Administering Rational ClearQuest

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

Rational®
the e-development company™

IMPORTANT NOTICE

Printed in the U.S.A.

Part number: 800-024552-000

Version: 2001A.04.00

Contents

Before you begin

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

Audience

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

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

Other ClearQuest documentation

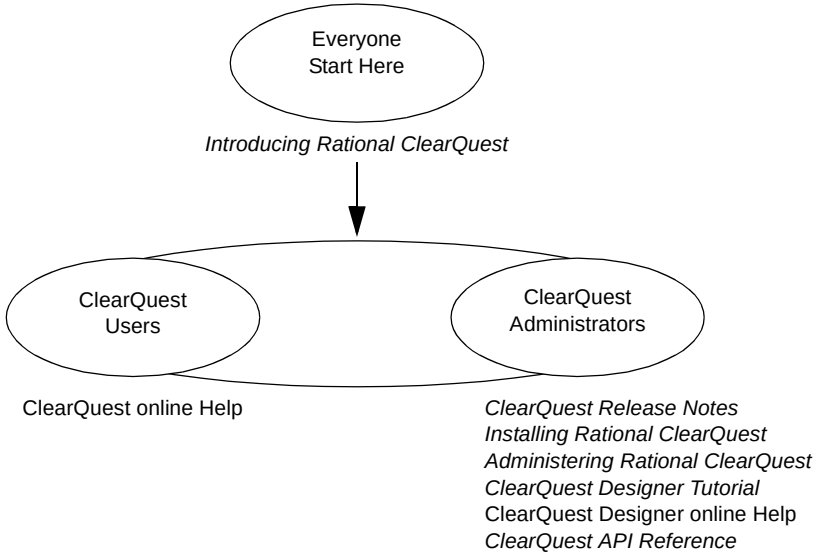
In addition to this guide, ClearQuest includes the following printed documentation:

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

ClearQuest includes the following online documentation:

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

ClearQuest documentation roadmap



Contacting Rational Software

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

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

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

1 Understanding ClearQuest administration

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

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

Note: The term *administrator*, as used in this guide, means any person with user-access privileges above that of Active User. For definitions of user-access privileges, see "ClearQuest user privileges" on page 125.

Starting ClearQuest Designer

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

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

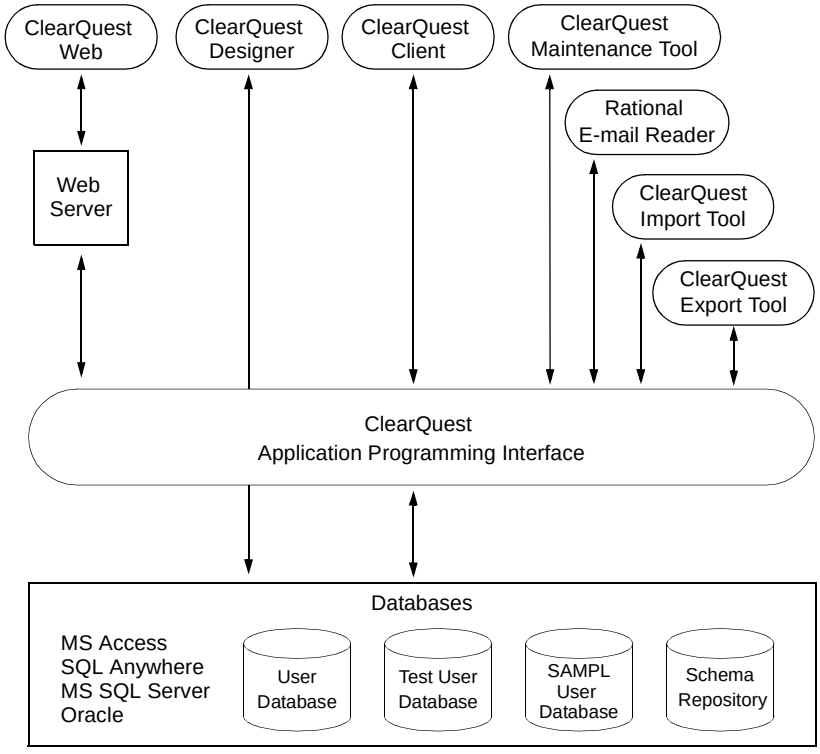
Note: The first thing you might want to do is add a password to the *admin* account or create a new user account to limit the access to ClearQuest Designer. See Chapter 7, "Administering users," specifically "Modifying user profiles" on page 128.

Overview of ClearQuest architecture

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

How components work together

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



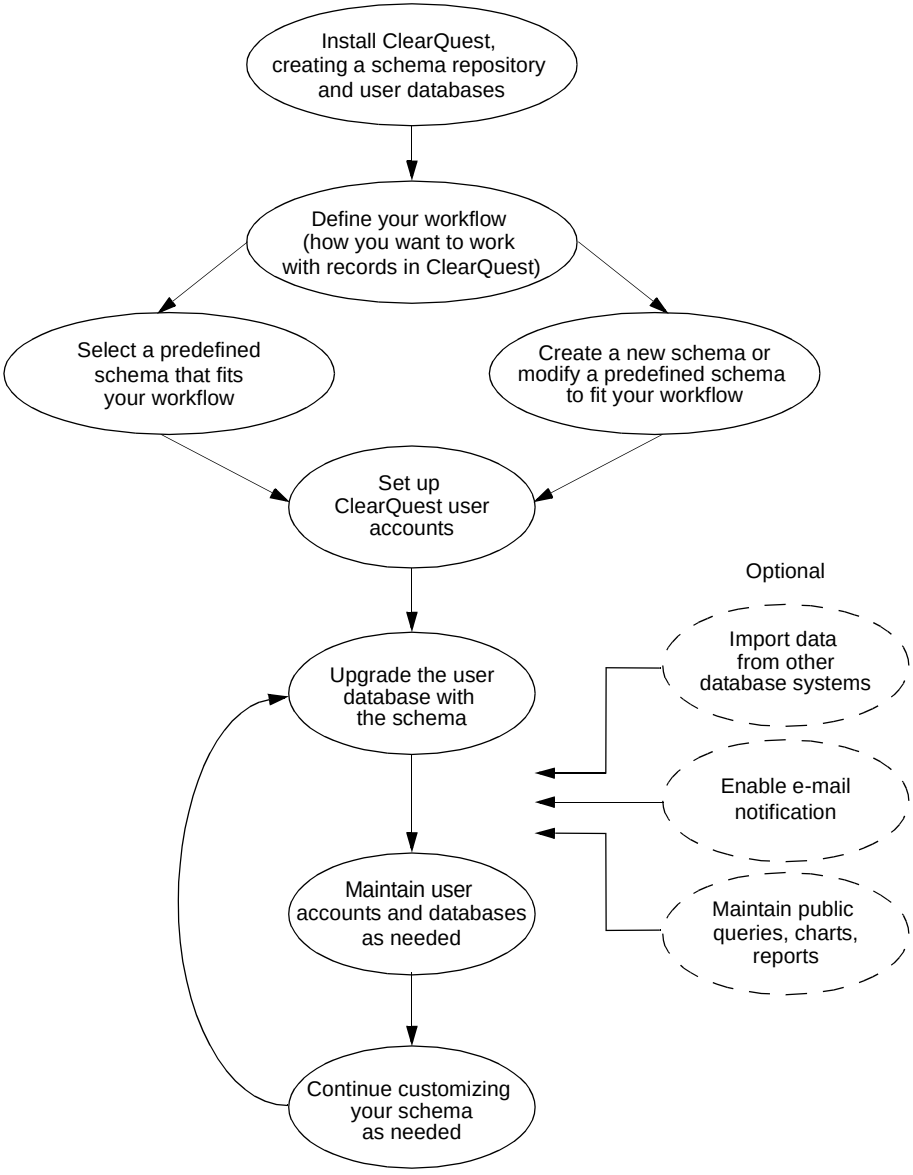
How components are used

Here's how you use ClearQuest components:

Component	Used by	Use to
Client tools		
ClearQuest for Windows	Everyone	Submit, modify, and track defect and change requests, and analyze project progress by running queries, charts, and reports.
ClearQuest for UNIX	Everyone	Submit, modify, and track defect and change requests, and analyze project progress by running queries.
ClearQuest Web	Everyone	Access ClearQuest across multiple platforms through Netscape Navigator® or Microsoft Internet Explorer. Submit change requests and run queries. On Windows you can also run charts and reports.
ClearQuest Microsoft Project Tracker	Everyone	Exchange and synchronize data between Microsoft Project 2000 and ClearQuest.
Administrator tools		
ClearQuest Designer	ClearQuest administrator	Customize ClearQuest, manage ClearQuest schemas and databases, and administer users and user groups.
ClearQuest Import Tool	ClearQuest administrator	Import data, including records, history, and attachments, from other change request systems.
ClearQuest Export Tool	ClearQuest administrator	Export ClearQuest data from one ClearQuest user database to another user database that uses a different schema.
ClearQuest Maintenance Tool	Everyone	Set up and connect to the schema repository during installation and when you upgrade to a new ClearQuest version.
Rational E-mail Reader	ClearQuest administrator	Enable ClearQuest users to submit and modify records by e-mail. See Chapter 10, "Administering ClearQuest E-mail."

Overview of administrator tasks

Below is an overview of the ClearQuest administrator tasks.



Working with ClearQuest schemas and databases

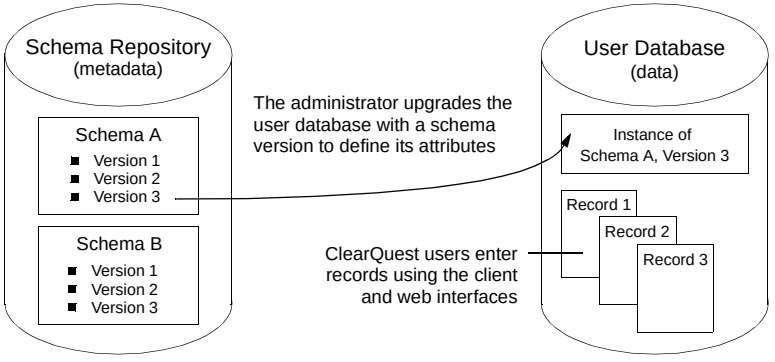
A ClearQuest schema contains the metadata that defines your workflow, that is, how your users work with records within ClearQuest. It includes record definitions, form and field definitions, record states and the actions that can be performed on records, and optional hook code that customizes your workflow.

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

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

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

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



The schema repository can hold multiple schemas and versions

The user database is based on one schema and its successive versions

Defining your workflow

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

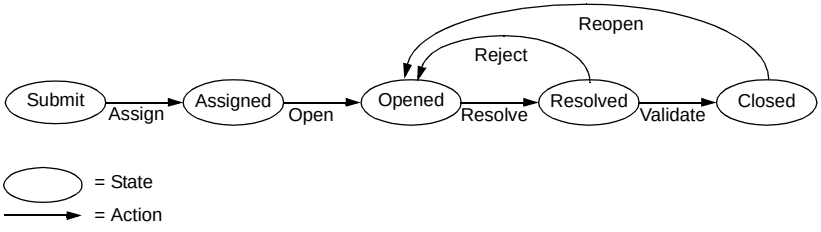
ClearQuest predefined schemas are made up of packages that contribute specific functionality to the schemas. For example, the Email package included in most schemas enables ClearQuest to send automatic e-mail notifications. The easiest way to add functionality to a schema is to add one or more ClearQuest packages. See “Adding packages to a schema” on page 47.

Working with record types, states, and actions

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

Defining a state model

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



Customizing your workflow

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

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

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

Before you begin customizing

Before you begin customizing ClearQuest, be sure to read:

- Chapter 3, “Working with ClearQuest Schemas.” This chapter is essential reading whether you are using a predefined schema or creating a schema from scratch. It details the process for working with schemas, from checking them out of the schema repository to upgrading the user database with the latest schema changes.
- Chapter 4, “Customizing a schema,” especially the section “Planning schema customizations” on page 54.
- Appendix A, “ClearQuest schemas and packages” for a complete list of ClearQuest’s predefined schemas and schema packages.

Getting ClearQuest users started

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

Task	Do this	To find out how, see
Install ClearQuest and set up databases	Create a schema repository and one or more user databases	*Installing Rational ClearQuest*
Define your workflow	Plan states and actions; create a state transition model; define record types, forms and fields; plan hook code	This chapter and Chapter 3, “Working with ClearQuest Schemas” Chapter 4, “Customizing a schema”
Select a predefined schema that fits your workflow		Appendix A, “ClearQuest schemas and packages”
Customize the selected schema or build a new schema from scratch (optional)	Begin by adding predefined packages to build the functionality you need	“Adding packages to a schema” on page 47 Appendix A, “ClearQuest schemas and packages”
	Customize the schema as needed	Chapter 3, “Working with ClearQuest Schemas” and Chapter 4, “Customizing a schema”
	Write hook code to further refine your workflow (optional)	Chapter 8, “Using hooks to customize your workflow”
Upgrade your user database with the selected/customized schema		Chapter 3, “Working with ClearQuest Schemas”
Create ClearQuest login accounts and user groups and define user access permissions	Add ClearQuest users, subscribe them to a user database, create user groups, set permissions	Chapter 7, “Administering users”

Task	Do this	To find out how, see
Import data from other database systems (optional)	**Note:** You must set up user accounts before importing data from other database systems	Chapter 11, “Importing data into ClearQuest”
Enable e-mail notification (optional)	Have each ClearQuest user enable e-mail notification in ClearQuest	ClearQuest Help
	Set up e-mail rules	Chapter 10, “Administering ClearQuest E-mail”
Enable ClearQuest Web (optional)		*Installing Rational ClearQuest* Chapter 9, “Administering ClearQuest Web”
Customize ClearQuest queries, charts, and reports (optional)	Customize ClearQuest queries, charts, and reports and save them in the ClearQuest Public Queries folder	ClearQuest Help
	Alternatively, give permissions to other users, such as project managers, to do this	Chapter 7, “Administering users”

2 Managing databases

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

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

ClearQuest required databases

ClearQuest requires at least two databases:

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

For a description of how ClearQuest databases and schemas work together, see “Working with ClearQuest schemas and databases” on page 5.

Where to keep ClearQuest databases

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

Maintaining database integrity

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

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

Selecting a database vendor

ClearQuest supports the following databases:

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

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

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

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

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

Creating new databases

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

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

Note: You must create your schema repository before you can begin customizing ClearQuest. You should also set up a test user database for testing new schema customizations. See "Working with a test database" on page 37.

Once your databases are set up, you can begin using and customizing ClearQuest. See Chapter 3, "Working with ClearQuest Schemas".

Backing up your data

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

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

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

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

Overview of database maintenance

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

Activity	Tool/Interface	Comments
Create a new schema repository or user database	Your database vendor tool, and ClearQuest Maintenance Tool and ClearQuest Designer	To create new empty databases, use the database vendor of your choice. To connect and initialize the schema repository and SAMPL user database, use the ClearQuest Maintenance Tool. To create new user databases, use ClearQuest Designer. **Note:** You can create Microsoft Access and SQL Anywhere databases from within ClearQuest Designer.
Back up a database	Your database vendor tool	To preserve your data, perform regular backups.
Modify data	ClearQuest client, web interface, ClearQuest Designer, ClearQuest API	To maintain database integrity, **use only ClearQuest tools to directly modify ClearQuest data or tables**. Do not use your database vendor tools to modify ClearQuest data.
Use a newer version of a schema for a user database	ClearQuest Designer	See “Applying schema changes to a user database” on page 44.
Change the schema for a user database	ClearQuest Export Tool and ClearQuest Import Tool	Use the ClearQuest Export Tool to export the data from the user database. . . . . . then use the ClearQuest Import Tool to import the data into a new user database associated with a new schema. See “Moving a database to a different schema” on page 22.

Activity	Tool/Interface	Comments
Move a database or change database vendors	ClearQuest Designer or ClearQuest Maintenance Tool	To move a user database to a new location or to a different database vendor, use ClearQuest Designer. To move a schema repository to a new location or to a different database vendor, use the ClearQuest Maintenance Tool. See “Moving a database to a new location or to a new vendor” on page 17.
Delete/undelete a user database	ClearQuest Designer	ClearQuest Designer removes the link between the schema repository and the database, but does not delete the physical database. To delete the physical database, use your database vendor tool. See “Deleting a user database” on page 33.
Upgrade to a new ClearQuest version	ClearQuest Maintenance Tool	To upgrade to a new version of ClearQuest, use the ClearQuest Maintenance Tool.

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

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

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

Preparing to move a database

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

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

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

2. Notify all users to log off from ClearQuest.

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

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

3. Back up all your databases.

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

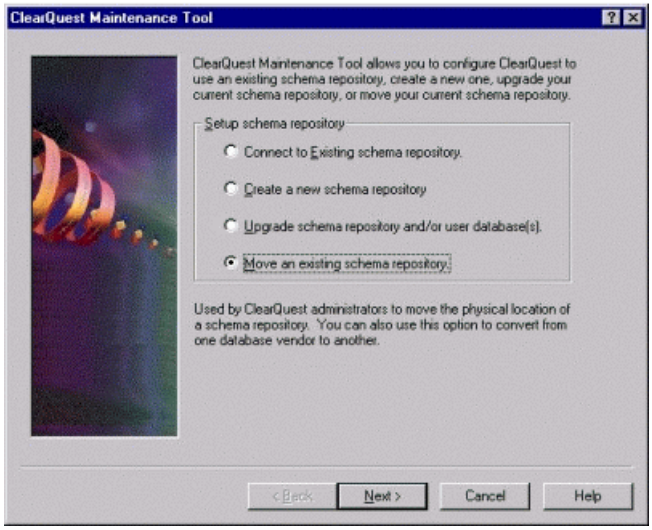
Moving a schema repository

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

Note: You must have Super User privileges to move a schema repository. See “ClearQuest user privileges” on page 125.

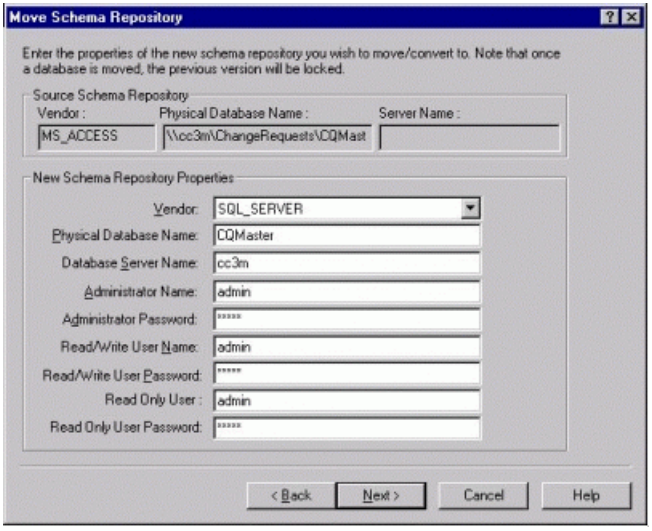
To move a schema repository to a new location:

1. Complete the steps in “Preparing to move a database” on page 17.
2. Select **Rational ClearQuest Maintenance Tool** from the Start menu.
3. In the Maintenance Tool, select **Move an existing schema repository** and click **Next**.



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

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



6 When the conversion is complete, exit the ClearQuest Maintenance Tool.

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

Moving a user database

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

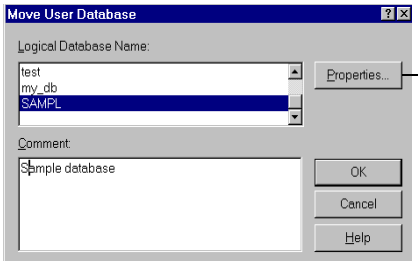
Note: If you have already moved the database outside of ClearQuest and only need to change the database properties, use the tasks described in “Updating user databases” on page 28.

1 Complete the steps in “Preparing to move a database” on page 17.

2 Select **Rational ClearQuest Designer** from the Start menu.

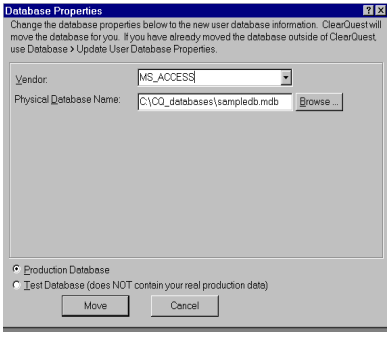
3 In ClearQuest Designer, select **Database > Move User Database**.

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

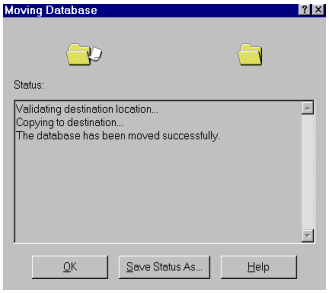


Click to open the Properties dialog box

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



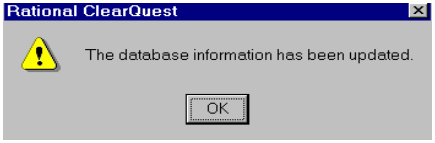
6 In the Moving Database dialog box, the message in the Status area informs you whether the database move is successful.



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

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

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



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

Moving a database to a different schema

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

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

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

To move a database to a new schema:

1. Before you begin, read "Important considerations when moving to a new schema" on page 23.
2. Create or modify the schema you want to start using.

   See Chapter 3, "Working with ClearQuest Schemas" and Chapter 4, "Customizing a schema."
3. Create a new user database and associate it with the new schema.

   Select **Database > New Database** in ClearQuest Designer to create a new user database. ClearQuest Designer prompts you to associate the new database with a schema.
4. Use the ClearQuest Export Tool to export the data from your current user database.

   See "Using the ClearQuest Export Tool" on page 29.
5. Use the ClearQuest Import Tool to import the data into the new database.

   See Chapter 11, "Importing data into ClearQuest" for information on using the Import Tool.

Important considerations when moving to a new schema

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

- After you create a new schema, be sure to test it on a test database before upgrading your new user database. Import a small subset of data and test the schema to make sure it works correctly. See “Working with a test database” on page 37.
- If your new schema has dynamic lists, be sure to populate those lists before importing your data. See “Working with choice lists” on page 149.
- If your data has dependencies for reference or reference_list fields, you should import the data in the proper order. For example, you might need to import the project record type first, followed by the main data record type, then attachments, and finally history.
- If you are moving to a new schema repository, you must export user information from the old schema repository and import it into the new one before moving other data. You can use ClearQuest Designer to export user data. See “Exporting and importing users and groups” on page 136.
- If the new schema uses different state names than the old schema, you must edit the export file to change the state names. For example, if the old schema uses the state name *submitted* but the new schema calls that state *new*, you must edit the export file and replace *submitted* with *new* before importing that data into the new database.
- The ClearQuest Export Tool exports the record ID into the display_name field for the history and attachments record types. (See “Working with record types” on page 56.) Remember to map the display_name field to the original “old” ID field when importing data.

Setting up multiple schema repositories

ClearQuest now provides a utility that allows you to set up multiple schema repositories. In previous versions of ClearQuest, when you logged in from the Designer you were automatically connected to the schema repository that was originally configured during installation. Using the `installutil adddbset` command, you can now add additional schema repositories to ClearQuest and see them in a list. You can choose a schema repository from this list when you log into ClearQuest.

