNewSchemaRepo**. It is owned by SQL Server userid **dbowner** whose password is **P**.
- The name of the SQL Server database that will hold the new copy of the user database is **MyNewUserDB**. It is also owned by SQL Server userid **dbowner**.

To copy the schema repository:

installutil convertschemarepo admin "" SQL_SERVER ^ MySQLServer MyNewSchemaRepo dbowner P dbowner P dbowner P ""

To copy the user database:

installutil convertuserdb admin "" MYDB SQL_SERVER ^ MySQLServer MyNewUserDB dbowner P dbowner P ""

To unlock the originals after the copy is complete:

installutil unlockschemarepo admin "" MYDB SQL_SERVER ^ MySQLServer MyNewSchemaRepo dbowner P ""

installutil unlockuserdb admin "" MYDB SQL_SERVER ^ MySQLServer MyNewUserDB dbowner P ""

Copying a Schema with cqload

You can copy an entire schema into a schema repository or copy a partial schema into another schema by using the **cqload** command line utility.

Preparing to Use cqload

Make sure that the schema to which you want to apply **cqload** is checked into the schema repository. If the schema is checked out, check it in.

Note: If you run **cqload** while ClearQuest Designer is running, you must exit the Designer and then log in again to see your changes.

Copying a Schema into a Schema Repository

1. Export the schema (all revisions) you want to copy using the command **exportschema**. See *exportschema* on page 376.
2. Import the copied schema into the schema repository using the command **importschema**. See *importschema* on page 377.

Copying a Partial Schema into an Existing Schema

1. Export part of a schema (one or more revisions) using the command **exportintegration**. See *exportintegration* on page 378.
2. Import the copied schema into an existing schema using the command **importintegration**. See *importintegration* on page 379.

exportschema

Use the **exportschema** subcommand, part of the **cqload** command line utility, to export an entire schema to a text file. You can then use the **importschema** subcommand to import the schema into another schema repository.

To export a partial schema, use the **exportintegration** subcommand.

Syntax

cqload exportschema [**–dbset** *name*] *login password schema_name* **"***schema pathname***"**

Where	Represents
–dbset *name*	Specifies the schema connection name. Required if there are multiple schema repositories in the system.

Where	Represents
login	The ClearQuest login name of the user. This user must have Super User privileges.
password	The ClearQuest password for the user. If there is no password, enter an empty set of double quotes.
schema_name	The name of the schema in your schema repository that is to be exported to a text file. For example, TeamTest.
schema_pathname	The text file version of a schema to be saved. Enclose the pathname in double quotes.

Example

cqload exportschema –dbset 2003.06.00 admin "" DefectTracking "c:\schema.txt"

The preceding example exports the contents of the DefectTracking schema to the file c:\schema.txt.

importschema

Use the **importschema** subcommand, part of the **cqload** command line utility, to import an entire schema from a text file and add it to your schema repository. This subcommand is useful for sharing schemas with sites that can not access your schema repository or that use a different schema repository.

Before using **importschema**, you must export the schema using the **exportschema** command. To import a partial schema, use the **importintegration** subcommand.

Syntax

cqload importschema [**–dbset** *name*] *login password* **"***schema_pathname***"**

Where	Represents
–dbset *name*	Specifies the schema connection name. Required if there are multiple schema repositories in the system.
login	The ClearQuest login name of the user. This user must have Super User privileges.
password	The ClearQuest password for the user. If there is no password, enter an empty set of double quotes.

Where	Represents
schema_pathname	The pathname to the file that contains the textual representation of a schema that has been saved by the exportschema subcommand. Enclose the pathname in double quotes.

Example

cqload importschema –dbset 2003.06.00 admin "" "c:\schema.txt"

The preceding example imports a schema (exported in text form and saved in c:\schema.txt) into the current schema repository.

Note: C:\schema.txt was created using the **cqload exportschema** command. During that process, the name of the exported schema was saved in this file. When you import this schema, that schema name will be used to create the schema with **cqload importschema**. If that name is already in use in the schema repository, the import will fail.

exportintegration

Use **cqload exportintegration** to export specific versions of a schema in a form that can be used to modify an existing schema. To import the data into another schema, use the **importintegration** subcommand. The exporting and importing schemas should use similar record types.

Syntax

cqload exportintegration [**–dbset** *name*] *login password schema_name begin_rev end_rev* [*record_type_to_rename*] **"***schema_pathname***"**

Where	Represents
–dbset *name*	Specifies the schema connection name. Required if there are multiple schema repositories in the system.
login	The ClearQuest login name of the user. This user must have Super User privileges.
password	The ClearQuest password for the user. If there is no password, enter an empty set of double quotes.
schema_name	The name of the schema associated with the integration. The schema name is case sensitive.