A schema repository and its associated databases together comprise a *database set*. When you use the `installutil adddbset` command, you must specify a name for the schema repository and user database—the database set—to connect to. The name you choose is arbitrary. This name will appear in the list of database sets available in the Schema Repository window when you first open ClearQuest, followed by the login screen.

If you have set up multiple repositories with the `installutil adddbset` command:

- You will see a list of database sets when you open ClearQuest.
- You must choose a database set from the list.
- You will be prompted to enter your user name and password to log in.

Adding another schema repository to ClearQuest

To add another schema repository to ClearQuest, enter the following parameters at the command prompt:

```
<your ClearQuest path> \installutil adddbset <dbset_name>
<db_vendor> <server_hostname> <db-path\filename.suffix or
database name> <ro-login name> <ro-login password>
<connection options: use "" if not Oracle; if Oracle,
HOST=<host>;SID=<sid>; server_ver=<ver7>;
client_ver=<ver>
```

Each parameter is explained below:

`<your ClearQuest path>`

The directory where you have installed ClearQuest. For example, `C:\Program Files\Rational\ClearQuest`

<dbset_name>

The name you give to the database set (e.g., the schema repository and its associated user database). The name is arbitrary; you may name the database set anything.

<db_vendor>

The vendor of the database you're connecting to. Following is the correct syntax to use for each vendor:

- For Microsoft Access — `MS_ACCESS`
- For SQL Anywhere — `SQL_ANYWHERE`
- For SQL Server — `SQL_SERVER`
- For DB2 — `DB2`
- For Oracle 7 or 8 — `ORACLE`

<server_hostname>

The name of the server that hosts the database you're connecting to. Following is the correct syntax to use for each vendor:

- For Microsoft Access — The machine host name.
- For SQL Anywhere — The server name.
- For SQL Server — The machine host name of the SQL Server.
- For Oracle — SQL *Net alias for Oracle.
- For DB2 — The database name.

<db-path\filename.suffix or database name>

The name of the database to which you're connecting. For Access or SQL Anywhere, the database name must be in double quotes if there is a space in the file name. Following is the correct syntax to use for each vendor:

- For Microsoft Access — `"\\full\UNC\path\to\file.mdb"` Or, `X:\local\path\to\file.mdb`

- For SQL Anywhere — `\\full\UNC\path\to\file.db` Or, `X:\local\path\to\file.mdb`
- For SQL Server — The name of the database containing your schema repository.
- For Oracle — The database login who owns the tablespace.
- For DB2 — The database name.

<ro_login name>

Read only login name. This parameter is the same login you specified when you installed and configured ClearQuest. Following is the correct syntax to use for each vendor:

- For Microsoft Access — ""
- For SQL Anywhere — “admin”
- For SQL Server — The user ID of the user who owns the database.
- For Oracle — The same parameter as the database entry.
- For DB2 — The DB2 user ID.

<ro_login password>

Read only login password. Following is the correct syntax to use for each vendor:

- For Microsoft Access — ""
- For SQL Anywhere — “admin”
- For SQL Server — The password associated with the ro-login parameter.
- For Oracle — The password associated with the ro-login parameter.
- For DB2 — The DB2 password.

<connection options>

This parameter only applies when connecting to an Oracle database; however, when not connecting to an Oracle database you

must use two sets of double-quotes ("") in the parameter string in place of the Oracle entries.

If connecting to an Oracle database, check with your Administrator for the correct Oracle options. Oracle may include the following options:

- Host — The name of the database server machine. If your ClearQuest deployment includes both Unix and Windows clients, the Host name is mandatory.
- SID — The name of the database instance. If your ClearQuest deployment includes both Unix and Windows clients, the SID name is mandatory.

See “Configuring vendor databases” in the *Installing Rational ClearQuest* guide for more details on connecting to databases and database options.

Examples of using the installutil adddbset command

The following examples illustrate how to use the `installutil adddbset` command at the command prompt to connect to Microsoft Access and Oracle.

MS Access example:

```
E:installutil adddbset foo MS_ACCESS jbob
"jbob\dbs\cqmaster.mdb" "" "" ""
```

Note that double quotes are entered in the `<connection options>` parameter of the string. This parameter of the string *must* be completed to use the `installutil adddbset` command. When not connecting to an Oracle database, a pair of double quotes must be entered instead of the Oracle options.

Oracle example:

```
E:\Program Files\Rational\ClearQuest>installutil
adddbset foo2 ORACLE bar_host cquser cquser password
client_ver=7.0
```

Updating user databases

Use the **Update User Database Properties** menu option to change a selected database's properties if you have moved your user database outside of ClearQuest. (If you have *not* moved your database outside of ClearQuest, use the tasks described in "Moving a database to a new location or to a new vendor" on page 17 to change properties prior to moving the database.)

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

2 Click the desired database name.

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

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

Viewing database properties

1 In ClearQuest Designer, click **Database > View Database Properties**.

2 From the Database Property dialog box, highlight the desired database name and click the **Properties** button to display the properties of the database.

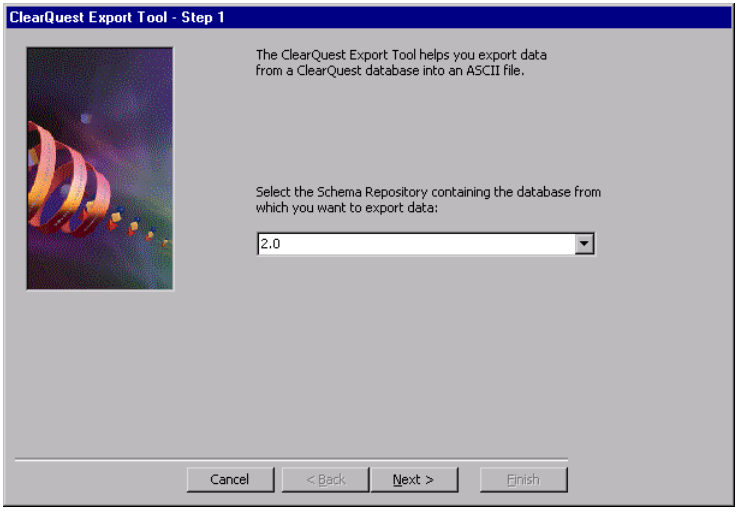
Note: This window is read-only. To move and/or edit database properties, see "Moving a database to a new location or to a new vendor" on page 17 or "Updating user databases" on page 28.

Using the ClearQuest Export Tool

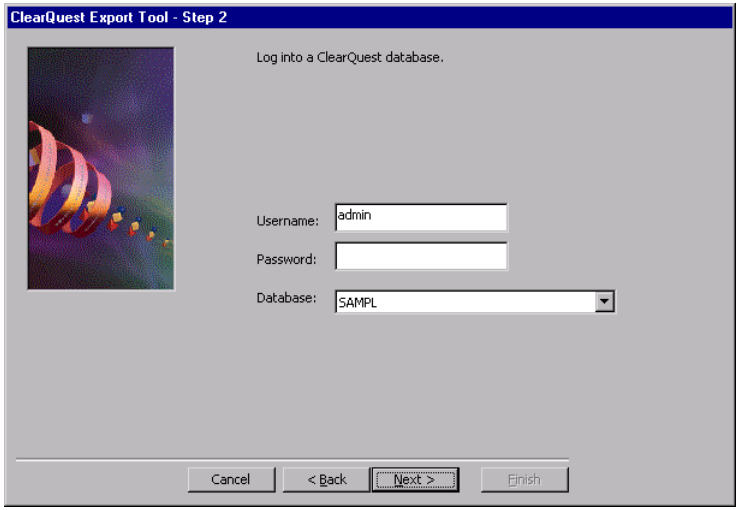
The ClearQuest Export Tool exports data from a ClearQuest database into an ASCII format optimized for the ClearQuest Import Tool.

To run the Export Tool:

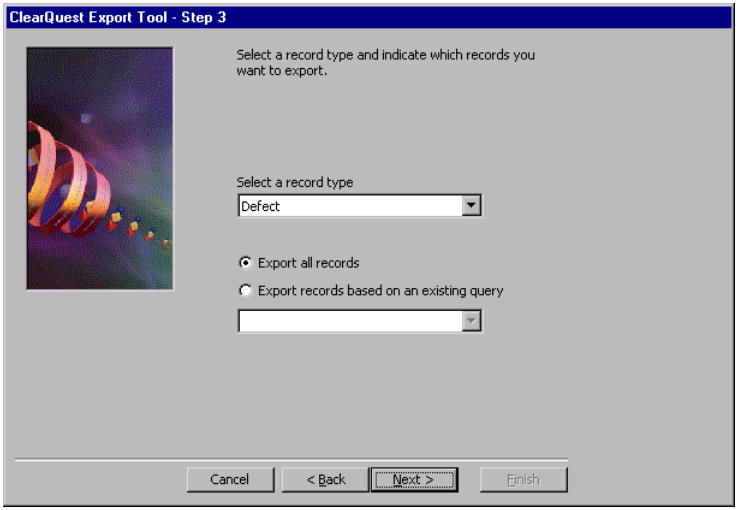
1. Select **Rational ClearQuest Export Tool** from the Start menu.
2. In the Export Tool, select the schema repository containing the database from which you want to export data. The default is the schema repository associated with the current version number of ClearQuest.



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

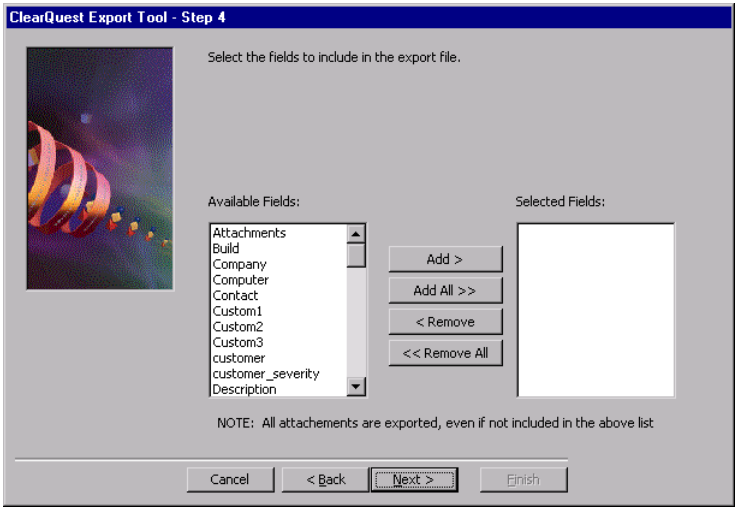


4 Select a record type and indicate which records you want to export. You can export all records or select a ClearQuest query to export specific records.

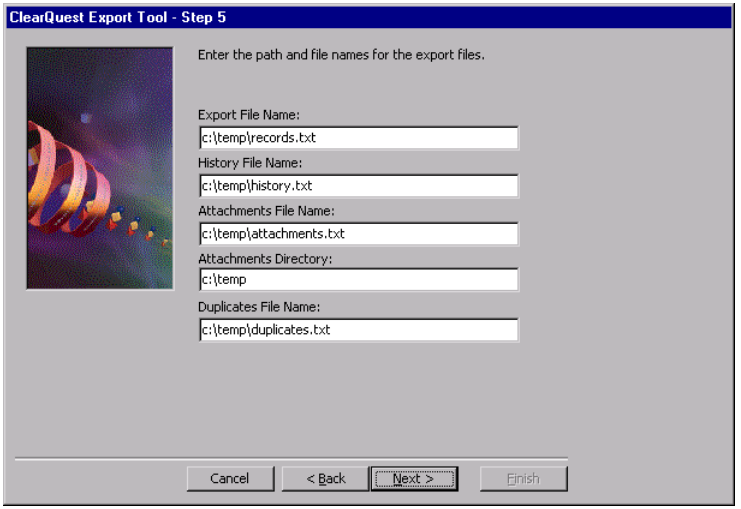


The history, attachments, and duplicates data are automatically exported for the selected record type.

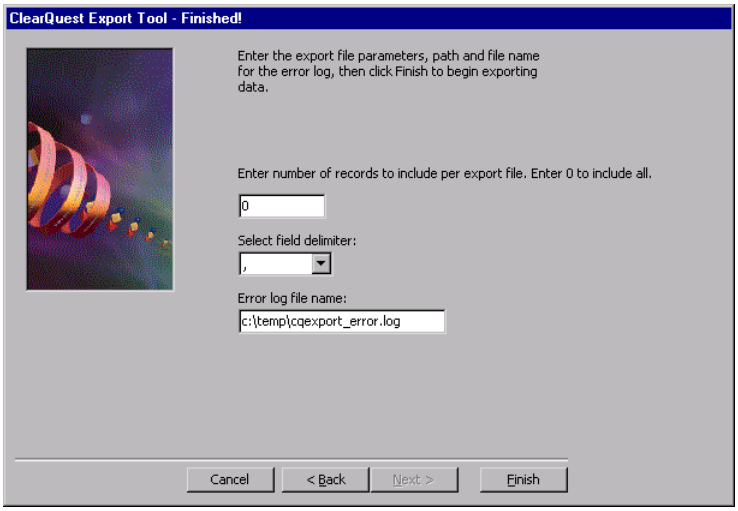
5 Select the fields to export.



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



7 Enter the final export parameters.



- You can choose to include all records in one file or enter the number of records you want per file.
- Select the field delimiter for the export files. The most common format is comma-delimited, but you can also use colons, semi-colons, pipes, or tabs.
- Enter the path and name for the Error log.

8 Click **Finish** to begin the export. ClearQuest displays a dialog box when the export is complete and lists any detected errors.

Deleting a user database

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

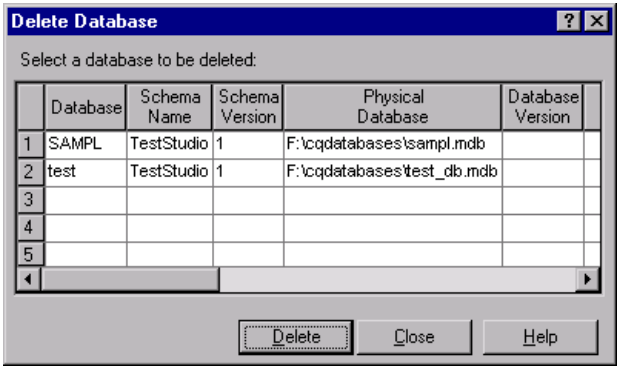
You can restore a deleted database at a later date by restoring the link between the database and the schema repository. See “Undeleting a user database” on page 34.

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

To delete a user database from ClearQuest:

1 Select **Rational ClearQuest Designer** from the Start menu.

2 Select **Database > Delete Database** to open the Delete Database dialog box.



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

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

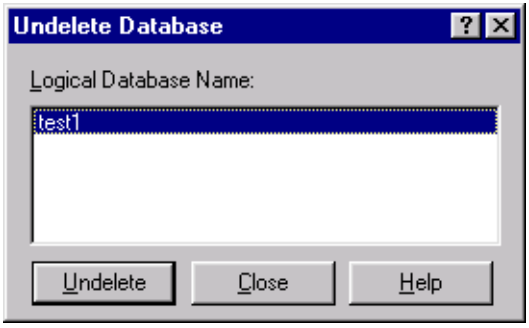
Undeleting a user database

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

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

To undelete a user database:

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



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

3 Working with ClearQuest Schemas

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

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

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

Note: To work with schemas, you need Schema Designer or Super User privileges. To learn how to set user privileges, see Chapter 7, “Administering users.”

More information? For a list of predefined schemas and packages, see Appendix A, “ClearQuest schemas and packages.” For information on how to customize a schema, see Chapter 4, “Customizing a schema.”

Overview of working with schemas

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

1. Create a new user database to be used as a test database and associate it with the schema you want to customize. See "Working with a test database" on page 37.
2. Check out a schema from the schema repository (see "Checking out a schema" on page 39), or create a new schema (see "Creating a new schema" on page 40).
3. Set the test database for the schema by selecting **Database > Set Test Database**. See "Working with a test database" on page 37.
4. Choose the scripting language to use for hook code. See "Selecting a scripting language" on page 40.
5. Customize the schema, validating often, especially when making complex changes. See "Validating schema changes" on page 41.

   Note: A quick way to customize a schema is to add ClearQuest packages to get the functionality you need. See "Adding packages to a schema" on page 47.
6. Save your work in progress by selecting **File > Save Work**. You can save your work and log off while a schema is checked out to you, then resume work when you log back in. This does not create a new schema version or affect your user database.
7. Test your work by selecting **File > Test Work**. This is a safe and easy way to test your work in the ClearQuest client without affecting your user database. See "Working with a test database" on page 37.
8. Repeat steps 5 through 7 until you are satisfied with your schema.
9. Check the schema into the schema repository to save the new version of the schema. See "Checking in a schema" on page 43.
10. Apply the new schema version to the user database. Select **Database > Upgrade Database** and select the user database. See "Applying schema changes to a user database" on page 44.

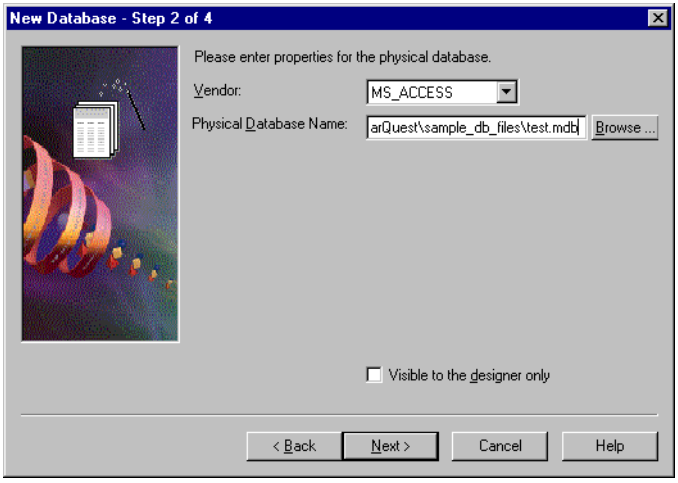
Working with a test database

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

To work with a test database:

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

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



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

2 Check out the schema you want to customize, then select **Database > Set Test Database** and select the test database you associated with this schema.

3 At any time while you are customizing the schema, select **File > Test Work**. ClearQuest saves and validates your schema changes, reporting any errors in the Validation pane at the bottom of the ClearQuest Designer window. See “Validating schema changes” on page 41.

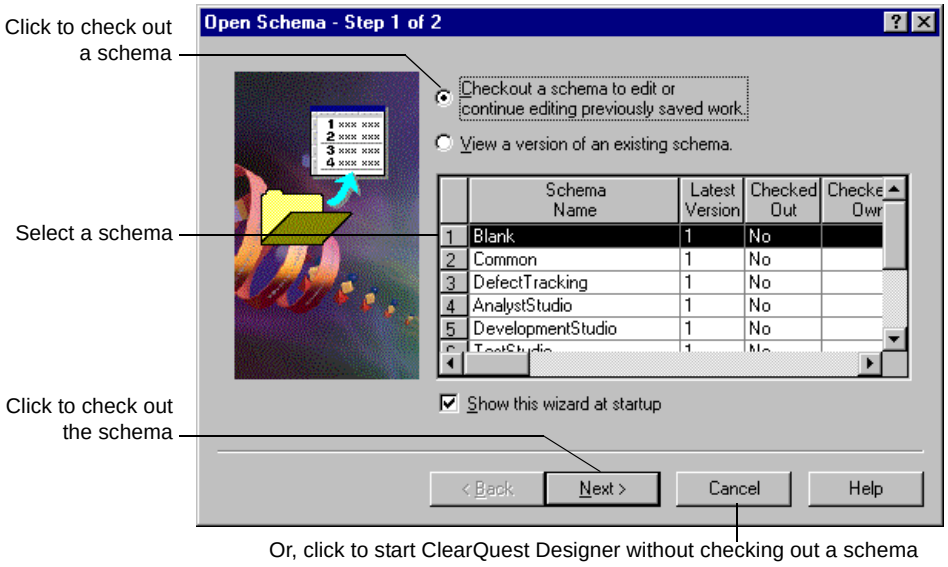
ClearQuest then upgrades your test database with the current schema version. On Windows, ClearQuest automatically starts the ClearQuest client so you can test how the schema performs. To test the schema from the UNIX or Web clients, log into the test database manually. (Remember, the test database will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

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

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

Checking out a schema

ClearQuest maintains schemas under version control in the schema repository. To customize a schema, you must first check out the schema from the schema repository. When you check out a schema, ClearQuest creates a new version of the schema for you to edit. After you log into ClearQuest Designer, the Open Schema dialog box is automatically displayed:



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

You can also check out a schema from within ClearQuest Designer.

To check out a schema from with ClearQuest Designer:

1. Select **File > Open Schema**.
2. Select **Check out a new schema to edit**.
3. Select a schema and version and click **Next**.
4. Enter a comment in the **Comments** text box and click **Finish**.

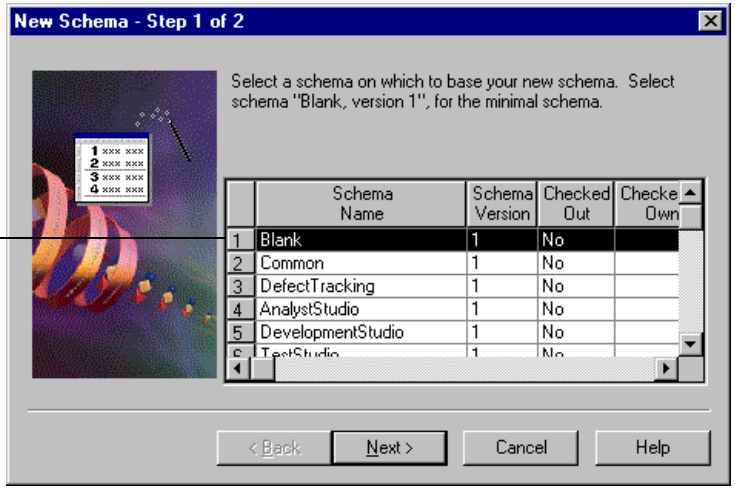
Creating a new schema

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

To create a new schema:

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

2 Select a schema to base the new schema on.



Select a schema to base the new schema on

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

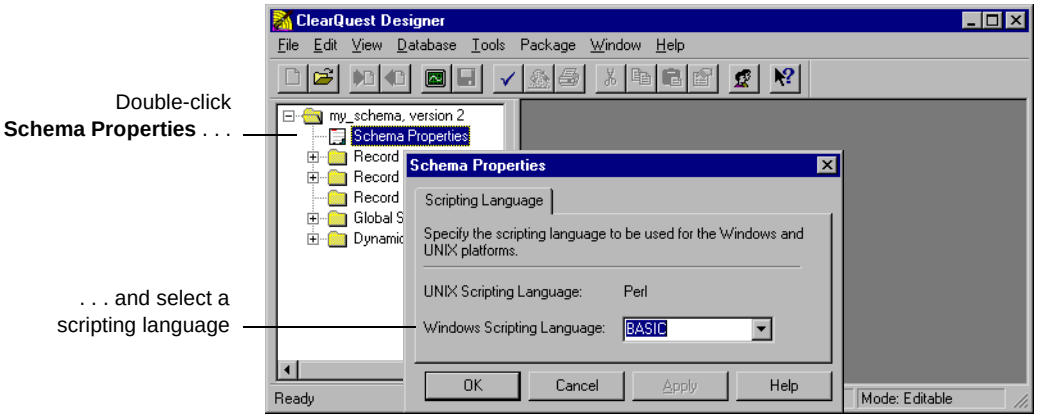
Selecting a scripting language

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

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

To select a scripting language:

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



More information? See Chapter 8, "Using hooks to customize your workflow." For information on the scripting languages themselves, see the following resources on the Internet:

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

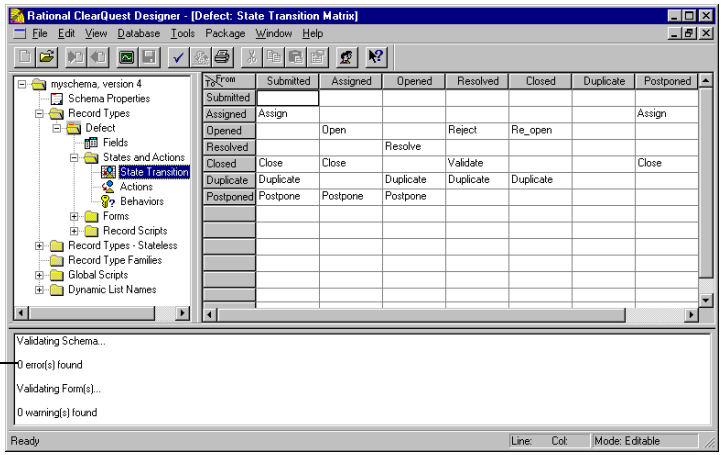
Validating schema changes

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

To validate a schema:

1 Select **File > Validate**.

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



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

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

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

If you cannot validate all of your changes, you can save your schema changes and continue editing at a later time. You can also undo a schema checkout to revert the schema to its previous version. See "Undoing a schema checkout" on page 43.

Note: Validation does not guarantee that a schema will function as desired. Be sure to test a schema with a test database before checking it in, and back up your user database before applying schema changes to it. See "Working with a test database" on page 37.

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

Checking in a schema

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

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

To check in a schema:

1 Select **File > Check In**.

ClearQuest validates the schema, displaying any errors in the Validation pane at the bottom of the ClearQuest Designer window. You cannot check in a schema if there are validation errors. See "Validating schema changes" on page 41.

2 If necessary, correct validation errors.

For help on a validation message, right-click the message and choose **What's This** from the shortcut menu. When the schema passes validation, ClearQuest Designer displays the Checkin dialog box.

3 If necessary, edit the comments in the **Comment** text box to reflect changes that you made to the schema and click **OK**.

Undoing a schema checkout

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

To revert to the last checked-in version of a schema:

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

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

Applying schema changes to a user database

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

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

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

Note: Before applying schema changes to your user database, back up your user database.

To apply schema changes to a user database:

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

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

2. Select a database from the **Database** list and click **Next**.

3 Select a schema version from the **Versions** list and click **Finish**.

More information? Look up *databases, schemas* in the ClearQuest Designer Help index. Also see Chapter 2, "Managing databases."

Viewing a schema

You can open a schema for viewing in ClearQuest Designer, but you cannot make changes to it.

To view a schema:

1 Select **File > Open Schema**.

2 Select View a version of an existing schema, select a schema and version, and click **Finish**.

If you open another schema, ClearQuest closes the schema you are viewing. You can also close a schema manually.

To close a schema that you are viewing:

- Select **File > Close Work**.

Deleting a schema

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

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

To delete a single version of a schema from the schema repository:

- Select **File > Delete Schema Version** and select the schema version to delete.

To delete *all* versions of a schema from the schema repository:

- Select **File > Delete Schema** and select the schema to delete.

Selecting a ClearQuest schema

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

You can add predefined packages to your own customized schema, or use them to enhance ClearQuest predefined schemas. See "Adding packages to a schema" on page 47.

ClearQuest includes the following predefined schemas:

Schema	Description
AnalystStudio	Compatible with Rational Suite AnalystStudio. Contains customization for use with Rational RequisitePro.
Blank	Contains only system fields. Use the Blank schema to create a schema from scratch.
Common	Contains metadata that is common to all of the ClearQuest schemas.
DefectTracking	Contains the fields necessary to start using ClearQuest to track defects in a software-development environment.
DevelopmentStudio	Compatible with Rational Suite DevelopmentStudio. Contains fields and rules that work with Rational's Purify, Quantify, and Pure Coverage.
Enterprise	For use with Rational Suite Enterprise. Contains fields and hooks that work with all Rational products.
TestStudio	Compatible with Rational Suite TestStudio. Contains fields and rules that work with Rational's TeamTest, RequisitePro, Purify, Quantify, and PureCoverage.
UnifiedChangeManagement	Supports the UCM process by providing integration with Rational ClearCase. See "Adding a Rational UCM integration" on page 247.

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

Adding packages to a schema

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

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

For a complete list of packages and the record types and fields they modify, see "ClearQuest packages" on page 210.

Keep in mind the following when adding packages to schemas:

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

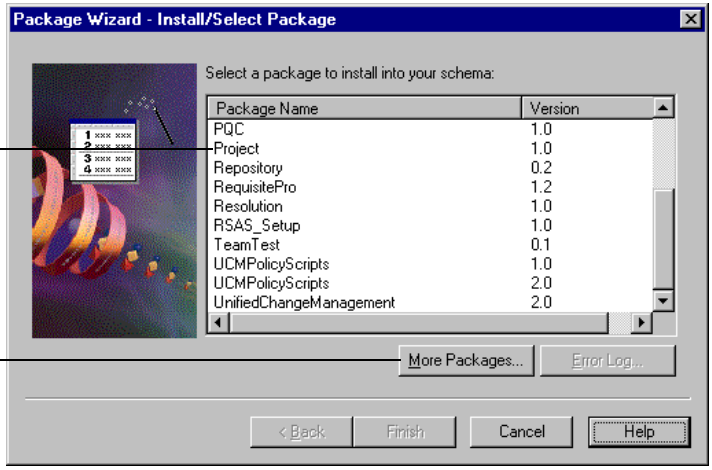
Warning: You cannot create your own packages to use with the Package Wizard. However, you can export schemas or schema versions and then import them into another schema repository using the CQLOAD command line utility. For more information, look up *cqload* in the ClearQuest Designer Help index.

Note: You cannot uninstall or otherwise remove a package from a schema. Be sure to plan carefully before installing a package into a schema.

To add a package to a schema, you use the Package Wizard. The following instructions assume that you are beginning with a schema that is checked in (**File > Check In**) and work closed (**File > Close Work**):

1 Add the package.

Select **Package > Package Wizard** and select a package to add, then click **Next**.



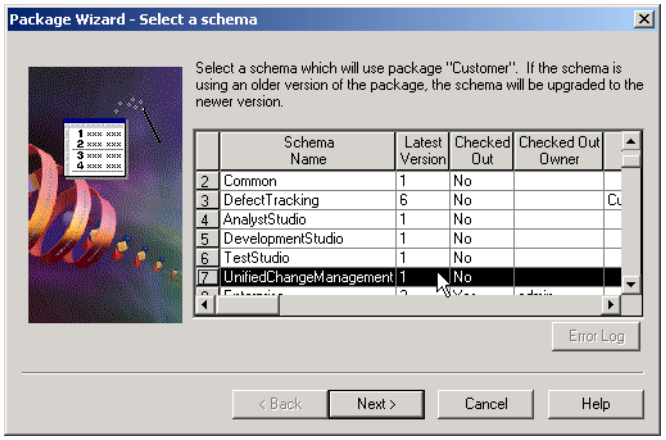
Select a package to install into the schema

If the package you want isn't listed, click **More Packages** and install the package into the schema repository

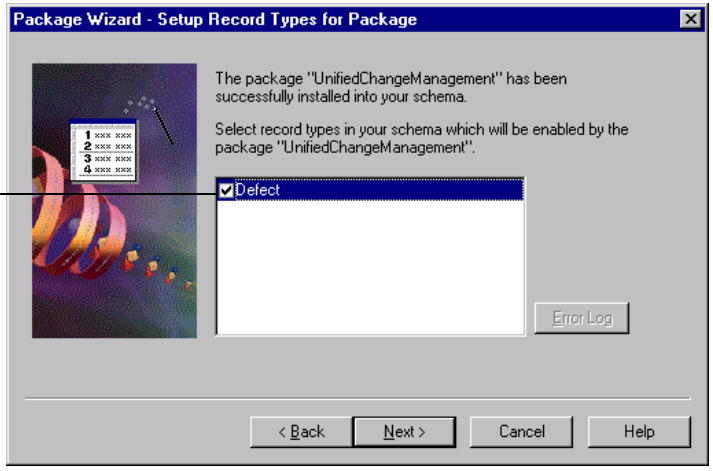
If your schema was not closed (**File > Checkin Schema** or **File > Close Work**), then the Package Wizard will not list the packages that are already part of your schema. It is recommended that you check in your schema or close work for your schema before adding packages to it.

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

2 Select a schema to add the package to, then click **Next**.



3 If the list box of the Setup Record Types for Package dialog box appears, then enable the record types that you want the package applied to.



Select the record type to enable

Some packages modify record types that already exist in your schema. You can select the record types to enable for the package. In other words, the package you are adding can add more fields to existing record types, but you must indicate which record type(s)

you want these fields added to. The changes may also include additional actions, scripts, and so on.

Note: If you decide not to enable this record type, you can do so later. Select **Package > Setup Record Types for Packages** from the main menu. You can also do this to enable this package for record types you add later.

For descriptions of packages, including the record types they modify, see “ClearQuest packages” on page 210.

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

4 If the Setup State Types dialog box appears, specify a state type for each record type in the **Record Type** drop-down list.

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

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

State-type models are listed in “State-type models for packages” on page 222. To learn how to map state types, see “Mapping state types” on page 76.

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

5 If you are installing the UnifiedChangeManagement package, you must also create default actions. See “Using default actions” on page 84.

4 Customizing a schema

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

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

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

Before you can customize a schema, you must check out the schema from the schema repository. For instructions on how to work with schemas, see Chapter 3, “Working with ClearQuest Schemas.”

To customize a schema, you need Schema Designer or Super User privileges. For more information, see Chapter 7, “Administering users.”

Planning schema customizations

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

1 Define your data.

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

- Decide what type of records you need. You may want separate record types for hardware defects and software defects, each of which would have its own set of fields. See “Working with record types” on page 56.
- List the fields you need on each record form. You can use fields to link records and you can customize your record type through field data types and hooks. See “Working with fields” on page 63.

Note: A quick way to add record types and fields to a schema is to add one or more ClearQuest predefined packages to the schema. See “Adding packages to a schema” on page 47.

2 Define your state model.

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

- List the states that you need and the actions that can be performed on the record in each state. Draw a state-model diagram showing how records will move through your system as the result of actions performed on them. For example, in a simple defect tracking system a record might move from the submitted state to opened, to fixed, and then to the verified state. See “Defining your state model” on page 74.
- Create the states and the actions you need. You can customize your workflow by adding action hooks. See “Working with actions and action types” on page 79 and “Customizing actions by adding hooks” on page 83.

- Use field behaviors to associate fields with specific record states. You can define whether the field is mandatory, optional, or read only in each state. See “Defining field behavior” on page 66.

Note: Some ClearQuest packages, such as the Resolution package and the UnifiedChangeManagement package, use state types to define a state's role in your state model. If you add these packages to your schema or add a new state to a schema that uses state types, you must map the new state to a state type. See “Mapping state types” on page 76.

3 Build your record forms.

Create forms for your users to submit new records and to work with records. Decide on the tabs you will need on the forms and where to locate each field. Using form controls, add the fields you've created to the record form. See *Working with forms* on page 87.

4 Add hooks if desired.

- Define the relationships between fields, determining whether the value in one field affects the values in other fields. Understanding these relationships will help you decide how to use hooks. See “Customizing fields by adding hooks” on page 72.
- Plan additional controls or permissions to enhance how your record form is used. For example, decide who should be notified when a record is submitted or changed and who can perform certain actions. See “Customizing actions by adding hooks” on page 83.

More information? See Chapter 8, “Using hooks to customize your workflow.”

Working with record types

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

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

ClearQuest supports state-based and stateless record types.

Note: ClearQuest packages can add and modify record types. For more information, see “ClearQuest packages” on page 210.

State-based record types

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

Note: You can group two or more state-based record types as a record type family to allow ClearQuest users to query across multiple record types. See “Adding a new record type family” on page 59.

Stateless record types

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

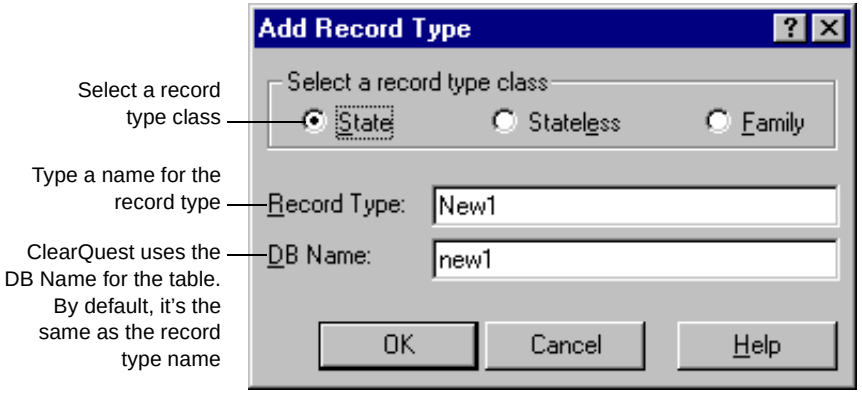
ClearQuest maintains four stateless system record types: History, Attachments, Groups, and Users. You cannot delete system record types.

Adding a new record type

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

To add a new record type:

1. In ClearQuest Designer, select **Edit > Add Record Type/Family.**
2. In the Add Record Type dialog box, select the class for the record type (**State** or **Stateless**).
3. In the **Record Type** text box, type a name for the record type.



4. If you are adding a stateless record type, you must select one or more of its fields to be the unique key. See "Selecting a unique key for a stateless record type" on page 58.

Note: You can use ClearQuest predefined packages to add record types to a schema; packages can add record types and enhance existing record types by adding fields, hooks, forms, and states. See "Adding packages to a schema" on page 47.

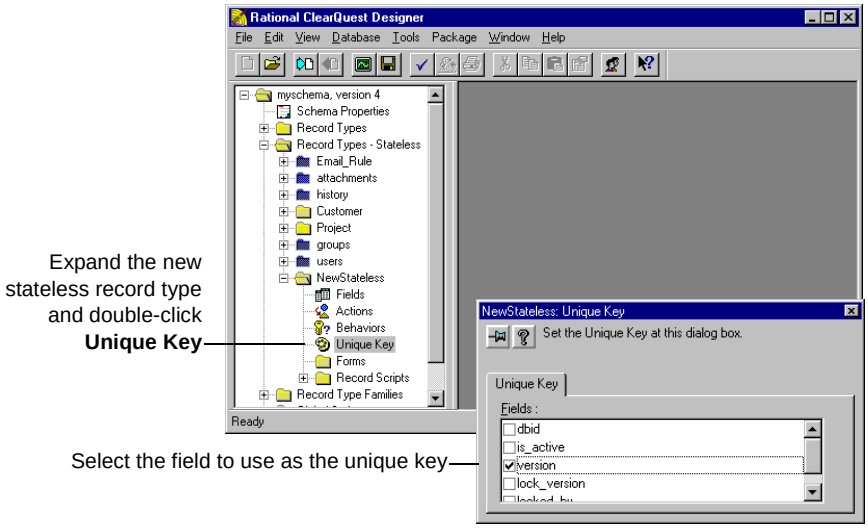
Selecting a unique key for a stateless record type

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

To select the unique key for a stateless record type:

1 In the Workspace, expand the stateless record type and double-click **Unique Key**.

2 In the Unique Key dialog box, select the field or fields to be used as the unique key.



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

Selecting a default record type

Each schema must have a default record type. Default record types can be state-based or stateless. ClearQuest uses the default record type to create a shortcut button in the ClearQuest client that can be used for submitting records of that type. ClearQuest

also uses the default record type in situations where no other record type is explicitly specified.

To make a record type the default:

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

Adding a new record type family

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

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

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

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

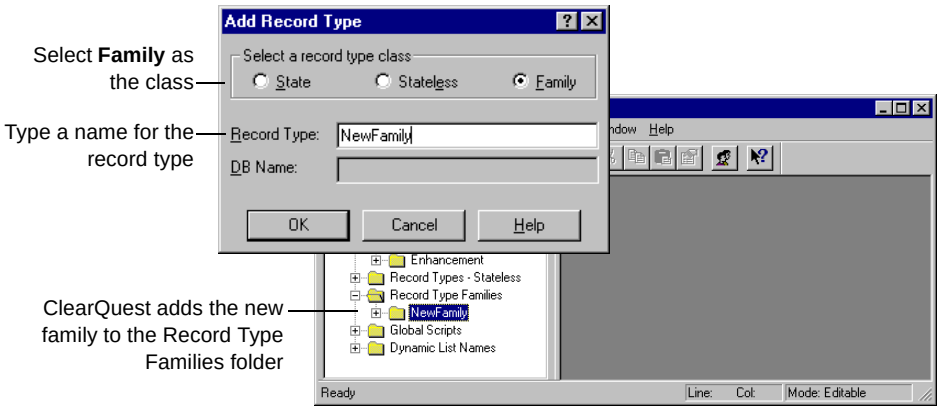
  To learn how to add fields to a record type and to define their data type, see "Working with fields" on page 63.

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

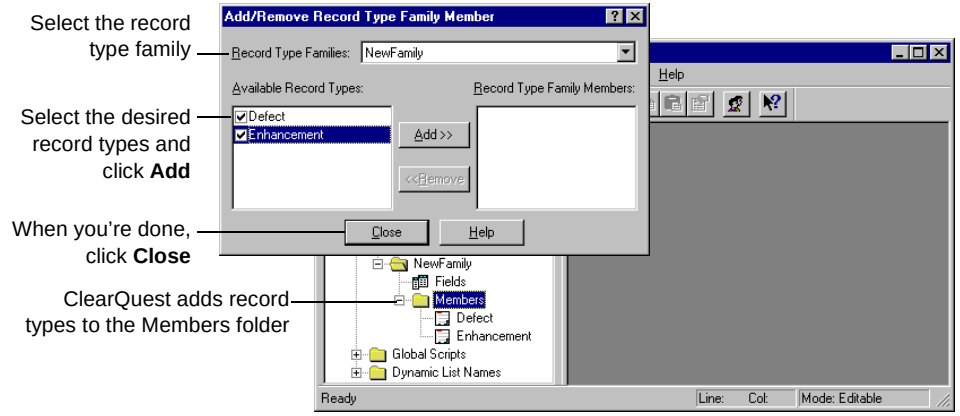
To add a new record type family to a schema:

1. Select **Edit > Add Record Type/Family.**
2. In the Add Record Type dialog box, select **Family.**

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



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



5 Add common fields to the member record types to allow ClearQuest users to query the new family. See "Working with fields" on page 63.

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

Renaming a record type

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.

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

To rename a record type or family:

1 Select **Edit > Rename Record Type/Family.**

2 Select the class of the record type that you want to rename: **State**, **Stateless**, or **Family.**

3 Select the record type from the **Current** drop-down list.

4 Type a new name in the **New Name** text box.

Within a schema, each record type and record type family name must be unique.

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

Deleting a record type or family

You cannot delete the following record types:

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

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

To delete a record type or family:

1 Select **Edit > Delete Record Type/Family.**

2 Select the class of the record type to delete: **State**, **Stateless**, or **Family**.

3 Select the record type or family from the **Record Type** list.

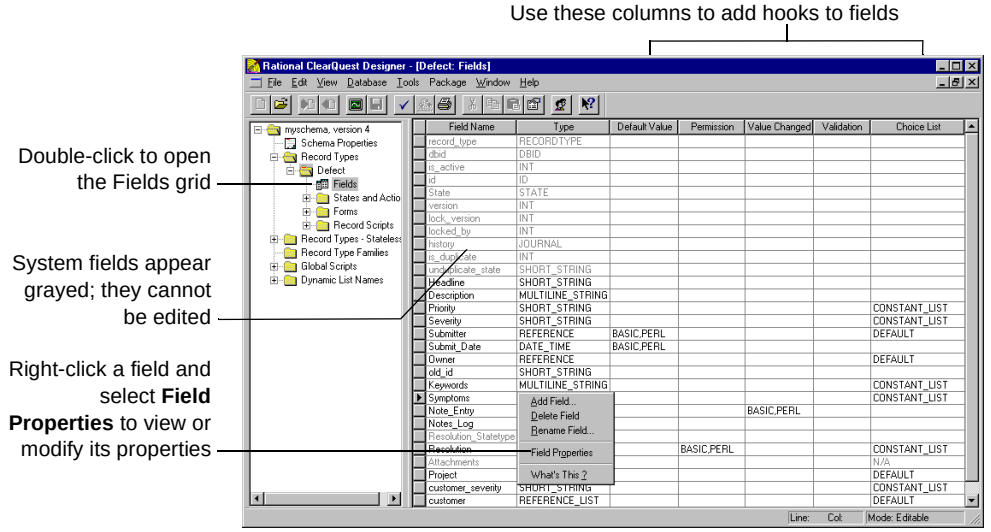
Note: You can also right-click the record type or family in the Workspace and select **Delete** from the shortcut menu.

Working with fields

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

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

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



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

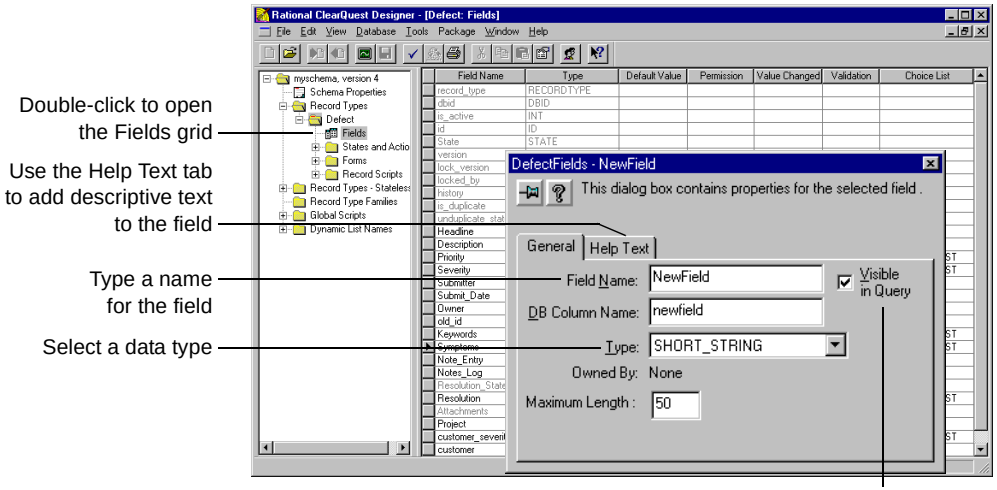
Each ClearQuest record type includes system fields. These fields are required for every record of that type. System fields appear grayed in the Fields grid.

Adding a new field

To add a new field:

1 Display the Fields grid.

2 Select **Edit > Add Field**.



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

Note: To make a new field available to users, you must add the field to the record form. See "Adding controls to forms" on page 96.

Adding Help text to the field

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

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

Selecting a field data type

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

ClearQuest supports the following field data types:

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 in conjunction 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.
SHORT_STRING	A variable-length character string with a 254-character maximum. You set the length in the Properties dialog box when defining the field. Enter a value between 1 and 254 in the Maximum Length field.