Where	Represents
begin_rev	First schema revision whose changes you want to save.
end_rev	Last schema revision whose changes you want to save.
record_type_to_rename	Optional name of a record type in the schema that will be renamed on import by **cqload importintegration**. If this option is omitted, you must supply a null option **“”** in its place.
schema_pathname	Pathname of the file that will contain the results of exporting the schema revisions. Enclose the pathname in double quotes.

Example

cqload exportintegration –dbset 2003.06.00 admin “” Enterprise 5 5 “” ^ “c:\tmp\export.txt”

The preceding example exports only changes made in version five of the **Enterprise** schema.

cqload exportintegration –dbset 2003.06.00 admin ‘’ Enterprise 5 8 “”^ “c:\tmp\export.txt”

The preceding example exports changes made in versions five through eight of the **Enterprise** schema.

cqload exportintegration –dbset 2003.06.00 admin “” Enterprise 5 5 ^ ChangeRequest “c:\temp\scriptchanges.txt”

The preceding example exports only changes made in version five of the **Enterprise** schema and specifies that the record type **ChangeRequest** is to be renamed on import. You must specify a new name for the record in the *new_record_type_name* option of the subsequent **cqload importintegration**.

importintegration

Use **cqload importintegration** subcommand to import schema modifications exported with **cqload exportintegration**.

Note: If ClearQuest MultiSite is in use, **importintegration** can only be performed at the working master site.

Syntax

cqload importintegration [**–dbset** *name*] *login password schema_name* [*new_record_type_name*] *"integration name" integration_version "schema_pathname" "form_name"*

Where	Represents
–dbset *name*	Specifies the schema connection name. Required if there are multiple schema repositories in the system.
login	The ClearQuest login name of the user. This user must have Super User privileges.
password	The ClearQuest password for the user. If there is no password, enter an empty set of double quotes.
schema_name	The name of the schema associated with the integration. This name is the simple name of a schema that is defined in your schema repository (For example, TeamTest). This schema will increase by one revision number after running **cqload**.
new_record_type_name	Optional. The name to which the record type specified in the *record_type_to_rename* option of a **cqload exportintegration** command will be renamed when imported into the new schema. If this option is omitted, you must supply a null option "" in its place.
integration_name	The name you give the integration. It can be any alphanumeric indicator of the integration you are loading.
integration_version	The version you are loading. It should be numeric.
schema_pathname	The pathname of a file that has been produced by **exportintegration**. Enclose the pathname in double quotes.
form_name	The forms to which the new tabs (created by the integration) will be added. If no form updates, type "" to indicate no form update.

Example

cqload importintegration –dbset 1.1 admin "" Testit "" Email_Integ 1 ^ "c:\tmp\export.txt" ""

The preceding example imports the exported schema into the **Testit** schema, leaving all record types with their original names.

cqload importintegration –dbset 1.1 admin "" Testit Defect Email_Integ 1 ^ "c:\tmp\export.txt" ""

The preceding example imports the exported schema into the **Testit** schema, renaming the record type specified in the *record_type_to_rename* option of the previous **cqload exportintegration** command to **Defect**.

Importing and Exporting Dynamic Lists with importutil

You can use the **importutil** command line utility to import or export a dynamic list from text files. This can by done by applying the following **importutil** subcommands:

- **importlist** imports a list to a user database.
- **exportlist** exports a list from a user database.

The exported dynamic list is a text file. If necessary, you can use a text editor to modify its content.

Exporting the Dynamic List

Use the **exportlist** subcommand to export a dynamic list to a text file.

Syntax

importutil exportlist [**–dbset** *name*] *login password dbname listname* **"***output_file_name***"**

Where	Represents
–dbset *name*	Specifies the schema connection name. Required if there are multiple schema repositories in the system.
login	The ClearQuest login name of the user. This user must have Super User privileges.
password	The ClearQuest password for the user. If there is no password, enter an empty set of double quotes.
db_name	The logical name of the user database.
list_name	The name of the dynamic list for the user database *db_name*.
output_file_name	The path and name of the exported file. Enclose the pathname in double quotes.

Example

importutil exportlist –dbset 2003.06.00 "admin" "" PUSER domains ^ "C:\temp\dynalist.txt"

The preceding example exports the contents of the domains dynamic list from the database PUSER and stores them in the file C:\temp\dynalist.txt.

Importing the Dynamic List

Use the **importlist** subcommand to import a dynamic list to a user database. The dynamic list import will overwrite the content of the existing dynamic list with the same name.

Syntax

importutil importlist [**–dbset** *name*] *login password dbname listname* **"***output_file_name***"**

Where	Represents
–dbset *name*	Specifies the schema connection name. Required if there are multiple schema repositories in the system.
login	The ClearQuest login name of the user. This user must have Super User privileges.
password	The ClearQuest password for the user. If there is no password, enter an empty set of double quotes.
db_name	The logical name of the user database.
list_name	The name of the dynamic list for the user database *db_name*.
input_file_name	The path and name of the imported file. Enclose the path name in double quotes.

Example

importutil importlist –dbset 2003.06.00 "admin" "" PUSER domains ^ "C:\temp\dynalist.txt"

The preceding example imports the contents of the file C:\temp\dynalist.txt and uses it to create a dynamic list named domains in the database PUSER.

Moving Workspace Items Between User Databases with bkt_tool

Use **bkt_tool** to copy and move all the contents of a Public Folder, available in the ClearQuest Client Workspace, from one user database to another. This utility is helpful when you are developing a new schema based either on the Blank or Common

predefined schema and want to work with queries, charts, and reports without having to create them all.

This utility is also useful when you are working with test databases. For example, you may want to test queries, charts, and reports in the test database before introducing schema changes in the production database. In this case, you can use **bkt_tool** to copy and move all the production workspace items to the test database.

The subcommands available with **bkt_tool** let you:

- Export all Workspace contents from the Public Folder, available in the ClearQuest Client, to a file.
- Import all the Workspace contents into another user database.
- Update the Workspace contents for a given user database.
- Delete Workspace contents for a user database.

bkt_tool Command Line Options

All **bkt_tool** commands have similar syntax and command line options.

bkt_tool *option* **–dbset** *dbset* **–us "***username***" –password** *password* **–db "***dbname***"** {**–directory "***pathname***"** | **–storagefile "***pathname***"**}

Where	Represents
option	one of **–i** (import), **–u** (update), **–e** (export), **–d** (delete), **–t** (translate)
dbset	Specifies the schema connection name. Required if there are multiple schema repositories in the system.
username	The ClearQuest login name of the user. This user must have Super User privileges.
password	The ClearQuest password for the user. If there is no password, enter an empty set of double quotes.
dbname	The logical name of the user database from which you are exporting the Workspace contents.

Where	Represents
pathname	The path name of the exported file. Enclose the path name in double quotes. If **–directory** is specified then the *pathname* containing the various files comprising the load must exist if the *option* specified is **–i** or **–u**. The *directory* will contain a text file named CONTENTS.CFG and a series of files 0001.DAT through *nnnn*.DAT that contain the binary data for each of the Workspace items. If **–storagefile** is specified then the *pathname* is the path name of an IStorage compound document.

Export

Use the **–e** subcommand to export all the Workspace contents of a Public Folder, available in the ClearQuest Client, from one user database to another.

Syntax

bkt_tool –e –dbset *dbset* **–us "***username***" –password** *password* **–db "***dbname***" {–directory "***pathname***" | –directory "***pathname***"}**

Example

bkt_tool –e –dbset 1.1 –us "admin" –password "" –db "CLSIC" –directory ^ "d:\temp\bkt_test"

The preceding example exports the Workspace contents of the Public Folder from the CLSIC user database to the directory d:\temp\bkt_test.

Import

Use the **–i** subcommand to import the exported file containing Workspace items, into a user database. If the queries, charts and reports already exist in the user database, the **bkt_tool** command line utility displays an error and does not allow you to continue with the import.

Syntax

bkt_tool –i –dbset *name* **–us "***username***" –password** *password* **–db "***dbname***" –directory "***pathname***"**

Example

bkt_tool –i –dbset 1.1 –us "admin" –password "" –db "test1" –directory ^ "d:\temp\bkt_test"

The preceding example imports the contents of d:\temp\bkt_test to the **test1** user database.

Update

Use the **–u** subcommand to update existing Workspace items in the ClearQuest client for the user database. If the queries, charts and reports already exist in the user database, the **bkt_tool** command line utility overwrites the existing items with the latest data.

Syntax

bkt_tool –u –dbset *dbset* **–us "***username***" –password** *password* **–db "***dbname***" –directory "***pathname***"**

Example

bkt_tool –u –dbset 1.1 –us "admin" –password "" –db "test1" –directory "d:\temp\bkt_test"

The preceding example updates the **test1** user database from the directory d:\temp\bkt_test.