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

Defining field behavior

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

ClearQuest supports the following field behaviors:

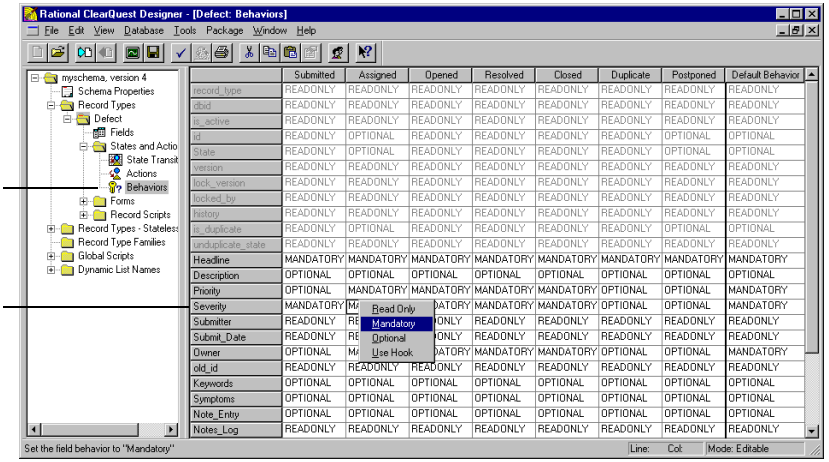
Behavior	Description
Mandatory	The user is required to enter a value in this field before applying the changes to a record. Failure to do so will result in a runtime validation error. Mandatory fields show up in red on the record form.
Optional	The user can enter data into this field but is not required to do so. **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, use the Behaviors grid.

To define the behavior for a field:

1. In the Workspace, expand **Record Types.**
2. Expand the record type and **States and Actions**.

3 Double-click **Behaviors**.



4 Right-click the cell you want to modify and select the behavior.

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

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

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

Changing the name of a field

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

To change the name of a field:

1 Display the Fields grid.

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

3 Type a new name for the field.

Deleting a field

The following restrictions apply to deleting fields:

- If you delete a field, you must also delete the field from the record form. See “Deleting controls from forms” on page 99.
- You cannot delete, rename, or modify a system field. System fields appear dimmed in the Fields grid.
- When you delete a field, ClearQuest retains the DB Column Name it generated for the field (by default, the same as the field name). If you later reuse the same name to create a new field, ClearQuest will generate a unique DB Column Name.
- If you explicitly refer to the name of a field in script code, you need to remove any references to the field from your script.
- In the ClearQuest client, any queries that use this field will become invalid.

To delete a field:

1 Display the Fields grid.

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

Using fields to link records

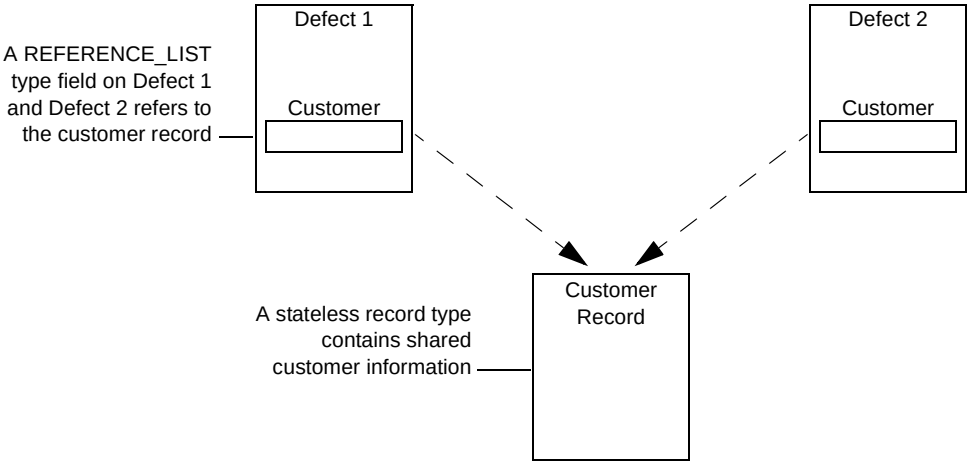
You can use fields to link records of the same type or records of different types. You can link records to:

- Share common data
- Create a parent-child hierarchy

Linking records to share common data

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

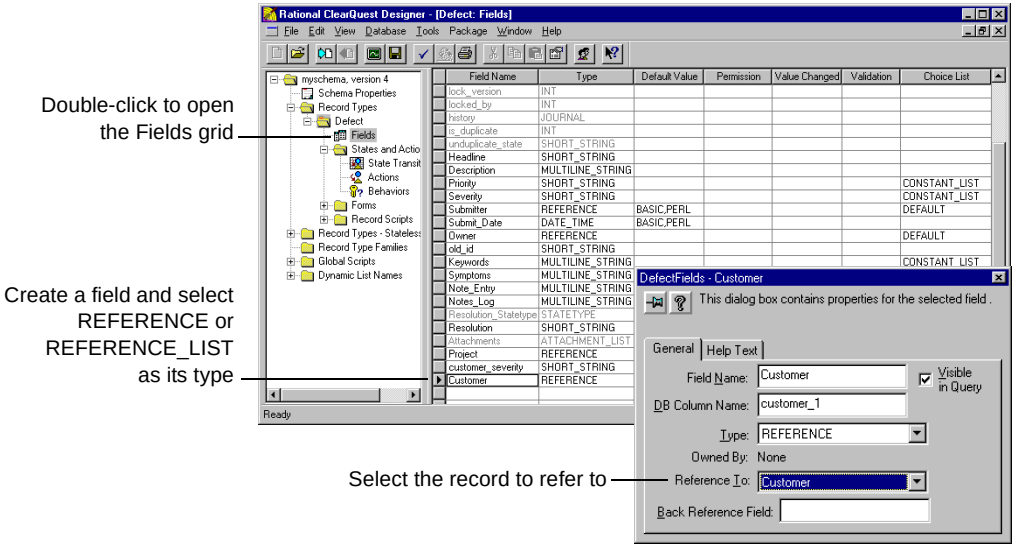
For example, you might have the same customer data that must be entered for multiple records.



To link records to share common data:

1. Display the Fields grid for the record type and create a new field, selecting REFERENCE or REFERENCE_LIST as its type.
2. Right-click the field and select **Field Properties** from the shortcut menu.

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

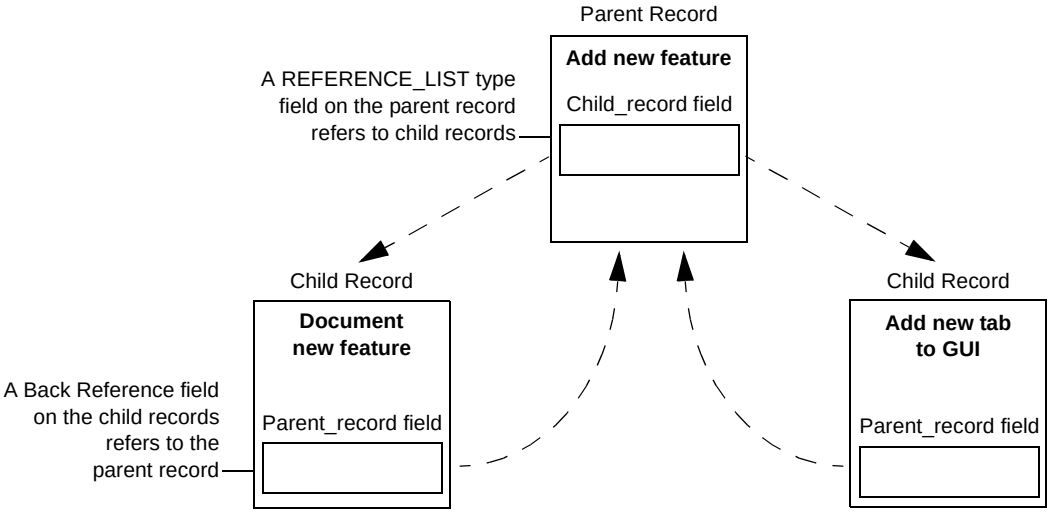


4 To make the field available to users, you must add it to the record form. Use a parent/child control with a REFERENCE_LIST field type. See “Adding controls to forms” on page 96.

Linking records to create a parent/child hierarchy

You can use REFERENCE or REFERENCE_LIST type fields to link records of the same type to create a parent/child hierarchy. For example, you can relate a parent record that requests the addition of a new feature to one or more child records that describe related

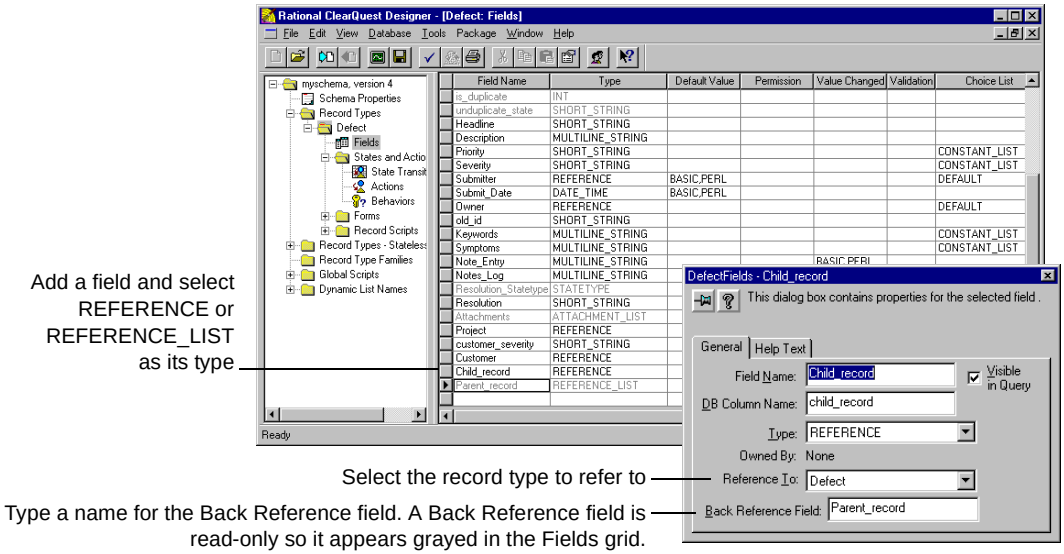
tasks, such as documenting the new feature and adding a new tab to the interface.



To link records to create a parent/child hierarchy:

1. Add a REFERENCE_LIST or REFERENCE type field to the parent record.
2. Right-click the field and select **Field Properties** from the shortcut menu.
3. Enter a name for a Back Reference field. A Back Reference field is a read-only field that makes traversing parent-child relationships easier by allowing you to view the link from the child record point

of view. This field is automatically added to the field list of the child record.



4 Add the new fields to the record forms. Use a parent/child control with a REFERENCE_LIST field type. See “Adding controls to forms” on page 96.

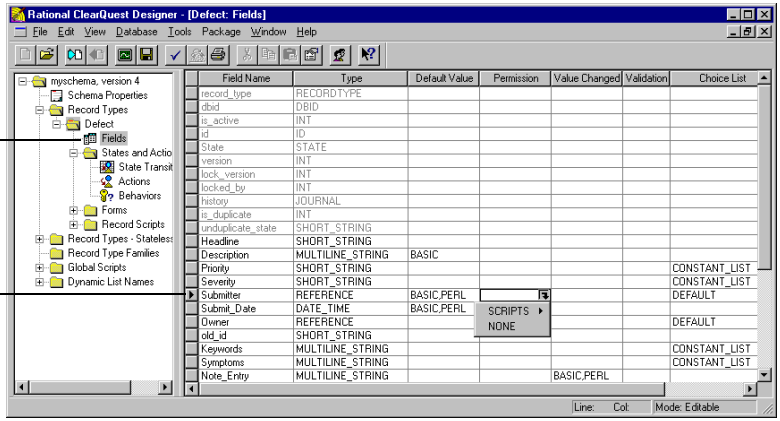
More information? Look up *records, linking related* in ClearQuest Designer Help index. See also “Action hook for setting the value of a parent record” on page 170 of this guide.

Customizing fields by adding hooks

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

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

To define a field hook, use the Fields grid.



To create dependent fields, use a Value Changed hook (see “Creating a dependency between fields” on page 147). To find out how to create a choice list for a field, see “Working with choice lists” on page 149.

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

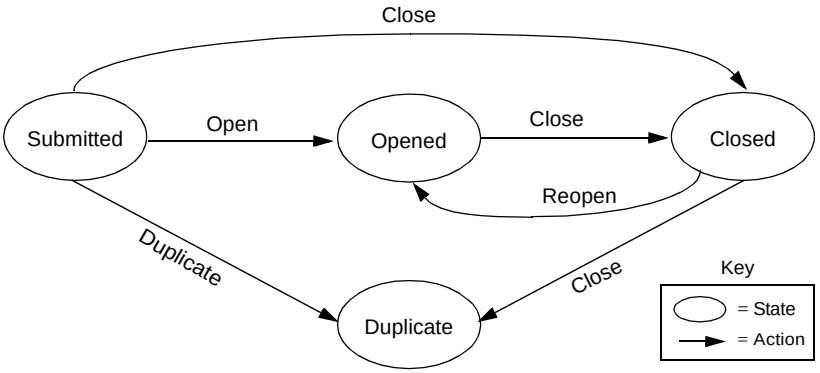
More information? For a complete description of field hooks and information about how they work with action hooks, see Chapter 8, “Using hooks to customize your workflow.” Also see “Selecting a scripting language” on page 40. Look up *field hooks* in the ClearQuest Designer Help index.

Defining your state model

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

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

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



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

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

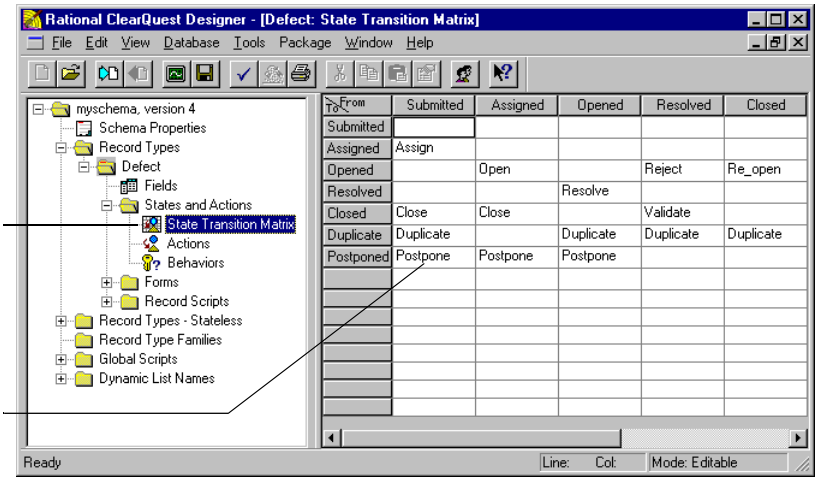
Using the State Transition Matrix

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

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

Double-click to open the State Transition Matrix

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



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

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

Adding a new state

To add a new state:

1. Display the State Transition Matrix.
2. Select **Edit > Add State**.
3. Type a name for the state and click **OK**.

   ClearQuest automatically adds the new state to the row and column headers in the State Transition Matrix.

   Once you create a new state, you must also create a state transition that defines that state's place in the state model. See "Creating a state transition" on page 82.

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

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

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

Mapping state types

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

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

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

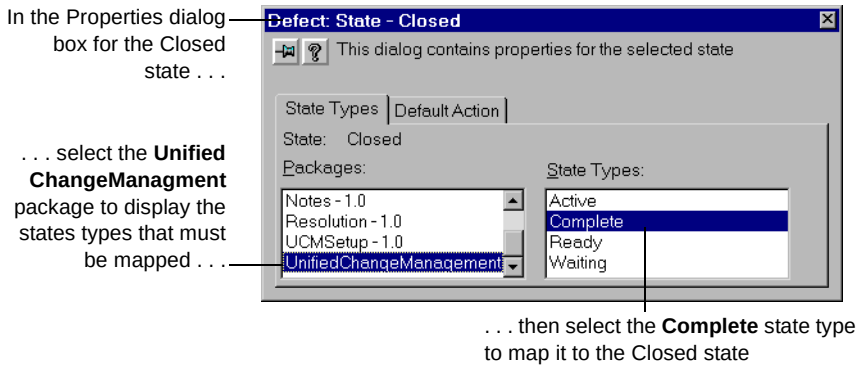
To map state types:

1 Display the State Transition Matrix for the desired record type.

2 Right-click a state and select **Properties** to display the Properties dialog box for the state.

3 In the **State Types** tab, select a package that requires state type mapping, then select a state type.

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



4 Close the dialog box to save the state mapping.

5 Repeat this process for each new state in the record type.

Note: If you are using the UCM schema or package, you must also assign default actions for your states; see "Step 3: Set the default actions for UCM" on page 250.

Changing the name of a state

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

To rename a state:

1 Display the State Transition Matrix.

2 Select the row of the state that you want to rename.

3 Select **Edit > Rename State** (or right-click and select **Rename State** from the shortcut menu) to open the Rename State dialog box.

4 Type a name in the **New Name** text box.

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 Open the State Transition Matrix.

2 Select the row of the state that you want to rename.

3 Select **Edit > Delete State** (or right-click and select **Delete State** from the shortcut menu).

Working with actions and action types

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

ClearQuest supports the following types of actions:

Action type	Description/Comments
Base	A Base action is a secondary action that runs as a side-effect of every other action. Base actions allow you to write an action hook only once, but use it with multiple actions. Each time an action fires, the Base action checks to see if the hook criteria is met; if it is, the base action completes its process. 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	Allows users to delete a record from the database. Delete actions appear in the list of actions in the ClearQuest client.
Duplicate	Duplicate actions are 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.

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

Adding and modifying actions

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

To display the Actions grid:

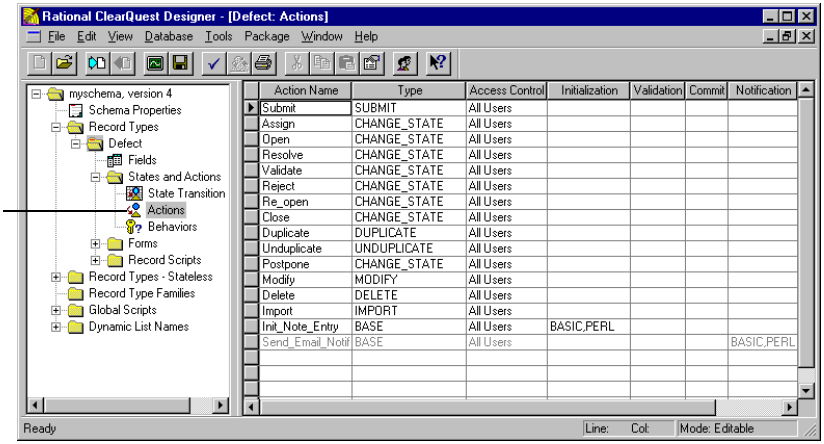
1 In the Workspace expand **Record Types** or **Recrod Types - Stateless**.

2 Expand the desired record type.

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

Double-click to open the Actions grid

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

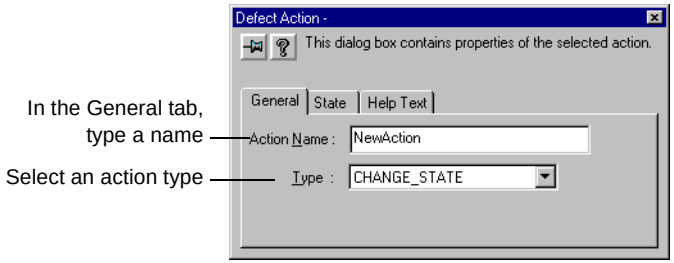


Adding a new action

To add a new action:

1 Display the Actions grid.

2 Select **Edit > Add Action**.



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

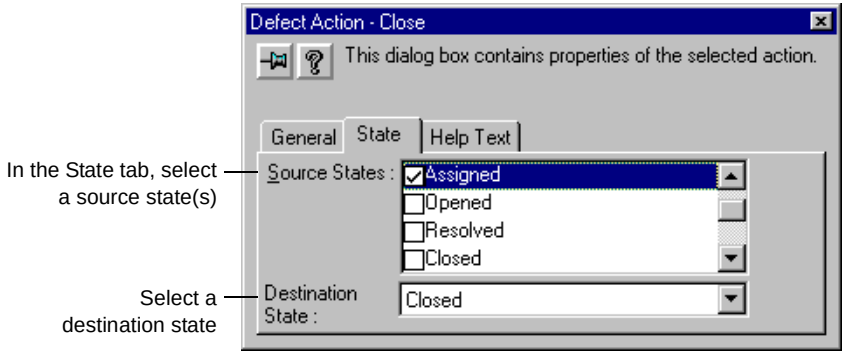
Creating a state transition

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

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

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

2 In the State tab of the Action Properties dialog 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.

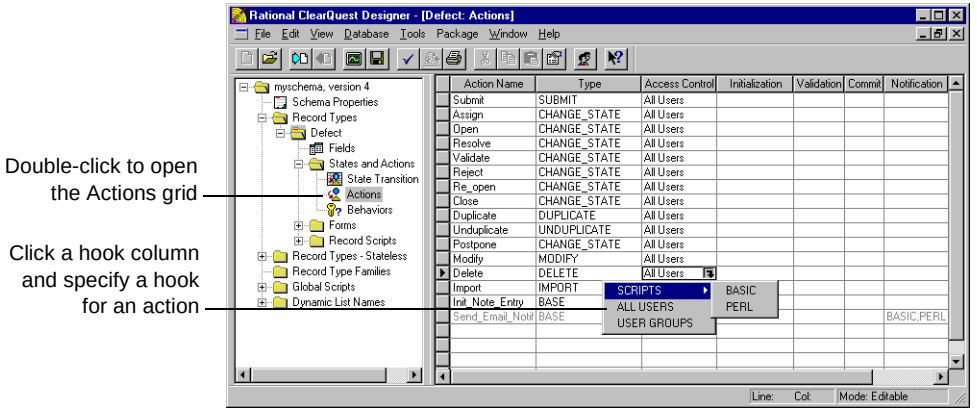
3 After creating the state transition, dispaly the State Transition Matrix to verify that ClearQuest applied the transitions to the states. See “Using the State Transition Matrix” on page 75.

Customizing actions by adding hooks

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

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

To define an action hook, use the Actions grid.



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

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

an access control action hook, see “Controlling user access to actions” on page 135. Also see “Selecting a scripting language” on page 40.

Using default actions

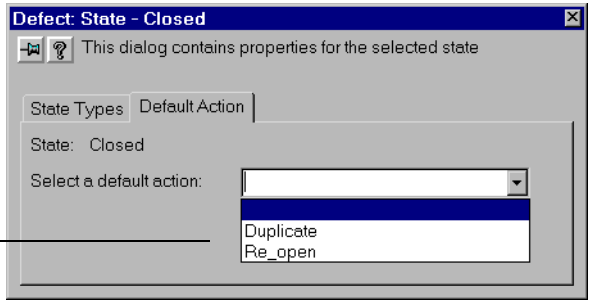
You can define default actions for states. A default action for a state appears in bold in the ClearQuest client **Actions** menu. Associating a state with a default action:

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

Note: If you use the UCM schema or package, the default actions of your states must provide a valid path through the state type model. For more information, see “Step 3: Set the default actions for UCM” on page 250.

To select a default action for a state:

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



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

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

5 Working with forms

Each record type can have two forms associated with it: a record form and a submit form. A record type must have a record form, but a submit form is not required. A submit form allows users to use a different form to submit records of that type. When a user first submits a new record, ClearQuest displays the submit form. Later, when the user views and works with the record, ClearQuest uses the record form. If the record type does not have a submit form, ClearQuest uses the record form both for submitting records and for working with them.

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

- Working in the Forms window
- Creating and modifying forms
- Creating and modifying dialog tabs in forms
- Adding form controls to dialog tabs
- Reusing forms
- Creating forms to view and modify imported data

Working in the Forms window

You create and customize forms in the Forms window.

To display the Forms window:

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

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

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

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

Working with forms

You must define at least one Record form for each record type in a schema. Every form, whether a Record form or a Submit form, has at least one tab; you can add additional tabs.

Creating forms

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

To create a form:

1. In the Workspace, right-click the **Forms** folder and select **Add** from the shortcut menu.

   ClearQuest displays a new empty form in the right pane and highlights the name of the form in the Workspace so that you can change it.

2. In the Workspace, type a name for the form.

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

3. In the Workspace, right-click the form name and choose **Record Form** or **Submit Form**.

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

Renaming forms

To rename a form:

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

Changing the size of forms

You can change the size of a form to optimize the space used by the controls. If a form is not large enough to display all controls, we recommend that you add additional tabs to display the additional controls. Changing the size and position of the form also changes the size and position of the form's dialog tabs and default buttons.

To change the size of a form:

- In the Forms window, do either of the following:
  - Drag a corner or side of the form.
  - To set the width and height of the form to specific values, right-click in a tab and select **Form Properties** from the shortcut menu. In the Form Property Sheet, enter width and height values.

Changing the font of forms

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

To view your changes to the font, you must close and then re-open the form.

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

1. In the Workspace, right-click the form and select **Fonts** from the shortcut menu.
2. In the Font dialog box, select the new font and font properties.

3 Close the form and re-open it to display the new font.

Deleting forms

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:

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

Adding tabs to a form

If your form has more controls than can be easily displayed on the main tab, you can add additional tabs and move some controls to the new tabs. Tabs allow you to organize controls into groups. You can add as many tabs as needed for a form.

To add a tab:

- Display the form and select **Edit > Add Tab**.

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

Renaming tabs

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 rename a tab:

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

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

Restricting access to a tab

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

To restrict access to a tab:

1. Right-click the tab and choose **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

Tabs are displayed in the order in which you add them, but you can change the order by assigning a new index number to a tab. (An index number identifies the order of the tab; the first tab has index number 0.) You must close and re-open the form to see your changes.

To change the order of tabs:

1 Open the form and click the tab whose order you want to change.

2 Double-click the tab, or right-click the tab and choose **Tab Properties** from the shortcut menu.

3 In the Tab Properties dialog box, select a new index number (0 indicates first tab) for the display order of the tab.

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

The tab you selected moves to the new position. You must change the index number for tabs individually, and close and re-open the form to see your changes.

Deleting tabs

If you no longer need a tab, you can delete it from your form. Deleting a tab also deletes all controls on that tab.

Warning: Deleting a tab cannot be undone.

To delete a tab:

- Click the tab and select **Edit > Delete Tab**.

Copying tabs

You can copy a tab, including all controls, and paste it into another tab in the same or different form. Copying and pasting a tab is a quick way to create a form if it is similar to an existing form. After pasting the tab into a form, you can modify it.

To copy a tab:

1. Display the tab that you want to copy.
2. Choose **Edit > Copy Entire Dialog to Clipboard**.

To paste the tab:

1. Display the tab to contain the pasted controls.
2. Choose **Edit > Paste**, or right-click in the tab and select **Paste** from the shortcut menu.

Working with form controls

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

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

ClearQuest supports the following form controls:

Form control	Description
ActiveX	Incorporates any registered ActiveX control into a form.
Attachment	Displays a list of attached files and includes a set of controls that allow users to add, remove, or view attached files.
Check Box	A two-value control you can use for Boolean values or any field that has only two values. To specify the two values, right-click the control on the form and select 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	A control used to visually group one or more other controls.
History	Displays the actions that have been applied to a record.
List Box	Displays a list of possible values for a particular field.

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

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

Adding controls to forms

Before you can add a field to a form, you must create the field in the Fields grid. For details, see "Adding a new field" on page 64.

You can add controls to a form using the Control palette, Field List, or Form Controls menu. The Form Controls menu allows you to create the same controls as the Control palette; in addition, it contains commands to create ActiveX and Parent/Child controls. The Field List creates a control that is appropriate for the type of field you choose.

After adding a control, you cannot change its type.

Adding a control using the Control palette or Form Controls menu

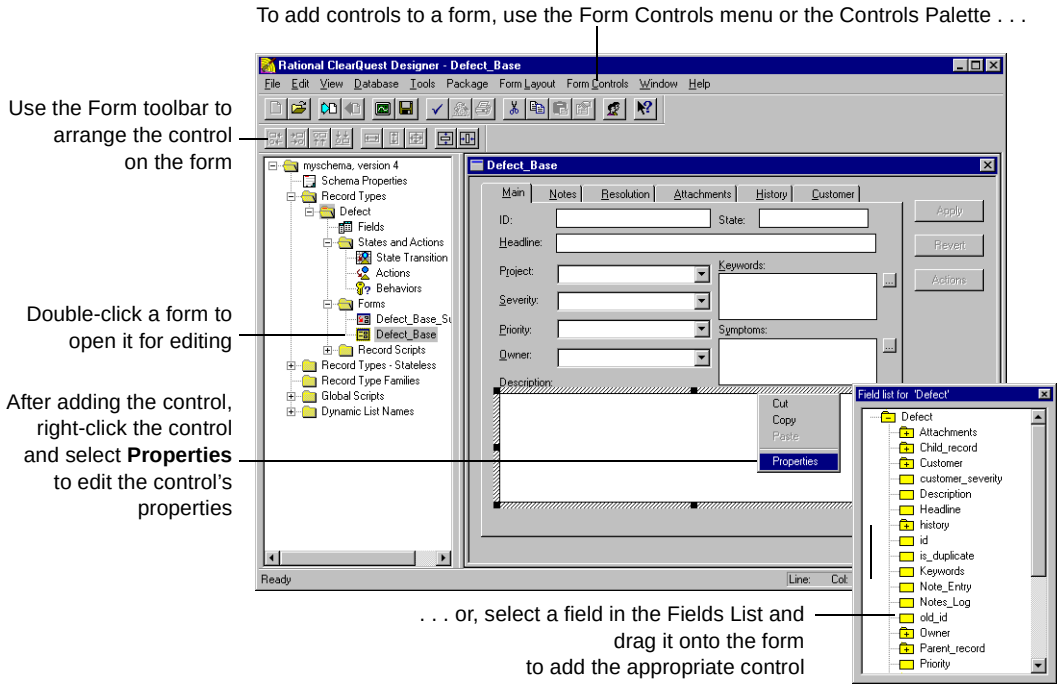
To add a control using the Control palette or Form Controls menu:

1 Display form.

2 Click the control you want to add in the Control palette, or select a command from the Form Controls menu.

3 If the Control palette is not visible, choose **View > Control Palette**.

4 Click where you want to place the control and drag until the control is the desired size.



Adding a control using the Field List

If you add a control using the Field List, ClearQuest creates a text box for most simple data types. If you add a control using the Field List, ClearQuest adds the following control.

Field type	Default control
Attachment List	Attachment
Constant List Choice List	Drop-down List Box

Field type	Default control
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 Sting Constant List	Drop-down List Box
Short Sting	Text Box

To add a control using the Field List:

1 If the Field List is not visible, choose **View > Field List**.

2 Drag a field from the Field List to a tab.

Selecting controls in forms

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.

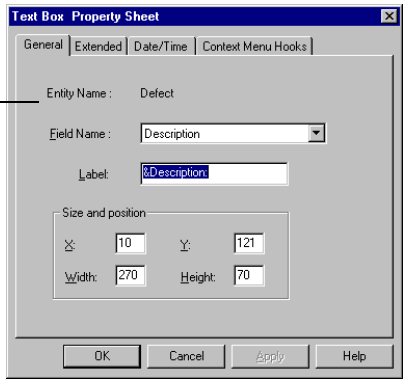
3 To select additional controls, hold down the Shift key as you click.

Editing field control properties

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

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

Use the Property Sheet to change the properties of a control



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

More information? For detailed descriptions on the properties for each control type, select **Working with record forms > Form Control Reference** in the ClearQuest Designer Help.

Deleting controls from forms

If you no longer need a control, you can delete it from the form. Deleting a control removes the control and its label from the form but does not remove the associated field from the schema. If you no longer need a field, you must delete the field from the Fields grid.

If you delete a field from a schema, ClearQuest does not delete controls associated with the field. You must delete the controls manually.

To delete a control:

1 In the Forms window, click the tab containing the control you want to delete.

2 Select the control or controls to be deleted.

3 Select **Edit > Delete**, or press the Delete key.

Changing the size and location of controls on a form

ClearQuest provides several tools to assist you in arranging controls on a form. You can select a control and drag it to a new location or 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 and provides additional tools.)

Some tools (such as the Align and Same Size tools) use the first control you select as the basis for modifying the other controls.

The following table lists each tool and its function.

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	Align right Aligns the right edge of each control with the right edge of the control that was selected first.
Align top	Aligns the top edge of each control with the top edge of the control that was selected first.
Align bottom	Aligns the bottom edge of each control with the bottom edge of the control that was selected first.
Align vertical centers	Aligns the vertical center of each control with the vertical center of the control that was selected first.
Align horizontal centers	Aligns the horizontal center of each control with the horizontal center of the control that was selected first.
Space evenly across	Arranges the selected controls horizontally so that there is an equal amount of space between all controls. The left-most and right-most controls are not moved.
Space evenly down	Arranges the selected controls vertically so that there is an equal amount of space between all controls. The top-most and bottom-most controls are not moved.
Center vertically in dialog	Adjusts the horizontal center of each control so that it matches the horizontal center of the tab.

Layout Tool	Description
Center horizontally in dialog	Adjusts the horizontal center of each control so that it matches 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.

Moving controls

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

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

Aligning controls

You can align controls to one another.

To align controls:

1. Select a control.

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

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

Resizing controls

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

To resize an individual control:

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

To make controls the same size:

1. Select a control.

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

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

Changing the tab order of controls

By default, the tab order of controls is the order in which they are added to the form. (The tab order determines which control receives focus when a user press the Tab key. When a user presses Tab, the focus moves to the next control in the tab order.) 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 Form window, click the tab.
2. Select **Form Layout > Set Tab Order**.

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

3. Click the control you want to be first in the tab order.
4. Click the remaining controls in the order you want them to receive focus.

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

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

Creating forms for multiple platforms

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

Also, some hooks work differently in ClearQuest Web client. See "Using hooks in ClearQuest Web" on page 178 for information on using hooks on the Web client.

More information? See *Administering ClearQuest Web* on page 175.

Creating forms to view and modify imported data

If you are updating a schema to accommodate imported records, you can modify a 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 at least one field to display the Old ID of the imported record. The Old ID field is not required for new records but is useful for identifying imported records using information from the old database.

Reusing forms

If you have a form that is common to multiple schemas, you can create the form, export it, import it into other schemas, and then use it as is or modify it. You must import the form into a record type with a name identical to the record type that exported the form.

Exporting a form

Exporting a form saves all tabs, all controls on all tabs, and all form, tab, and font properties. But, if you want to use just a single tab in another form, you can. See “Copying tabs” on page 94.

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

To export a form:

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

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

Importing a form

You must import the form into a record type with the same name as the record type from which you exported the form. You cannot export a form and then import it into a different record type within the same schema.

When importing a form, ClearQuest maps the fields associated with the form to the fields of the selected record type. ClearQuest maps fields based on their name and does not verify that the fields

have matching types. If a field in the form does not correspond to a field in the selected record type, you must either reassign the corresponding control to a valid field or delete the control from the form.

To import a form:

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

6 Using security in ClearQuest

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

These security features are particularly valuable when more than one group has access to the same database; if, for example, you are alllowing more than one group to submit defects to the same database; or, when multiple projects share the same database. You might open your production database to a variety of customers so they can all submit defects and enhancement requests to your database; however, you don't want your customers to see any records that they are not entitled to see. You don't want customers at one company to see records submitted by customers at a different company or records submitted by your internal users.

This chapter covers the following topics:

- Hiding records in ClearQuest
- Steps required for record hiding
- Security use case example

The security example presented in this chapter illustrates the steps required to set up security in ClearQuest.

Hiding records in ClearQuest

The following are the main elements used to hide records in ClearQuest:

- The Controlled Record Type — The record type you want to control (for example, the Defects record type).
- The Security Context Field — The reference type field in the controlled record that references another record containing user group information.
- The Security Context Record Type — The record type referenced by the security context field that contains data about users (for example, the Project or Customers record types).
- User Groups — The groups to whom you will grant privileges.

Record hiding in ClearQuest is accomplished by placing a security field in the record type of the records you want to control. This field references another record containing data that determines which users can see or change the controlled record. The record containing this data is of the *security context record type*.

Simply put, the security context field in the controlled record type references the security context record type that contains the groups with privileges to see or change the controlled record. This allows you to hide specific records in the controlled record type, i.e., the record you want to control. Only users who are in the group list of the security context record will be able to see the controlled record.

The record you choose to be the security context record type is simply the record that is referenced by the security context field. This record is the container for data that determines which user groups have access to the controlled record.

The security context field links the controlled record to the security context record. For example, in order to control which customers are allowed to see defects, you would place a security context field in the Defects record and reference this field to the Customer record. The record type can be either state-based or

stateless. You would assign user groups to each customer record and grant these groups privileges to see certain records in the Defects record type.

Designing a security system

The ClearQuest security features provide you with a mechanism for restricting user access to records in your database based on membership to user groups.

Before you impose security on a record type, you will have to decide the following three things:

- Decide which record types you want to impose security on.
- Organize user groups according to the security levels or context you want to use. This is also referred to as the security context.
- Define an existing record type that will be used to define the security context.

1. Decide which record types you want to impose security on

Say you want to restrict access to defects in your ClearQuest system. In this case, you will want to impose security on the Defect record type. This is referred to as your security record type.

2. Organize user groups according to the security levels or context you want to use

Organize user groups according to the security levels or context you want to use. This is also referred to as the security context.

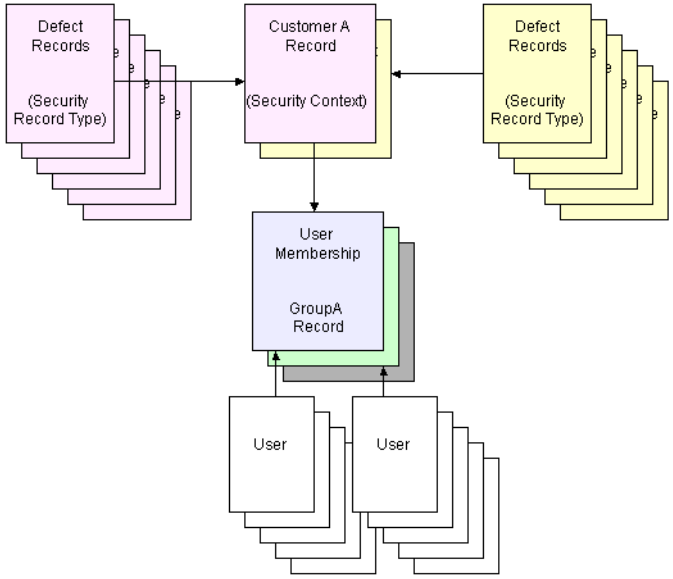
For instance, you may wish to limit access to users based on the company they work for, the project they work on, or the department they report to. Let's say you want to limit access based on the company they work for, this way you can provide your customers access to your database, but control which defects they can access.

3. Define an existing record type that will be used to define the security context

Define an existing record type that will be used to define the security context. The Security Context record type will be used as a pointer or reference to the group information needed to enforce security.

For example, if you organized your groups by customer, you should define a Customer record type to be the Security Context record type. When a defect is associated with or references a particular customer record, ClearQuest allows only those users associated with that customer to see the record.

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



If you implement the design shown above in ClearQuest, this is what the end user will experience: User A logs into ClearQuest.

He is a member of Group A, which is associated with Customer A. When user A runs a query or requests a record, ClearQuest will automatically filter all the defects based the security context “Customer A”. Only defects belonging to Customer A will be presented. When user A submits a record, ClearQuest will automatically determine what group the user has membership to and assign the Customer ownership accordingly.

Implementing security

In order to impose security on your systems you must:

- Create the user groups that align to your user access privileges. If you are basing your security access on specific customers, you would want to create customer groups, one for each different access permission. For this design you would create Group A, Group B, and Group X.
- Assign users to the groups.
- Create a security field in the record type you are controlling (in this case the Defect Record Type) and reference it to your Security Context record type, in this case the Customer Record Type.
- Update your forms to include the new reference field.
- Enable security on the security field. When you enable security, ClearQuest automatically creates a new tab in the forms for the referenced record type. In this case, ClearQuest creates a new tab in the forms for the Customer record type.
- Submit records for each security context. If your security context is Customer, then you will need to submit a record for each customer, one record for Company A, one for B, and so on.
- For each record you submitted above, designate the user groups that can access the customer record through the new tab created by ClearQuest when you enable security.
- Edit your existing security controlled records—in this case Defects—and select the applicable security context record, in this case the Customer record.

Security example

This example illustrates how you could open a database to multiple customers and set up security so that each customer is prevented from seeing data from the other customers. This example shows the steps used to hide records in ClearQuest.

In this example, you want to open your production database to your customers to enable them to do the following:

- Submit defects
- Check the status of their defects by either running a predefined query or by creating a new query
- Add more information to their existing defects, if needed

You do not want them to see defects submitted by other customers.

For this example, you have various customers such as Logic Equipment, Widgets Inc., Modern Software, and so on. When Modern Software logs in to your database, you do not want them to see defects filed by Widgets Inc., or by your own QE team. For example, Modern Software can create any query they want but the only information that will be presented to them in the results grid (or charts or reports) is the information related specifically to them, that is, defects submitted by them.

The steps required to complete this example are listed below. You must have Super User privileges to complete all of these steps.

Note: You must have Schema Designer privileges to complete steps 1 through 4.

1. Check out a schema to edit.
2. Create a security context field in the record you want to control. This field references the record containing the user data (the security context record). You may also use an existing field as the security field as long as it is a reference type field.
3. Place the field on a form so that you can access it later.
4. Check in the schema and upgrade the user database.

Note: You must have User Administrator privileges to complete steps 5 and 6.

5 Create the groups you want to view the controlled record and add members to the groups. You may also use existing groups.

6 In the ClearQuest client, submit a record for each security context record.

Note: You must have Security Administrator privileges to complete steps 7 and 8.

7 For each security context record, in the Ratl_Security tab, add the group or groups that can view that record.

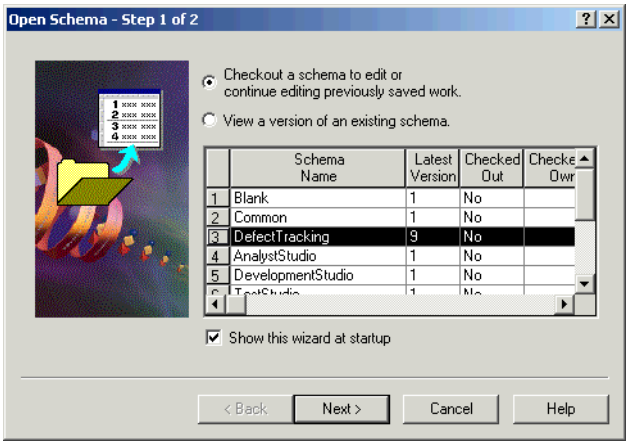
8 Edit the controlled record to assign the security context field in it the value of the security context record to give the groups added in step 7 access to the record.

Warning: Any controlled records that have empty security context field values are hidden. You must complete steps 7 and 8 to expose visibility of the controlled record. The security context field value cannot be null, or all records are hidden from users.

1. Check out a schema to edit

For this example, use one of the predefined schemas included with ClearQuest. Since you want to allow customers to submit defects into your production database, you therefore want to control the records of defect record type.

Login with at least User Administrator and Schema Designer privileges, and choose the schema, “Defect Tracking,” as shown below.

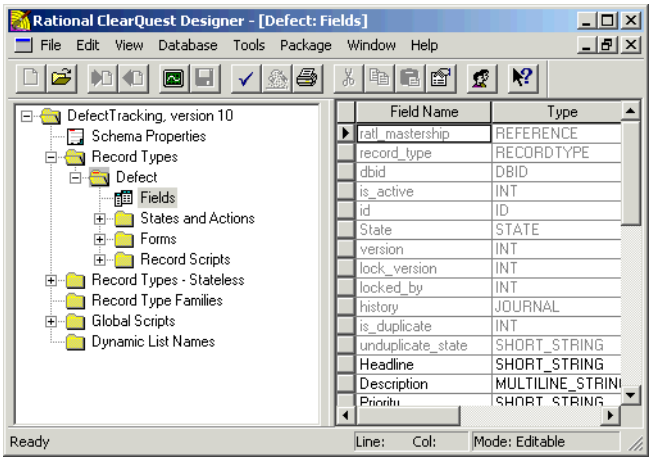


2. Create a security context field

A Security Context field is a field of type Reference in the controlled record type that references the security context record type.

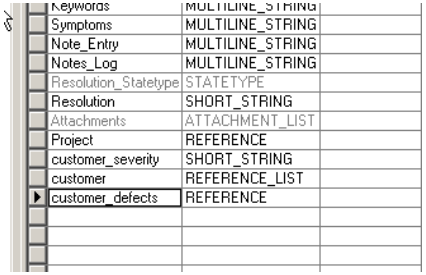
The referenced record type can be a record type you created or one added by a package; however, it cannot be a system record type (such as history, users, groups, and attachments).

In this example, you want to control defect records, so you will open the Fields grid under Defect in the Record Types tree, as shown below.



To create a Security Context field:

1. Create and name a Reference type field, or use an existing reference type field. (Double-click an empty field in the record to bring up the Properties dialog box.) In this example, you will name the security context field “customer_defects.” Note that in the next column the field is designated as a reference type.

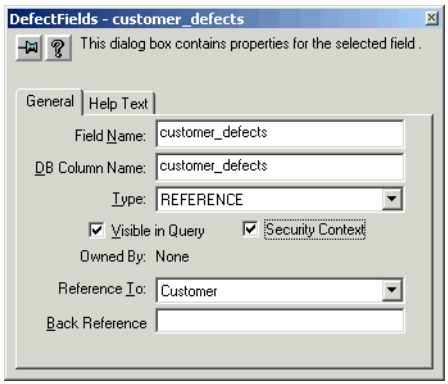


2. Reference the security context field to the record type you want to be the security context record. In the Properties dialog box, select the security context record type from the “Reference To” drop down list. For example, if you are restricting access by customers

and are using a Customer record as your security context record type, select Customer in the “Reference To” field, as shown in the example below. After you choose the record type to which you are referencing the security context field, the “Security Context” checkbox is enabled.

3 Check the Security Context checkbox.

In this example, the security context field is named “customer_defects.” Note that the “Security Context” checkbox is checked.