Executing Nightly Reports Using cqtool on UNIX

A common question concerns how to automate running various reports overnight. This is almost always coupled with some amount of e-mail notification. This is an example of using **cqtool**, the ClearQuest command line interface on UNIX, to dynamically create and execute an ad-hoc query that will display the defects that are in the submitted state. More detailed information on this example can be obtained by running:

man cqtool or cqtool new_query –man

There are three essential elements of **cqtool** use:

- Logging in to the database.
- Assembling a set of commands to execute.
- Determining the output.

All examples use the sample user database.

Logging in to the sample database is typically done with a database name of **SAMPL**, a user id of admin, and a blank password. Executing **cqtool login** will start a command line shell that lets you work interactively with ClearQuest on UNIX from the command line:

cqtool login –database SAMPL –user admin –password “”

To execute other commands, but in batch mode, replace **login** with the command to be executed. In the case of this example, we want **cqtool** to create a new query and execute it. This is done with the **new_query** command. The **new_query** command takes a number of parameters for field display and filter operations. Fields can be displayed with the **–field** *fieldid* option and filters are executed with *–filterop* variable value. This example displays the id, headline, and submitter fields for all defects that are in the submitted state.

cqtool new_query –type defect –field id –field headline –field submitter –eq state ^ submitted –database SAMPL –user admin –password “”

This prompts you to specify a database to use:

```
Master: SAMPL
```

and then return the query results. These are from sample database:

```
id            Headline                                              Submitter
--            --------                                              ---------
SAMPL00000011 change due amount is supposed to be red                 engineer
SAMPL00000012 would like logout button to be larger                   engineer
SAMPL00000016 too many spaces in “change due” field                   lead
SAMPL00000019 sales tax incorrect for NH                              lead
SAMPL00000021 inventory report is not running correctly               lead
SAMPL00000024 overriding price operation allows negative number       QE
SAMPL00000027 add item button is out of line with the other buttons   QE
SAMPL00000028 context sensitive help fails from reorder window        QE
SAMPL00000029 formatting does not look right in inventory report      QE
SAMPL00000030 add items fails for large quantities                    QE
SAMPL00000032 shortcut to logout does not work                        QE
SAMPL00000033 unable to add item already in sale list                 QE
SAMPL00000034 cancel sale leaves ite in purchase list                 engineer
SAMPL00000036 inventory report is displaying an empty column          engineer
SAMPL00000037 need report for items ordered on a given day            engineer
SAMPL00000038 sales tax amount is offset from label                   engineer
SAMPL00000039 need automatic logout with QEeout                       engineer
```

```
SAMPL00000040 spelling error in help for override price            engineer
Count: 22
```

Finally, the user can specify the output using the **–output_file** *filename* parameter. This can then be used to mail output to the administrator, for example. The complete example follows:

cqtool new_query –type defect –field id –field headline –field submitter –eq state \ submitted –database SAMPL –user admin –password “” –output_file \ /tmp/cqoutput

mail cqadmin@yourco.com < /tmp/cqoutput

rm /tmp/cqoutput

Database and Troubleshooting Guide

F

Rational ClearQuest is supported on a wide range of operating systems and databases. This section describes various techniques for diagnosing run-time problems that affect databases and database connections.

Oracle Connection Issues on UNIX

This section lists common error message types, gives examples of commands and resulting error messages on a ClearQuest host running UNIX, and provides suggestions for further analysis. It also provides information about generally useful techniques for resolving problems with Oracle connections on UNIX.

Database Connection Options

When connecting to Oracle databases, ClearQuest uses the database property **connect_options**, which determines behavior of the client under certain configurations. Default **connect_options** for all clients are established when the schema repository is created, stored in the schema repository, and replicated to each client when it connects. Clients that are running other versions of Oracle client software must override the default **connect_options** by using the **installutil registeroracleoption**s command.

If a user is unable to connect to an Oracle database from a specific client and the error (which can be viewed by selecting the **Details** check box on the login error message dialog box) indicates that a version of the OpenLink ODBC driver for Oracle references the wrong Oracle client version, you should either upgrade the Oracle client software on that host to the version specified in the default **connect_options** or run **installutil registeroracleoptions** on the client to override the default **connect_options**. The **installutil** executable is located in the ClearQuest installation directory.

The following **installutil registeroracleoptions** command specifies that this client host should use a **connect_options** string in which the value of CLIENT_VER is **8.1**. Other connect_options values used by the client remain as specified in the schema repository.

Note: You must be logged in as the machine administrator to make this change.

installutil registeroracleoptions "CLIENT_VER=8.1"

This command creates a registry key under **HKEY_LOCAL_MACHINE\Software\RationalSoftware\ClearQuest\2003.06.00\Core** with the name **OverrideOracleConnectOptions** and the value **CLIENT_VER=8.1**.