Note: The security context field must be a reference field type. It cannot be a reference to any system record type, including such record types as history, users, groups, attachments, and so on.

When you check the “Security Context” checkbox, ClearQuest Designer adds a dialog tab to the Submit and default form for the security context record type (in our example, the Customer record type). The tab is named Ratl_Security, but you can change its name. (To change the name of the dialog tab, see “Renaming tabs” on page 92.) You will use this tab in the ClearQuest client to select the groups that can view the record.

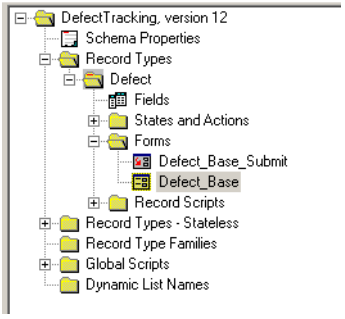
Note: You can add more than one security context field to a record type. For example, you might add a security context field that references the Customer record type and another security context field that references the Quality_Assurance record type. If you add

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 may also want to make the security field Read Only in the Submit state, and optional for other states. In addition, you may want to include a hook to populate the field automatically based on the user who logs in. You should also consider adding security to prevent users from performing actions that they should not. For example, you may want to restrict the Close action to internal users only, and not allow customers to delete a customer record.

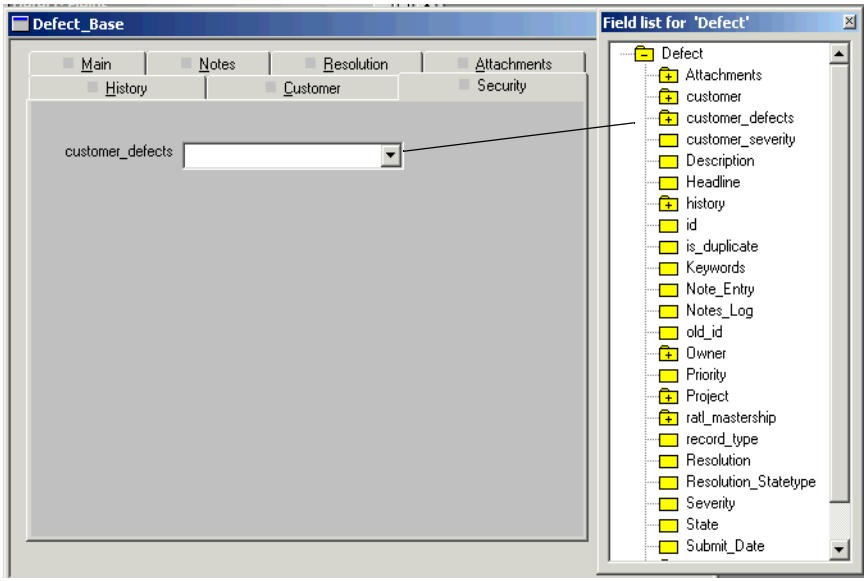
3. Add the field to a form

1 Under the Record Types tree in the Designer, open the form for the record type that your security field is in. In this example, the records in “Defects” are being controlled, so open the form, “Defect_ Base,” as shown below.



2 Add a new tab to your form and name it “Security.” You may also add the security field to an existing tab.

3 Find the security field in the Field List window and drag it onto your form. In this example, our security field is named “customer_defects.”



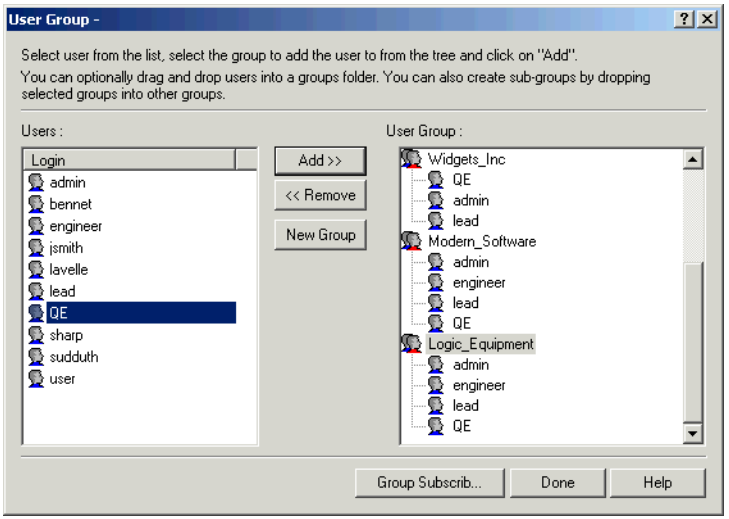
4. Check in the schema and apply the changes

In the Designer, check in the schema and upgrade the user database. Once you perform this step, these changes cannot be undone.

5. Create the groups and add users

Use the User Administration tool to create the groups to be associated with the security context record and add the members to the group. For example, if you are restricting records to users by company name, create a group for each company and add the company’s employees to the group. Create user login(s) for each user, grant them active user privilege and assign them to their user groups, if required.

For this example, create groups for Widgets Inc., Modern Software and Logic Equipment. Add users to these groups, as shown below.



For more information on creating groups, see “Creating user groups” on page 131; to add members to the groups, see “Adding users to a group” on page 132.

Note: You may also 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.

6. Submit the security context records

In the ClearQuest client, choose **Actions > New > Customer** to bring up the “Submit Customer” dialog. Submit a record for each customer.

In this example, submit a record for each of the companies to whom you are granting access to our database. Submit a record for Widgets Inc., Modern Software and Logic Equipment to the Customer record type.

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 (see step 7).

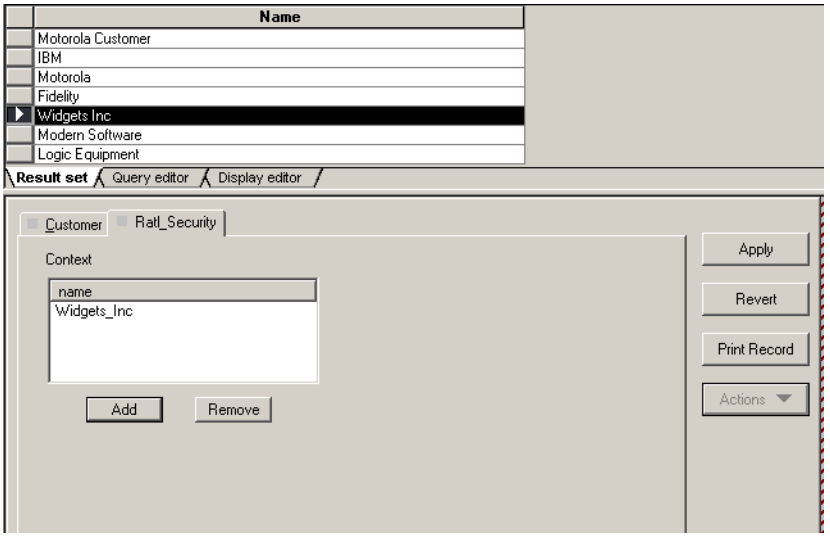
7. Associate groups with each security context record

In this step you associate specific groups with each security context record. For example, choose 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 security context records, perform the following steps in the ClearQuest Client:

1. Create and run a query that displays a list of all security context records. For our example, run a query on “All Customers,” since the security context record type is “Customers.”
2. Open a Customer record and click the Ratl_Security tab. Choose **Action > Modify**.
3. Using the “Add” button on the Ratl_Security tab, select the group or groups that can view the controlled record type. Apply the changes.
4. Repeat steps 3 and 4 for each additional record.

For this example, choose the Widgets Inc. record and add the group created earlier named “Widgets_Inc,” as shown below.



8. Edit specific records to grant privileges to groups

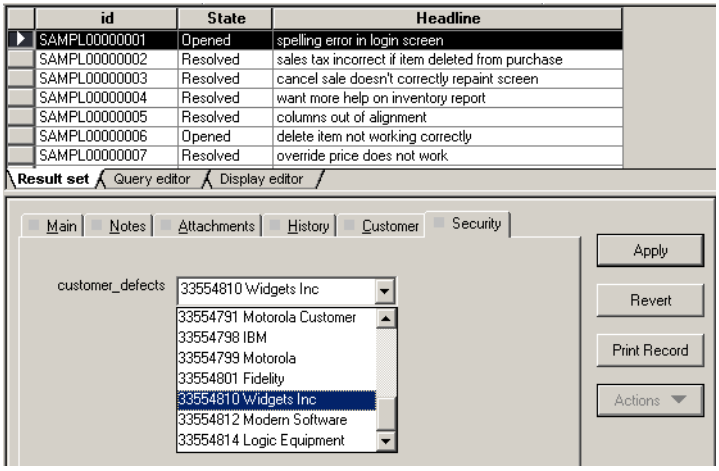
In this step you edit the specific records that you want to open to the users you select. This step assigns the value of the security context record to the security context field. We will open a defect record and grant privileges to our Widgets Inc. user group only.

In the ClearQuest client:

1. Open the database containing the records you want to control.
2. For this example, run a query on “All Defects,” in order to see all of the defect records.
3. Choose and edit the record you want to control. For this example, choose the defect record, “spelling error in login screen,” as shown below.

4 On the form, choose the tab containing the security context field. The security field is named “customer_defects,” so you will see this field on the security tab, with a drop down menu next to it.

5 In the drop down list for the security field, choose the customer whom you want to allow to view and change this record. Select the customer, Widgets Inc., since you want the users in the group, “Widgets_Inc,” to have privileges to the specified record, as shown below. Since you have not granted privileges to Logic Equipment or Modern Software users, this record is hidden from them when they run a query on defects.



6 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.

If you set up security as explained in this example, when an Widgets Inc. customer logs in he or she will be able to perform the following tasks:

- When he runs any of the queries for Defect record type, it will only return records that have Widgets Inc. customer reference.
- Edit existing defect records. The customer choice list will only show Widgets Inc.
- When the customer runs a query, chart or report for customer record type, he should only see the Widgets Inc. record.

7 Administering users

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

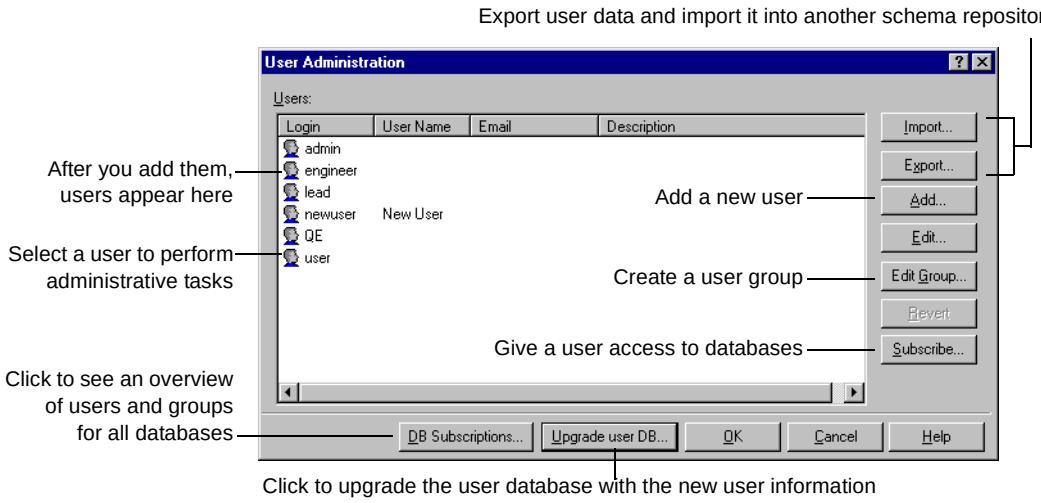
This chapter describes how to set up and administer ClearQuest users and user groups. The topics covered include:

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

Note: To administer users and user groups, you must have User Administrator or Super User privileges. ClearQuest Designer includes a default User Name (*admin*) that you can use to get started. You do not need to type a password. The *admin* user account is set up with the Super User privileges you need to perform all ClearQuest administrator functions. For more information, see “ClearQuest user privileges” on page 125.

Overview of user administration

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



Use the User Administration dialog box to:

- Create new users and user groups: First create users, then create a group and add the users to the group.
- Assign privileges to users and user groups that define the tasks they can perform within ClearQuest and ClearQuest Designer. See “ClearQuest user privileges” on page 125.
- Control the data that users and groups can access by giving them access to specific databases. See “Subscribing users to databases” on page 127.
- Export user data and import it into another schema repository. See “Exporting and importing users and groups” on page 136.

Note: Whenever you add or modify a user or user group, you must upgrade your user database(s) with the new information. See “Applying schema changes to the user database” on page 127.

You can also restrict user and group access to specific ClearQuest actions by adding an access-control hook to the action. See "Controlling user access to actions" on page 135.

ClearQuest user privileges

ClearQuest supports the following user and user group privileges:

Privilege	Allows user /group to
Active User	(Default) Log into ClearQuest and submit new records; modify existing records; and create, modify, and save personal queries, charts, and reports. Log into ClearQuest Designer to view schemas and to view user administration information. Cannot edit schemas or change user information.
Schema Designer	Perform all Active User tasks and: Create, modify, and save public queries, charts, and reports in ClearQuest. Use ClearQuest Designer to create and modify schemas.
User Administrator	Perform all Active User tasks and: Use ClearQuest Designer to create and administer users and user groups.
Public Folder Administrator	Create, modify, save, and delete public queries, charts, and reports in ClearQuest.
Dynamic List Administrator	Edit dynamic lists.
SQL Editor	Edit SQL for queries in ClearQuest client.
Security Administrator	Perform all SQL Editor tasks and: Edit the context group list field for security context record. View all records.
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. Edit ClearQuest Web settings.

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

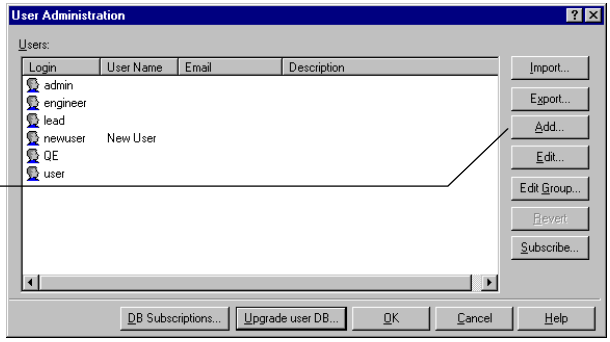
Working with users

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

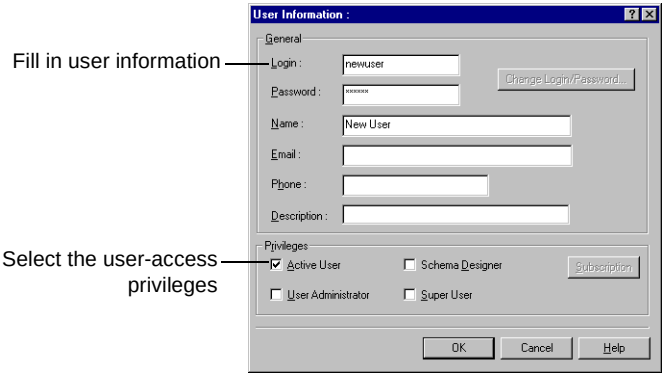
Adding new users

To add a new user:

1. In ClearQuest Designer, select **Tools > User Administration** to open the User Administration dialog box.
2. Click **Add**.



3. In the User Information dialog box, complete the user information and select the user-access privileges:



4. Click **OK**.

5 In the User Administration dialog box, click **Upgrade user DB** to add this new information to your user database(s). See the next section "Applying schema changes to the user database."

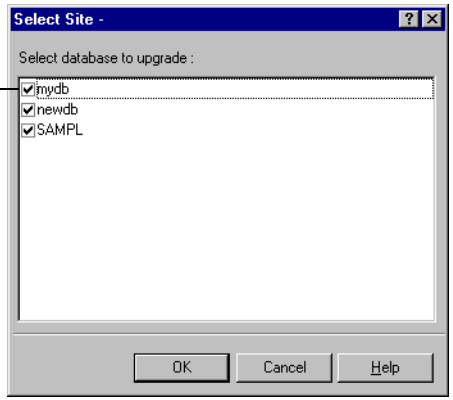
More information? Look up *users, editing an account profile* in the ClearQuest Designer Help index.

Applying schema changes to the user database

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

To apply the schema changes to the user database:

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



Select the databases to upgrade with the new user information

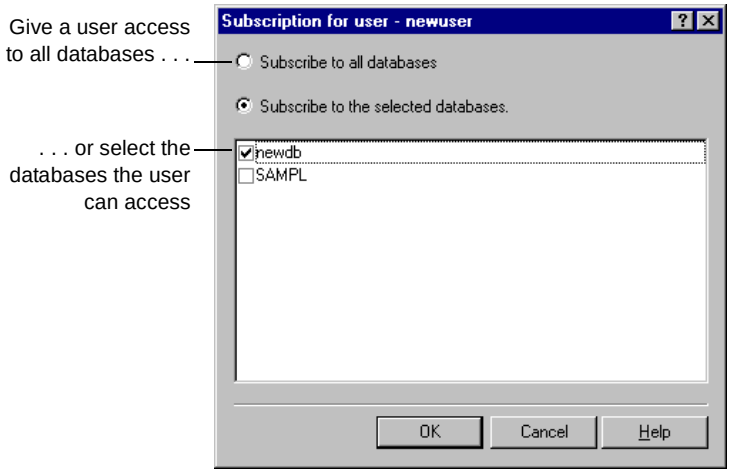
Subscribing users to databases

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

To subscribe a user to a database:

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

2 In the User Administration dialog box, select the user and click **Subscribe** to open the Subscription for User dialog box.



3 Click **OK**.

4 In the User Administration dialog box, click **Upgrade user DB** to add this information to the user database. See "Applying schema changes to the user database" on page 127.

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

Modifying user profiles

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

To edit a user profile:

1 Select **Tools > User Administration** to open the User Administration dialog box.

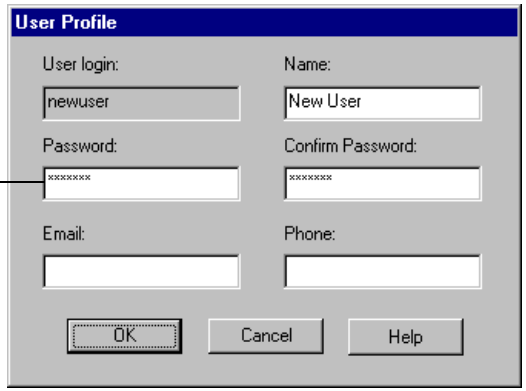
2 Click **Edit**.

Active users can also change some of their own user information, including their name, e-mail address, password, and telephone

number, from either ClearQuest Designer or the ClearQuest client. However, active users cannot change their own user login or privileges.

To modify a user profile from the ClearQuest client:

- Select **View > Change user profile**.



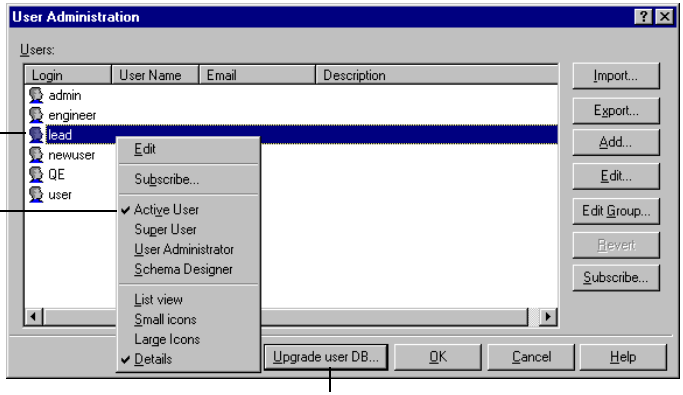
Changes made to the user profile apply to all the databases the user is subscribed to.

Making users inactive

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

To make a user inactive:

1. Select **Tools > User Administration** to open the User Administration dialog.
2. Right-click the user name and unselect all user privileges, including **Active User**.
3. Click **Upgrade user DB** to add the changes to the database.

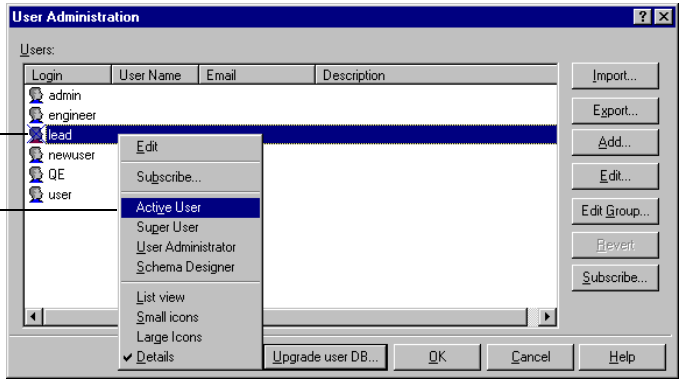


An inactive user appears with an X in the list of users.

To make an inactive user active again:

1 Select **Tools > User Administration** to open the User Administration dialog box.

2 Right-click the user name and select **Active User**.



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

Working with user groups

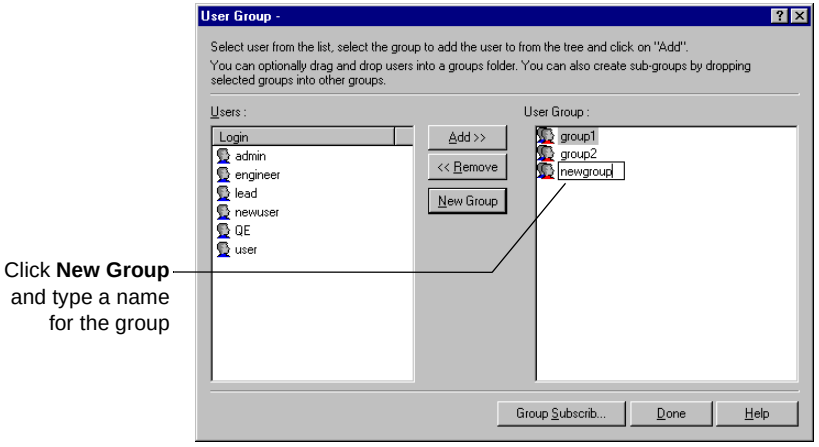
Combining individual users into a group can make your administration tasks easier. For example, you can:

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

Creating user groups

To create a user group:

1. Select **Tools > User Administration** to open the User Administration dialog box.
2. Click **Edit Group** to open the User Group dialog box.

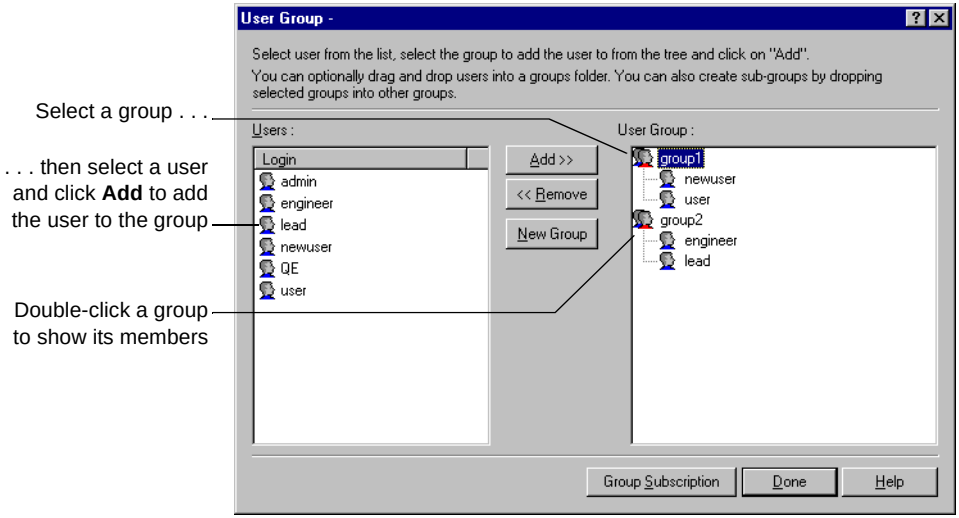


You can use the User Group dialog box to create user groups, add users to groups and remove them, and subscribe groups to databases.

Adding users to a group

To add users to a group:

1. Select **Tools > User Administration** to open the User Administration dialog box.
2. Click **Edit Group** to open the User Group dialog box:



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

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

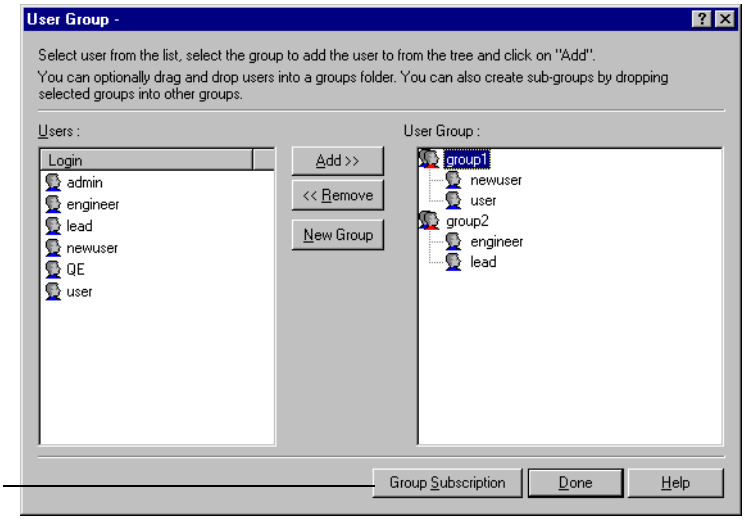
Subscribing user groups to databases

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

To subscribe a group to databases:

1 Select **Tools > User Administration** to open the User Administration dialog box.

2 Click **Edit Group** to open the User Group dialog box.



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

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

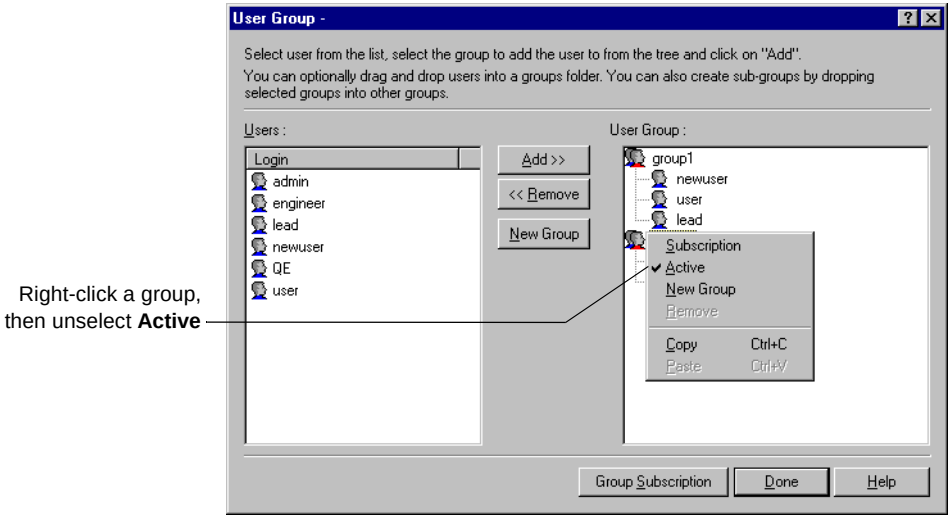
4 In the User Administration dialog box, select **Upgrade user DB** to add the changes to the user database. See "Applying schema changes to the user database" on page 127.

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

Making user groups inactive

To make a user group inactive:

1. Select **Tools > User Administration** to open the User Administration dialog box.
2. Click **Edit Group** to open the User Group dialog box:



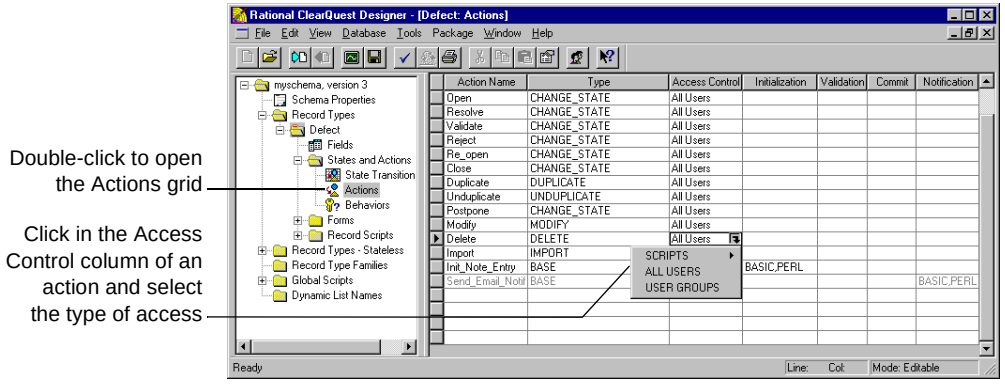
3. When you're finished, click **Done**.
4. In the User Administration dialog box, select **Upgrade user DB** to add the changes to the user database. See "Applying schema changes to the user database" on page 127.

Controlling user access to actions

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

To restrict access to an action:

1 Display the Actions grid.



2 Click inside the Access Control cell of the action, click the down-arrow icon, and select the type of access control to be associated with the action:

- Select **Scripts** to write your own access-control hook that restricts access to this action based on the criteria you define. For example, you can write a hook that allows access to an action to users who have Super User privileges. For more information, see Chapter 8, “Using hooks to customize your workflow.”
- Select **ALL USERS** (the default) to allow any ClearQuest user to access the action.
- Select **USER GROUPS** and select the groups you want to have access to this action.

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

Exporting and importing users and groups

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

Working with users in a MultiSite environment

When working in a ClearQuest MultiSite environment, it is important that you know which users have rights to change a record or object in any replicated database you're viewing. As an administrator, you need to understand some basic concepts about replicating databases in ClearQuest, as well as how user privileges are affected when working in a MultiSite environment. You also need to understand how to identify replicated databases and the user privileges that are associated with them. You can identify the privileges granted to a user of a replicated database by noting certain changes in the Designer interface. This section explains the basic MultiSite concepts, as well as how the Designer interface reveals the privileges a user has on a replicated database.

The following topics are covered in this section:

- Understanding mastership
- User privileges in a MultiSite environment
- Identifying mastership and user privileges
- Identifying group privileges

Understanding mastership

ClearQuest MultiSite allows you to replicate the same database on multiple sites while still maintaining the consistency of the data across all replicas. Data can be added to any replica, and each replica in the set is automatically refreshed with the latest information at scheduled intervals. Multiple sites function locally and are, at the same time, integrated with the whole set of replicas.

Maintaining the consistency of the data in each replica is accomplished with a ClearQuest feature called *mastership*. Each ClearQuest object, including records, schemas, queries, and so on, is assigned a mastering replica. The original site where the ClearQuest object was first created is the mastering replica assigned to that object. For example, if you create a new record at

your site (site A), your site is the mastering replica and has mastership of that record. A user at another site (site B) can see the record you created, but they cannot change that record in their replica of the database. In this example, the user at site B can request mastership of the record you created at site A. Mastership of the record can be reassigned from your database to the replica at site B, thus allowing the user at site B to change the record.

User privileges in a MultiSite environment

When you create a new user at your site you are creating a new user record, and your site has mastership of this record. This record can be changed only at your site. The privileges assigned to users and groups created at your site can only be changed at your site, because your site has mastership of these records.

The important thing to understand is that the site where a user was originally created determines which replicated databases they have privileges on.

If a user who is created at another site logs into a replica from your site, they do not have privileges to do the following tasks:

- Change their user profile
- Save defaults while submitting queries
- Assign startup queries
- Add workspace items to the Query menu as favorites

However, a user can log into a replica from another site and do the following tasks:

- Query records
- View records
- Create new records
- Modify records

Any changes a user makes on a replica will also appear in all replicas of that database at all sites. For example, if a user creates a new record on a replica, that record can be viewed on all replicas

of that database across all sites; however, that record can only be modified at the site it was created.

Identifying mastership and user privileges

As a ClearQuest administrator, it is very important that you know which users have rights to change a record or object in any replicated database you're viewing.

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 originating site where the user was created. You cannot change the privileges of a user who was mastered at another site. Since all users associated with a replica, regardless of the site on which they were created, are visible to you as an administrator, you need to be able to identify which users you can modify.

The User Administration dialog box in the Designer indicates when a user has or has not been mastered at your site. When you select a user in the User Administration dialog box, the "Edit" and "View" buttons indicate if the selected user was created at your site or at another site.

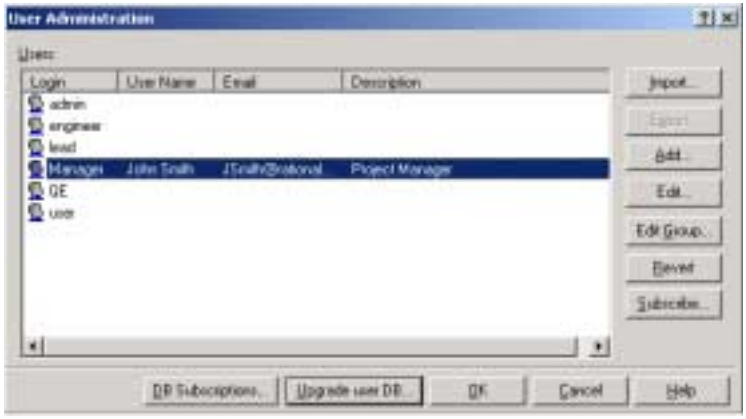
To determine whether a user was mastered at your site, choose **Tools> User Administration** to open the User Administration dialog.

In the User Administration dialog box:

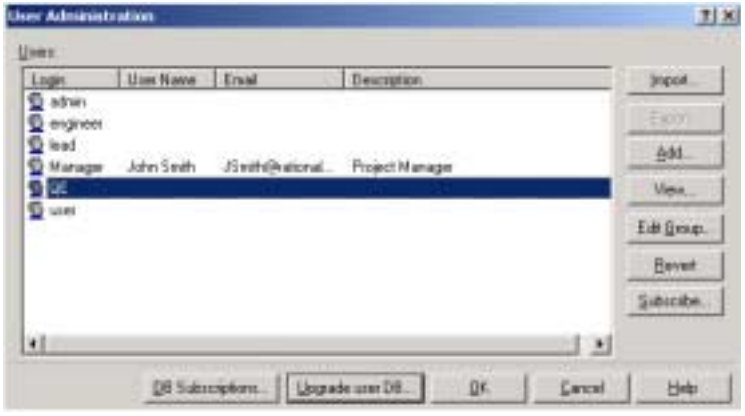
- The "Edit" button is enabled when you select a user created at your site.
- The "View" button replaces the "Edit" when you select a user created at another site.

For example, in the following figure the user, "John Smith," is selected and the "Edit" button is visible and enabled. The user

John Smith was therefore created at your site, and you can choose the “Edit” button to modify his user privileges.

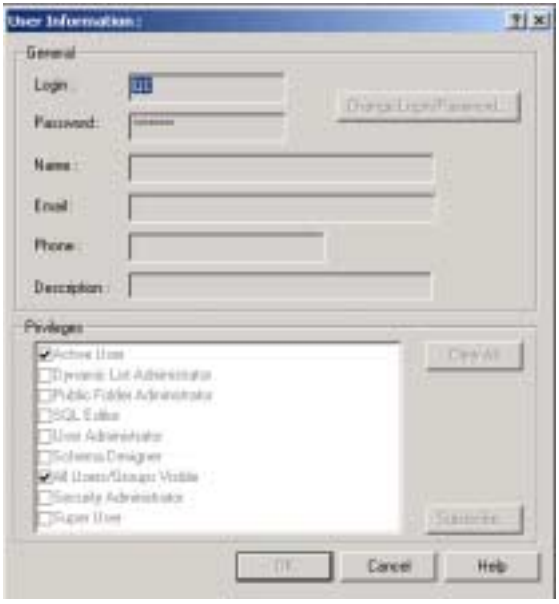


In the following example, the user, “QE,” is selected in the User Administration dialog box. The “Edit” button does not appear, and the “View” button appears in its place. Therefore, the selected user, “QE,” was created at another site. You cannot modify this user’s privileges from your site.



You can, however, choose the “View” button to see the User Information dialog box to see the privileges given to this user. When you choose the “View” button, the User Information dialog

box is displayed and you can view this user's profile, as shown in the figure below. Note, however, that the buttons and fields are not enabled, preventing you from modifying this user record and indicating that this user was created at another site.



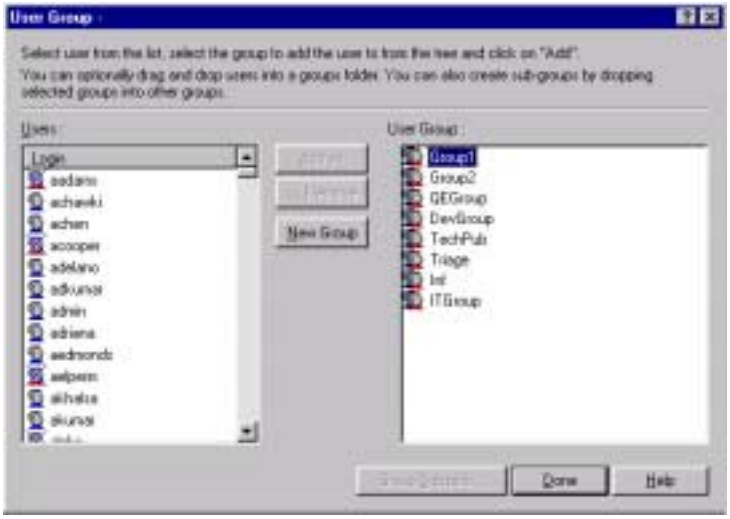
Identifying group privileges

A group can only be modified at the site at which it was created, just like a user. When you create a user group at your site, you create a group record and your site has mastership of that record. The group therefore can't be modified at any other site. For example, users can be added to, or deleted from, the group at your site only.

The group will appear on all sites associated with the replica. When you choose the "Edit Group" button on the User Information dialog box, the User Group dialog box is displayed. A list of all groups associated a replica can be viewed.

When you choose a group that was created at your site, the “Add” and “Remove” buttons are enabled on the User Group dialog box. This means that your site has mastership of this group record, and you can add or remove users from this group.

If you select a group that was created at another site, the “Add” and “Remove” buttons are disabled, as shown in the figure below. Another site has mastership of this group record, and you cannot add or remove users.



8 Using hooks to customize your workflow

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

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

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

More information? For more information on the ClearQuest API, select **Rational ClearQuest API Reference** from the Start menu. The *Rational ClearQuest API Reference* is an online reference guide for all ClearQuest API calls. To learn about using hooks in the web environment, see Chapter 9, “Administering ClearQuest Web.”

Overview of hooks

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

- Field hooks

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

- Action hooks

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

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

- Record scripts

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

- Global scripts

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

Note: You can write hooks in VBScript (for Windows) and Perl (for Windows and UNIX). By default, ClearQuest runs hooks in VBScript on Windows. To learn how to have ClearQuest run hooks in Perl for Windows, see “Selecting a scripting language” on page 40. When you finish editing a hook, select **Hooks > Compile** to check the syntax of your code.

Planning hooks for your workflow

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

- Be aware that hooks run with Super User privileges and are not subject to the usual access control or field behavior restrictions. This means you can use hooks to set and reset values in fields that are normally read-only. (You cannot reset ClearQuest system fields that are read-only, such as the History field.) Required fields remain required.
- You can take advantage of outside code to extend hook capabilities. VBScript has access to COM objects, and Perl can gain access to COM objects through a third-party package. In addition, Perl can take advantage of many third-party packages. For example, a hook could interact with the user through dialog boxes, or read from and write to external files. You are responsible for ensuring that the proper third-party objects are installed on the client machine.
- Although it is possible to include SQL code in your scripts, for best results you should use the ClearQuest API to run queries and to retrieve data within script code. ClearQuest allows you to rename record types and fields, and this will be problematic in any SQL code that you include.
- If you plan to run your hooks on ClearQuest Web, write them in VBScript and be sure to test them in the web environment. See "Using hooks in ClearQuest Web" on page 178.
- Test and debug your hook code so that you do not write incorrect values into your database or cause the program to hang while processing an infinite loop or waiting for nonexistent input. To view debugging information (the output of the OutputDebugString method), you can use dbwin32.exe, which ships with ClearQuest.

Using field hooks

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

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

Field hook	Use
Choice List	Returns a set of legal values. Use this hook with fields that are displayed using a list-type control, such as a list box or combo box. You can also provide values without scripting by using a constant or a dynamic list. See “Working with choice lists” on page 149.
Default Value	Sets the initial value of the field. This hook is called at the beginning of a Submit action. You can also assign a constant value as the default value.
Permission	Returns one of the BehaviorType constants indicating the user's access to the field. Use this hook to force workflow and/or security. (See the online *ClearQuest API Reference* for enumerated constants.) If you add a Permission hook to a field, you must modify the Behaviors grid so that at least one of the field's behaviors is set to USE_HOOK. Failure to do this will result in a validation error.
Validation	Validates the contents of the field. This hook is called when the value changes, to provide the user with immediate feedback regarding the validity of the field's contents before committing the record to the database.
Value Changed	Responds to changes in the value of a field. Use this hook to trigger updates for other fields (for example, dependent lists). After executing this hook, ClearQuest validates any field the script has modified by calling the field's Validation hook (if any).

More information? For instructions on how to specify field hooks, see “Customizing fields by adding hooks” on page 72. To learn about the sequence in which hooks run, see “Execution order of field and action hooks” on page 154.

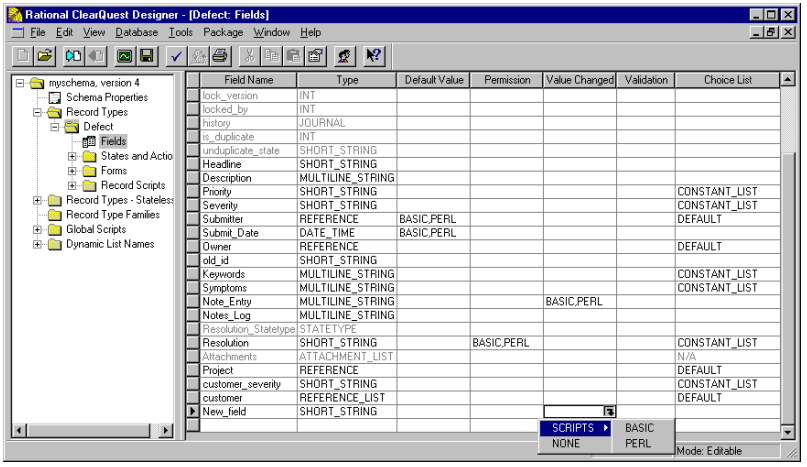
Creating a dependency between fields

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 that can introduce runtime errors if not written correctly. Be sure to test script hooks thoroughly before making them available to your users.

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

1 Display the Fields grid.



2 Click the Value Changed cell of the parent field.

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

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

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 170.)

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

Using dependent fields on ClearQuest Web

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

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

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

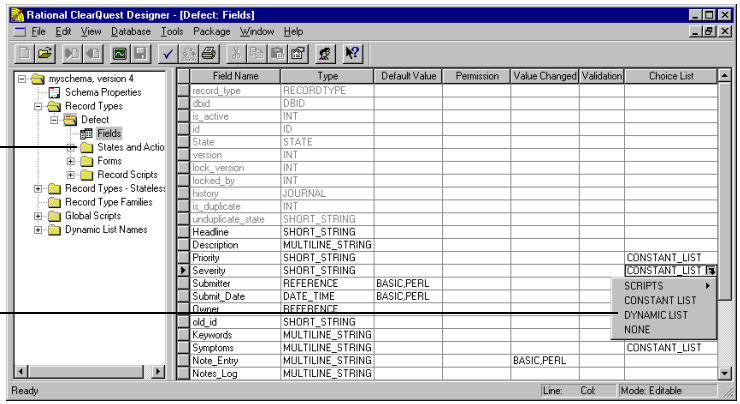
More information? See “Enabling dependent fields for ClearQuest Web” on page 178.

Working with choice lists

Rather than requiring that users type the value for a field, you can use a choice-list hook to provide a list of options from which to choose. 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.



Double-click to open the Fields grid

Click in the Choice List cell of a field and select a choice list

2 Click the Choice List cell.

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

- Select **SCRIPTS** to write a script for a choice list. You can define a dependent list whose values change when the user selects a certain value in another form control. (For an example, see “Hook for creating a dependent list” on page 165.)
- 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 list hook” on page 150.

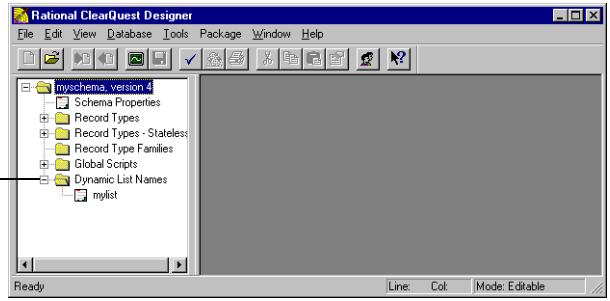
- Select **NONE** (the default) for Reference and Reference List field types only. The system presents a choice of all the active records of the given type as a default choice list for these field types.

Creating a dynamic list hook

To create a dynamic list hook:

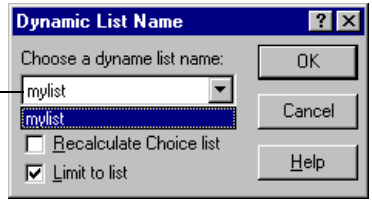
1. In the Workspace, right-click **Dynamic List Names** and select **Add** from the shortcut menu.
2. Type a name for the dynamic list.

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



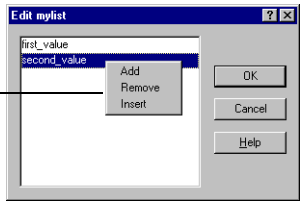
3. In the Fields grid, click the Choice List cell for the field, click the drop-down arrow icon and select **DYNAMIC LIST**.
4. In the Dynamic List Name dialog box, select the dynamic list name you just created.

Select the dynamic list name to use



5. Define the values for the dynamic list. You must have Schema Designer, Dynamic List Administrator, or Super User privileges to edit the list. (See “ClearQuest user privileges” on page 125.)