Note: If you reinstall ClearQuest, this registry setting may be deleted as part of the reinstall. Make sure it is set for the client after each install.

For more information on **installutil**, see *Using installutil* on page 369.

Unknown Host Errors

Unknown host errors occur when ClearQuest attempts to contact the OpenLink Request Broker and the host specified in the connect string but the host name cannot be resolved to an IP address or the IP address does not exist.

In this example, the connect string includes a host name (**badhost**) that does not resolve to an IP address. This type of error could also occur during initial registration of the ClearQuest schema repository or during ClearQuest logon processing.

pdsql –v ora –s badhost:SID –u system –p manager –co "SERVER_VER=8.1"

```
OpenLink: RPC: Unknown host
EXCEPTION: [OpenLink][ODBC]RPC: Unknown host
State: 08004 Native: 0
Connect String used: SVT=Oracle 8;
DRIVER=/files/a/rational/releases/ClearQuestClient.
2002.05.00/linux/shlib/db_ORACLE8;HOST=badhost; PROTOCOL=TCP/IP;
UID=admin; DATABASE=SID
```

If you receive an unknown host error:

1. Determine the host that ClearQuest is attempting to access for this particular operation. Do not assume that this is the host specified in the input command. To be sure, check the HOST parameter of the **connect_options**. In the example, it is **badhost**.

   The host specified in the HOST parameter should match the host name of the database server. If it does not, use the procedure in *Verifying ClearQuest Database Settings* on page 393 to determine the source of the host name, then retry the failed operation.

2 If the host specified matches the host name of the database server, then use the procedure in *Pinging a Host* on page 392 to determine or repair connectivity and retry the failed operation.

Program Unavailable Errors

Program unavailable errors occur when ClearQuest attempts to contact the OpenLink Request Broker on the host specified in the **connect_options** and the destination host is available but the OpenLink Request Broker cannot be contacted.

In this example, the name **goodhost** can be resolved to an IP address, but the OpenLink Request Broker on **goodhost** can not be contacted. This type of error could also occur during initial registration of the ClearQuest schema repository or during ClearQuest logon processing.

pdsql –v ora –s goodhost:SID –u system –p manager –co "SERVER_VER=8.1"

```
OpenLink: RPC: Program unavailable
EXCEPTION: [OpenLink][ODBC]RPC: Program unavailable
State: 08004 Native: 0
Connect String used: SVT=Oracle 8;
DRIVER=/files/a/rational/releases/ClearQuestClient.
2002.05.00/linux/shlib/db_ORACLE8;HOST=goodhost; PROTOCOL=TCP/IP;
UID=admin; DATABASE=SID
```

If you receive a program unavailable error:

1 Determine the host that ClearQuest is attempting to access for this particular operation. Do not assume that this is the host specified in the input command. To be sure, check the HOST parameter of the Connect String. In the example, it is **goodhost**.

   The host specified in the HOST parameter should match the expected host name of the database server. If it does not, use the procedure in *Verifying ClearQuest Database Settings* on page 393 to determine the source of the host name, then retry the failed operation.

2 If the host specified matches the expected host name of the database server, then use the procedure in *Verifying That the OpenLink Request Broker Is Running* on page 394 to determine or repair connectivity, then retry the failed operation.

Oracle Not Available Errors

Oracle not available errors (Oracle error code ORA-01034) occur when ClearQuest attempts to contact the OpenLink Request Broker on the host specified in the **connect_options**. The OpenLink Request Broker is available, but cannot contact the Oracle instance on the database server. This typically happens when the Oracle

instance id (SID) is incorrect in the **connect_options** or the Oracle database or its listener process are not running on the database server.

In this example, goodhost resolves to an IP address for the database server host, but **badSID** is not a valid Oracle instance ID. This type of error could also occur during initial registration of the ClearQuest schema repository or during ClearQuest logon processing.

pdsql –v ora –s goodhost:badSID –u system –p manager –co "SERVER_VER=8.1"

```
OpenLink: [Oracle Server]ORA-01034: ORACLE not
available[SQLSTATE:S1000]
EXCEPTION: [OpenLink][ODBC] ORA-01034: ORACLE not available State:
S1000
Native: 0
Connect String used: SVT=Oracle 8;
DRIVER=/files/a/rational/releases/ClearQuestClient.
2002.05.00/linux/shlib/db_ORACLE8;HOST=goodhost; PROTOCOL=TCP/IP;
UID=admin; DATABASE=badSID
```

If you receive an Oracle Not Available error:

1. Determine the host that ClearQuest is attempting to access for this particular operation. Do not assume that this is the host specified in the input command. To be sure, check the HOST parameter of the connect_options. In this example, it is **goodhost**.

   The host specified in the HOST parameter should match the host name of the database server. If it does not, use the procedure in *Verifying ClearQuest Database Settings* on page 393 to determine the source of the host name, then retry the failed operation.

2. If the host specified matches the expected host name of the database server, use the procedure in *Verifying Oracle Connectivity* on page 395 to verify the exact SID and current state of the Oracle instance and then retry the failed operation.

Diagnostic Procedures

This section describes some common diagnostic procedures that you may need to use when resolving Oracle connectivity problems on UNIX.

Pinging a Host

Much of ClearQuest connectivity to UNIX is dependent on being able to access a particular host as it has been specified during database configuration. You can use the UNIX ping utility to verify that a specific host can be contacted by name.

1 From the UNIX host that needs connectivity to the remote host, enter:

ping *remote_host_name*

If **ping** is not in the current user's path, it can typically be found in /usr/sbin/ping.

2 Depending on the client operating system, a successful ping will return:

remote_host_name `is alive`

or a message indicating that a certain number of bytes were received from that host.

3 There are several possible responses if **ping** fails to contact the host:

a `Unknown Host` *remote_host_name*

This response implies that the name server for the current host does not recognize the specified *remote_host_name*. If the remote host is a Windows machine, the site name server will likely need configuration to support name resolution from the UNIX client. Contact your site IT organization to assist in resolving this problem.

b `No answer from` *remote_host_name* or `Destination Host Unreachable`

No answer from *remote_host_name* implies that the destination host name is recognized by the current host but network configuration or availability of the target host prevents network connectivity. Contact your site IT organization to assist in resolving this problem.

Verifying ClearQuest Database Settings

On UNIX, ClearQuest makes an initial connection to the schema repository using information entered by the administrator in the Register Databases dialog box. The database connection information for the schema repository and all user databases is then downloaded to the client and stored in the ClearQuest databases directory. The connection information that was originally provided when the database was created using the ClearQuest maintenance tool, or later modified using the modify database properties feature of ClearQuest Designer, is the source of the data copied to the UNIX client.

To verify that these settings are correct:

1 Enter the following command on the ClearQuest client:

cqreg show

This displays all information that the ClearQuest client has about the schema repository and user database properties.

2 For Oracle databases, verify the **HOST** parameter in the **connect_options** string matches the host name of the database server. This host name must be accessible by the ClearQuest client (see *Pinging a Host* on page 392). The Server parameter refers to the SQLNet alias specified for the Windows client. This is not used by UNIX clients and may be ignored.

3 If the information is incorrect or out of date, do the following:

   a Check the schema repository properties in the ClearQuest maintenance tool and the user database properties in ClearQuest Designer. Resolve the discrepancies and upgrade the user database if you make any changes.

   b Execute the following command from the ClearQuest command line. (ClearQuest does not automatically refresh database connection information from the schema repository).

   cqreg refresh

Verifying That the OpenLink Request Broker Is Running

ClearQuest communicates to the Oracle database by using the OpenLink Request Broker. The OpenLink driver contacts the OpenLink Request Broker, which in turn spawns a database-vendor-specific agent to handle the actual requests. In order for this to function properly, the OpenLink Request Broker must be running and available on the database server host.

To verify that the OpenLink Request Broker is running:

1 From the Oracle Database server, enter the following command:

   ps –e | grep oplrqb

   This lists all running processes whose names include the string **oplrqb**. This is the process name for the OpenLink Request Broker. If it is not running, it needs to be started and enabled for restart at system boot time.

2 From the OpenLink installation directory, run the following command to start the OpenLink Request Broker if necessary:

   oplrqb start/stop/start

   Select option **S**. This starts the OpenLink Request Broker. To ensure that the request broker is started automatically at boot time, see the *Installation Guide* for Rational Server Products for Windows or *Installation Guide* for Rational ClearCase Product Family for UNIX.

Verifying Oracle Connectivity

ClearQuest communicates with the Oracle database by using the OpenLink Request Broker. The OpenLink Request Broker acts like any other Oracle client and must have connectivity and privileges to access the Oracle database.

To verify Oracle connectivity:

1. Log on as the user that runs the OpenLink Request Broker.
2. Traverse to the OpenLink install directory.
3. Use a text editor to open the OpenLink rules file oplrqb.ini. Search for the string **generic_ora8** (assuming that Oracle 8.*x* is in use). Look for the ORACLE_HOME and ORACLE_SID variables. Ensure these are set correctly (case must be correct).
4. To determine the actual SID executing on the Oracle database server, run:

   ps –ef | grep pmon

   and look for a line containing a string of the form `pmon`*nnnn*, where *nnnn* represents the SID.

When ClearQuest registers databases, it uses the information specified in the **Register Database** dialog box to initiate the first connection. The information stored in the schema repository is then used for subsequent connections. It is important to verify the connect options string by checking **Database > Database Properties** in ClearQuest designer. Verify that HOST, SID, and SERVER_VER are correct. In addition, verify that LOB_TYPE is set to LONG or CLOB.

Other Oracle Issues

This section lists other issues that may affect ClearQuest when using an Oracle database.

Searches with the Contains Operator Are Always Case-Sensitive

When using an Oracle database, searches using the **Contains** operator are always case-sensitive.

Debugging e-mail Notification Issues with dbwin32

ClearQuest, when enabled by means of a registry setting, writes useful debug information to the Windows debug log. This can be viewed using the dbwin32 tool

located under the ClearQuest install directory or any other tool that can browse the Windows debug log.

1 Create .reg files for native client and/or Web server machines with the following information:

Native Client

REGEDIT4
[HKEY_CURRENT_USER\Software\Rational Software\ClearQuest\Diagnostic]
“Trace”=“Email”
“Output”=“ODS”
"EMailSendVB”=“ODS”
Webserver
REGEDIT4
[HKEY_USERS\.default\Software\Rational Software\ClearQuest\Diagnostic]
“Trace”=“Email”
“Output”=“ODS”
“EMailSendVB”=“ODS”

2 Stop ClearQuest.

3 Import the .reg file created in Step 1 into your registry.

4 Start ClearQuest.

If you are running Windows 2000, you are required to have local administrator permissions to view debug output.

Index

A

action hooks
  about 204
  adding 206
  deleting 207
  e-mail notification and 246
  example, initialization 326
  list of supported 205–206
  modifying 207
  order of execution 208–210
actions and action types
  about 119
  adding to state model 121
  default 124
  deleting 125
  hooks 123
  list of supported 120
  modifying 123
admin user, default ID 161
administration tasks
  about 14
  delegating, and required permissions 21
  users and groups 161–162
administration tools 22
API
  about 11, 217–218
  common calls 218
architecture
  UNIX 12
  Windows 8

B

bkt_tool utility 382

C

ccase-home-dir directory xxviii
change management 1, 3
change requests
  lifecycle 2
  record types for 94
CharacterSetValidation package 38
ClearQuest data code page
  See data code pages
ClearQuest Designer 22–23, 143
command-line utilities 369
connections
  about 7, 55
  data code page value 55
  deleting 57
  duplicating 58
  editing 57
  profiles 60–61
  renaming 56
  UNIX client 58
controls on forms
  about 146
  ActiveX 349
  adding 148–150, 152
  attachment lists 350
  buttons 364
  check boxes 351
  combo boxes 352

deleting 154
displaying images 364
drop-down combo boxes 355
drop-down lists 353
duplicate base 356
duplicate dependents 357
group boxes 358
history 358
list boxes 359
list view 360
modifying 153
option button 361
parent/child list 362
script trigger 212
tabs 157–159
text boxes 366
conventions, typographical xxviii
cqload utility 376
cqtool utility 385
cquest-home-dir directory xxviii
customer support xxix

D

data code pages
about 16, 27
applying package for 38
changing, effects of 32
connections and 55
data corruption and 28
enforcing values for 27, 30
guidelines for selecting values 29
integrations and 297
interoperation with previous releases 38
issues for integrations 41
list of supported 31
MultiSite issues 40
setting 48
setting values 35
database management
about 43–44
backups 47
code page settings 48
connections 55
creating aliases 47
creating copies 371
creating databases 49
creating empty databases 47
databases
See also schema repositories, user databases, vendor databases
about 6
effect of data code page value 31
DB2 databases 76

E

e-mail notification
about 19, 235
action controls for rules 238
action hooks 246
addresses for rules 240–241
configuring users 244
creating rules 236
disabling rules 238
display fields for rules 239
enabling 236
enabling submission 247
format example 256
mailboxes 248
operators 238

Rational E-Mail Reader 248, 253
record format 254–255
round-trip 245
rule controls 237
troubleshooting 395
Web settings 227
exporting data
about 259, 267
cqload utility 376
dynamic lists 381
exportintegration subcommand 378
exportschema subcommand 376

F

feature levels, raising 66
field hooks
about 194
choice list 199–200, 202
creating 195
deleting 197
field dependencies 197–198
list of supported 195
modifying 196
order of execution 208–209
fields
about 100–101
adding to record type 101–102
deleting 108
linking records 109–110
modifying after schema checkin 108
parent/child links 112
reserved keywords in names 102, 108
forms
about 143
creating 144
deleting 146
exporting 160
importing 160
modifying 145–146
multiple platforms 146
renaming 145
reusing 159

G

global scripts
about 214–215
creating 215
deleting 216
modifying 216

H

hooks
about 11, 192
and action types 123
examples, CAL methods for UCM integration 342
examples, choice list 318
examples, dependent list 317
examples, set value of parent record 327
external applications 217
finding text in scripts 220
operating context 217
planning and development issues 192
schema validation and errors in 86
scripts for 193–194

I

importing data
- about 259
- attachment file format 266–267
- attachment files 271
- cqload utility 376
- data types supported 265
- discarded data 272
- duplicate records 275
- dynamic lists 381
- error recovery 275
- forms 160
- history file format 266
- import file delimiters 263
- import file format 264
- import files 262
- modifying forms 146
- prerequisites 271
- record IDs 261
- updating records 276

importintegration subcommand 379
importschema subcommand 377
installutil utility 369
integrations
- about 295
- code page issues 297
- dependent, adding 301–302
- dependent, list of 296
- e-mail system 296
- independent, list of 296
- independent, packages required 300–301
- Microsoft Visual SourceSafe 297, 310–311
- package upgrades 312
- Project Tracker 303–304
- Rational Administrator 302–303
- Rational TeamTest 304–305
- testing 300
- UCM 305–310
- viewing current packages 299

L

login, restricted characters 164

M

mastership 176
Microsoft Access 74
MultiSite, and data code pages 40

O

Oracle databases 75
- data code pages supported 48
- troubleshooting search problems 395
- troubleshooting UNIX connections 389
- unlocking, command example 374
- Web access setting 225

P

packages
- about 129
- data code page values 38
- enabling record types 298
- how to apply 135
- how to upgrade 140
- list of available 132

list of supported 279–280, 282–288
modifying components of 131
removing from schema 131
state type models 291–292
verifying applied 134
permissions 21
privileges
changing 170
list of supported 163
required for hooks 192

Q

queries, building 218

R

record scripts
about 210
adding to record type 213
deleting 214
form control events 212
modifying 214
Web access 211
record types
about 94
adding scripts 213
adding to schema 95
applying packages 298
controlled, and security 180
default 97
deleting 100
families 97–99
mapping to state types 116
renaming 99
security context 181
state models 288–289
stateless 96
states 94
treatment of existing when package applied 138
records
e-mail format 254
hiding 179
importing duplicate 275
linking 109–110
manipulating from API 218
restricting access 179
security context 181–182
updating imported 276
reports
generating with cqtool 385
Web client limitations 223

S

schema repositories
about 6
copying 371
creating 49
data code page value 28
data code page values 35–36
moving 62, 64
schemas
about 5, 79
adding components 129
changing database association 92
checking in 88
checking out 82
copying 376
creating from Blank schema 84

customizing 80, 93
defining requirements 17
deleting 91
predefined 277–278
predefined, how created 131
predefined, list of 17–18
predefined, modifying 126
restricting user actions 174
test database, setting 87
testing 80
upgrading user databases 89
validating changes 86
verifying packages applied to 134
viewing packages in 299
security
about 179
additional features 188
controlled record type 180
design issues 182
example 184
security context field 179, 181
SQL Anywhere 74
SQL Server databases 75
copying, command example 375
state models
about 113
adding states 116
changing state names 117
customizing 115
deleting states 118
design considerations 125
for record types 288–289
transition matrix 115
state type models
packages 291–292
state types
mapping package to schema 139
mapping to record types 116

T

test databases
creating 80
setting 87
troubleshooting 389
typographical conventions xxviii

U

UNIX client, database connections 58
upgrading features 66
User Administration dialog box 162
user databases
adding users 164
changing schema 92
creating 52
data code page value, setting 40
deleting 71
editing user profiles 169
granting access 167
moving 68–69
moving workspace items 382
removing access 168
restoring deleted 72
updating properties 70
upgrading information 162, 169
upgrading schema 89
upgrading user information 162
viewing properties 73
viewing subcribers 162
users and groups
about 161

active and inactive 171
administering groups 172–174
changing privileges 170
importing and exporting profiles 175
restricting access to actions 174–175
security context 180
supported privileges 163–164

V

vendor databases
about 8
data code page setting 48
installutil parameters 369
reserved keywords in, and fields 102
specifying parameters 73
supported character set 32, 34

W

Web access
about 223
ASP scripts 226
behavior of hooks 231–232
customizing interface 224
dependent form fields 198
display options 227–229
display options, predefined 229–230
e-mail notification 227, 244–245
restrictions and limitations 223, 230
settings for 225–226