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



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

Defining choice list behavior

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

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

Using action hooks

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

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

Action hook	Use	Executed
Access Control	Returns a Boolean indicating whether the specified user can initiate the specified action on a record. This hook is called before the user performs the action. You can write an access-control hook as a VBScript or Perl subroutine. See “Controlling user access to actions” on page 135.	When the action is about to start.
Initialization	Sets initial field values (or any task you specify). Allows complex initialization of a record. You can use this hook to set up field values before ClearQuest begins an action. This hook is called after the action has been initialized but before the contents of the record are displayed in a form. You must write an initialization hook as a script subroutine.	When the action starts.

Action hook	Use	Executed
Validation	Validates the field values you specify. If the user types invalid data, ClearQuest prompts the user for valid data. You can use this hook to check conditions that are difficult to verify inside the individual field validation hooks. For example, you can use this hook to verify information across a group of fields. ClearQuest executes this hook prior to committing any changes to the database. Validation hooks must use a script.	When the user commits the action.
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 10, “Administering ClearQuest E-mail” . Notification hooks must use a script.	After ClearQuest commits the transaction.

More information? For instructions on how to specify action hooks, see “Customizing actions by adding hooks” on page 83. To learn about the sequence in which hooks run, see “Execution order of field and action hooks” on page 154.

Execution order of field and action hooks

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

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

When an action begins

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

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

When a field's value is set

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

1. The **Field Value Changed** hook runs for each field that changes.

   If the **Limit to list** option is set and the user enters an illegal value, ClearQuest marks the field as invalid. Only when the user enters a legal value does the next hook run.

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

2 The **Field Validation** hook runs for each field.

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

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

Note: In ClearQuest Web, field hooks run only when the user invokes the Submit action. See “Using hooks in ClearQuest Web” on page 178.

When the record is validated

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

1 The **Field Validation** hook runs for each field in the record.

2 The **Action Validation** hook runs for the action that was performed.

When the record is committed

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

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

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

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

Using record scripts

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

There are three ways to invoke a record script:

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

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

- Use an action to invoke a record script. Create a Record_script_alias action and associate it with a record script. When the user selects the action, the record script is invoked immediately. To update the current record from within such a script, begin an action in the script using the ClearQuest API `EditEntity` method in the `session` object.

- Call a record script from within another script or hook by using the ClearQuest API `FireNamedHook` method in the `entity` object.

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

Using global scripts

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

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

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

Writing external applications

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

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

An external application must begin by creating a session object and logging into a ClearQuest database. See "Working with sessions" on page 159.

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

Using the ClearQuest API

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

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

Working with sessions

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

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

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

Working with queries

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

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

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

Working with records

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

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

Common API calls

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

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

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

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

API Call (VBScript/Perl)	Function
`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."

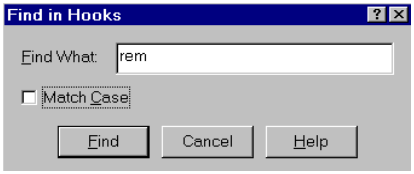
`QueryFilterNode.BuildFilter` `$QueryFilterNode->BuildFilter`	Creates support for a complex query.
`session.BuildResultSet` `$session->BuildResultSet`	Creates the ResultSet object necessary to run a query.
`ResultSet.Execute` `$ResultSet->Execute`	Runs the query with the current ResultSet object.
`ResultSet.MoveNext` `$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 hook script text

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

1 Open the schema for which you want to search hook script text.

2 Select **Edit > Find in Hooks** and enter the text string that you want to search using the following fields:



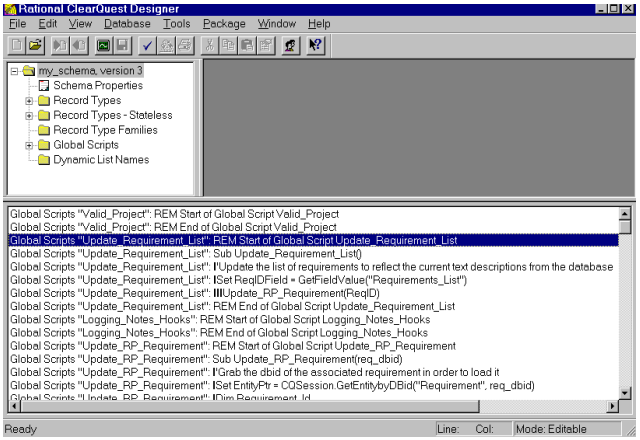
- **Find What**

Type the text for which you want to search.

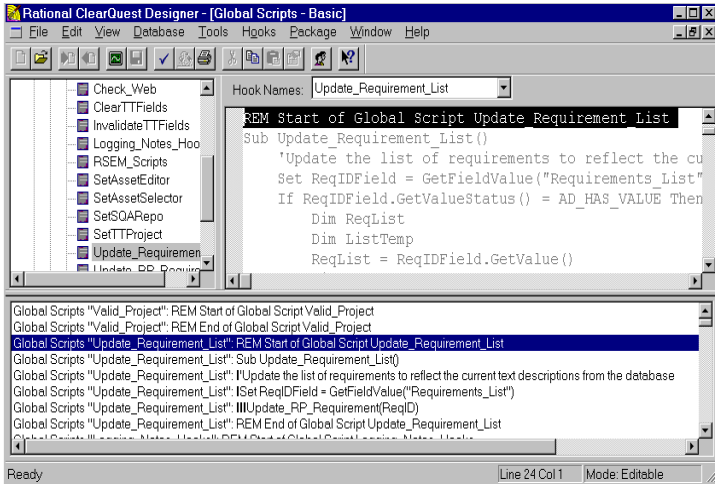
- **Match Case**

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

3 The Validation pane displays the search results:



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



Examples of common hooks

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

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

Hook for creating a dependent list

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

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

   VBScript:

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

   Perl:

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

2. To prevent your users from adding new members to the list, check the **Limit to list** option.

3. To clear the old value in client_os when a new value is selected in server_os, add the following to the server_os Value Changed hook:

   VBScript:

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

   Perl:

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

4 On the client_os field, create a Choice List hook:

VBScript:

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

Perl:

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

5 In the properties for the client_os hook, check **Recalculate Choice list**, so that whenever the server_os field changes, the values recalculate.

6 Add the client_os and server_os fields to your form using list box controls.

Field choice list hook to display user information

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

VBScript:

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

   Dim sessionObj
   Dim queryObj
   Dim filterObj
   Dim resultSetObj

   Set sessionObj = GetSession()

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

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

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

   ' run it
   resultSetObj.Execute

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

Perl:

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

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

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

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

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

    # run it
    $resultSetObj->Execute();

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

Action initialization hook for a field value

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

VBScript:

```
Dim session
Dim loginName

‘ Get the session
set session = GetSession

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

Perl:

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

Action hook for setting the value of a parent record

Use the following action hook in conjunction with parent/child linking to create a state-based relationship between a parent record and its children. This hook allows you to automatically move the parent record to the next state if all the children are in that state (see “Using fields to link records” on page 68).

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

VBScript:

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

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

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

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

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

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

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

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

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

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

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

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

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

    end if

  Next

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

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

    ParentObj.Commit
  end if
end if
```

Perl:

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

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

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

foreach $ChildID (@ChildArray) {

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

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

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

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


    if ($StateStatus eq "") {

        $StateStatus = $CurrentState;
        $SameState = 1;

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

    } elsif ($StateStatus eq $CurrentState)  {

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

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

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

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

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

9 Administering ClearQuest Web

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

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

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

ClearQuest Web considerations

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

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

Customizing ClearQuest Web

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

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

Limiting access to ClearQuest Web

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

- Submit records
- Run a single query that you provide

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

For example, you can enable beta customers to provide feedback through a password-protected “extranet.” First, you would create a ClearQuest user group (see “Working with user groups” on page 131) that consists of customers you want feedback from, and then distribute the ClearQuest URL to that group. That group will only be able to submit records and use the query that you provide. The query results are read-only.

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

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

Using hooks in ClearQuest Web

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

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

Enabling dependent fields for ClearQuest Web

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

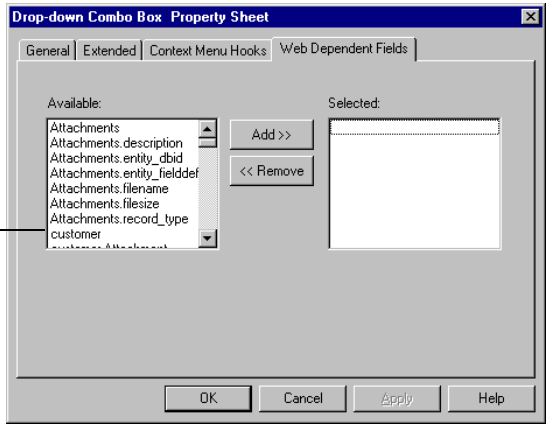
1. Create the dependent field relationship using a value-changed field hook.

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

2. Use the appropriate control to add the field to the record form. Both the field that creates the dependency and the dependent fields must use one of the following controls. You can mix and match them:
   - Drop-down list box
   - Combo box
   - Drop-down combo box

3. Right-click the control on the form and select **Properties** from the shortcut menu.

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



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

Displaying messages on ClearQuest Web

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

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

Using hooks to detect a web session

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

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

Web session detection in VBScript

```
dim currDBSession      ' Current Db session

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

Web session detection in Perl

```
my $currDBSession;   # Current Db session

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

10 Administering ClearQuest E-mail

ClearQuest takes advantage of e-mail in two ways:

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

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

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

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

Enabling automatic e-mail

To enable ClearQuest to send automatic e-mail:

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

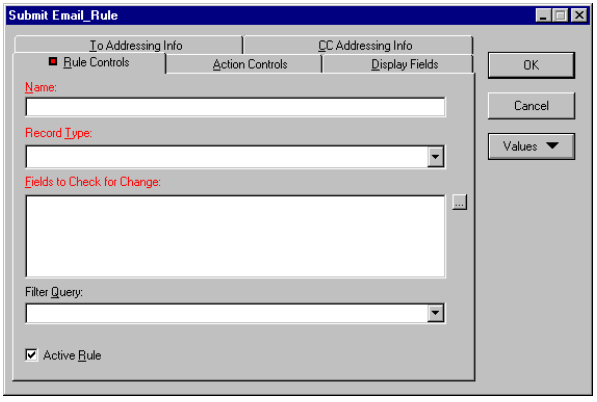
Setting up e-mail rules

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

You create e-mail rules by submitting Email_Rule records from the ClearQuest client. To add or modify an e-mail rule, you must have Super User or Schema Designer privilege (see “ClearQuest user privileges” on page 125).

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

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



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

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

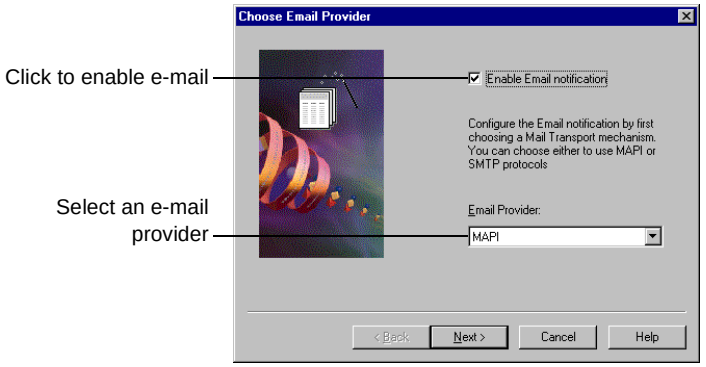
Note: Most ClearQuest predefined schemas include the Email_Rule record type. If the schema you are using does not, you can add this functionality by adding the Email package (see Appendix A, “ClearQuest schemas and packages”).

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

Configuring ClearQuest clients to send e-mail

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

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



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

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

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

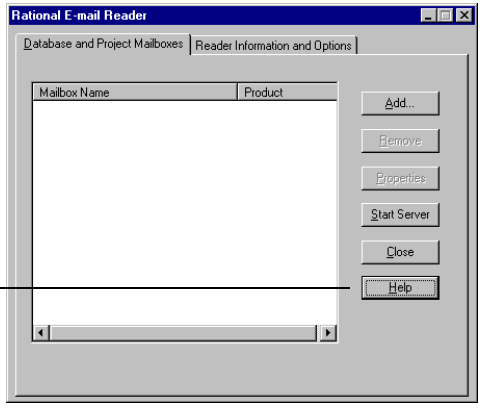
Submitting records by e-mail

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

Configuring Rational E-mail Reader

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

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



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

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

- The E-mail Reader requires a dedicated e-mail account that has been assigned for parsing records.

- You must designate one machine to run the E-mail Reader. To ensure that e-mail services are always available, run the E-mail Reader on the same machine as your database server.
- Any e-mail sent must be in the required format. See “Formatting e-mail for submission” on page 185.
- You should set up defaults to be used. This allows users to quickly respond to e-mail notification without taking into account the required format.
- Field values submitted by e-mail overwrite the existing data. If you want to append existing data, you must specify a field that allows users to append data, such as the Notes Log (available in most predefined schemas).

Formatting e-mail for submission

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

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

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

The Subject line follows these guidelines:

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

- The record ID is listed after the action, but is not necessary when submitting new records.

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

E-mail format example

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

Subject:RE: defect SAMPL00000017

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

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

Guidelines for using the e-mail format

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

Use the following guidelines when sending e-mail:

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

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

- If legal values are not used, the Rational E-mail Reader sends an error message to the submitter.
- E-mail submission does not support adding attachments.

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

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

Using "round-trip" e-mail

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

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

1. A user submits a record by e-mail, using the required format. (See "Formatting e-mail for submission" on page 185.) ClearQuest assigns the new defect ID number PROD0000137.
2. An e-mail rule you established notifies the lead engineer of this defect. (See "Setting up e-mail rules" on page 182.)
3. The lead engineer reads the e-mail entitled "PROD0000137," then responds by e-mail, including remarks in the description field, using the required format.

   Note: The reply should include only fields that you want to modify. Any field included in the message will overwrite existing data.
4. The Rational E-mail Reader processes the e-mail message and submits the modified record to the ClearQuest database.

11 Importing data into ClearQuest

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

Topics covered include:

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

Preparing for a successful data import

Before you begin importing data, plan your conversion carefully:

- Analyze your current data and workflow so that you know how to implement it in ClearQuest. For an overview of how to work with ClearQuest, see Chapter 1, "Understanding ClearQuest administration."
- Create an import schema to support the data you want to import. See "Creating an import schema" on page 190.
- Upgrade an existing user database or create a new database and associate it with the import schema. See "Creating a database for imported data" on page 192.
- Export your existing data to text files that use the ClearQuest import-file format. See "Creating an import schema," below.
- Use the ClearQuest Import Tool to import the data from the import files into your user database. See "Performing the data import" on page 197.

Note: Before importing all of your data, first test the entire import process on a subset of your data. See "Testing the import process" on page 192.

Creating an import schema

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

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

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

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

Note: ClearQuest does not validate data types during import. If the records you are importing contain data types that ClearQuest does not support, you can map those data types to other ClearQuest types. By default, ClearQuest maps unsupported data types as STRING type. See "Data types supported in ClearQuest" on page 194.

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

More information? See Chapter 3, "Working with ClearQuest Schemas," and Chapter 4, "Customizing a schema."

Original record ID

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

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

Creating a database for imported data

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

More information? See "Applying schema changes to a user database" on page 44. Look up *databases, creating* in the ClearQuest Designer Help index.

Testing the import process

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

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

Creating a ClearQuest import file

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

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

Formatting the record import files

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

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

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

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

Important import-file format considerations

Keep in mind the following when formatting an import file:

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

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

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

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

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

Data types supported in ClearQuest

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

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

Data Type	Description	Behavior
REFERENCE	A reference to a display name in a state-based or stateless record type	Interpreted as a key to a stateless record type
REFERENCE_LIST	Multiple references to display names in a state-based or stateless record type	Interpreted as a list of references. Note that reporting is not supported for REFERENCE_LIST fields.
SHORT_STRING	A variable-length character string with a maximum length of 254 characters	Inserted as is
STATE	Reserved for system fields	ClearQuest validates the value of the field, but does not trigger actions required to reach the state

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

Formatting the history import file

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

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

Note: If you use the ClearQuest Export Tool to export data from another ClearQuest database, the Export Tool saves the original

(old) record ID in a field called “display_name.” Map the display_name field to the original (old) record ID.

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

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

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

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

Formatting the attachments import file

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

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

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

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

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

Performing the data import

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

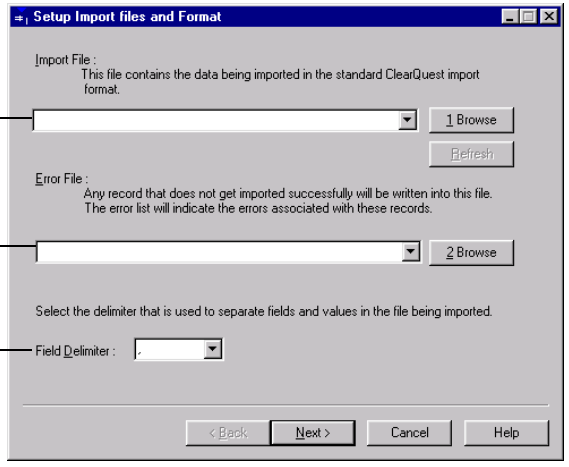
If you have multiple record types, you must import them in an order that allows referencing. For example, if you have a defect record type that refers to a project record type, first import the project record, then import the defect record.

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

Note: You can also import duplicate records separately. See "Importing duplicate records" on page 204.

To perform the import for records:

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



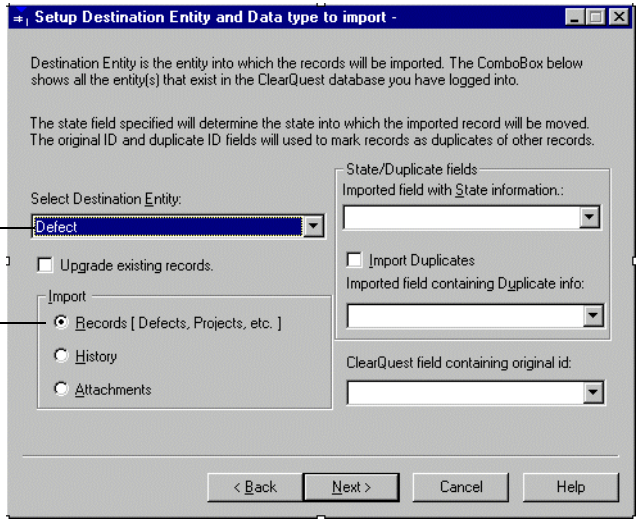
When you're done, click **Next** to open the Setup Destination Entity and Data Type to Import dialog box.

Note: If a record does not convert successfully, it is saved to the error file you specify. You can use this error file to fix problems and to reimport the record. See "Importing records from the error file" on page 205.

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

Select the record type to import the records into

Select the type of data you are importing



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

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

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

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

If the imported records contain duplicates, click **Import Duplicates** and select the exported field name that contains the duplicate information. For more information on importing duplicates, see “Importing duplicate records” on page 204.

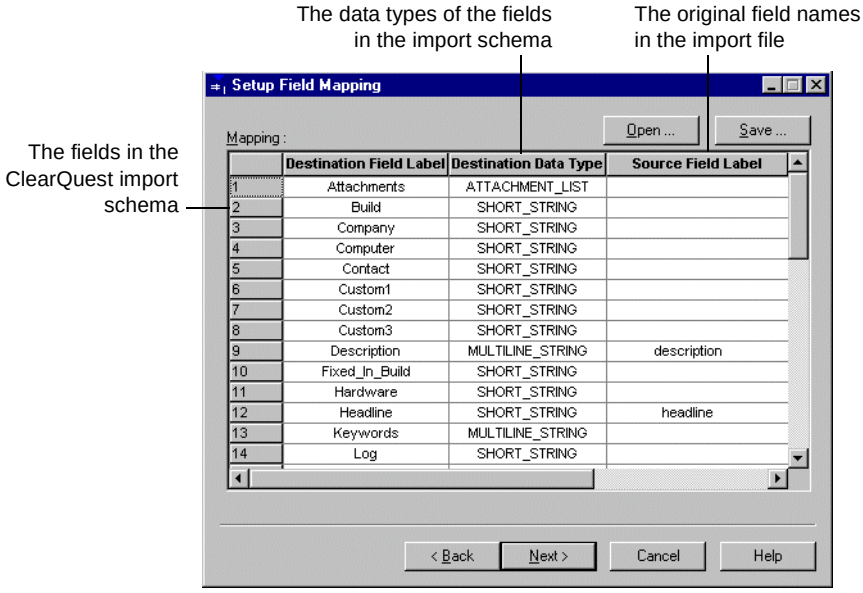
Select the **ClearQuest field containing original ID**. This is the field in the import schema that contains the original record ID of the exported data. This field is important for importing history, attachments, and duplicate records.

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

If you used the ClearQuest Export Tool to export duplicate record information separately, click **Upgrade existing records** and **Import Duplicates**. This adds the duplicate information to the existing records. If you are importing duplicate records from another application, see “Importing duplicate records” on page 204.

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

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

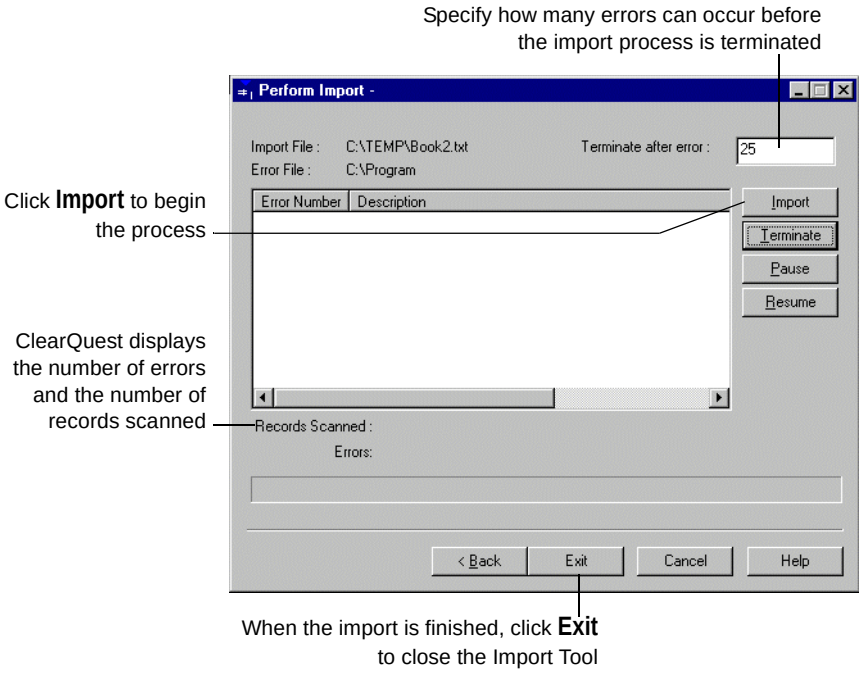


- You cannot map a field in the import file to more than one field in the ClearQuest schema.
- If the time field is mapped to a date-time field in ClearQuest, today's date is appended to the time. Also, if the date field is mapped to a date-time field in ClearQuest, the current time is appended to the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

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

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

5 In the Perform Import dialog box:



Importing history

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

Before importing history information, be sure you have an import file containing the history fields. See “Formatting the history import file” on page 195.

To import history, follow the procedure for “Performing the data import” on page 197, with the following changes:

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

All other steps remain the same.

Importing attachments

After importing your records, you are ready to import your attachments. Before importing attachments, be sure you have an import file containing the attachment information. See “Formatting the attachments import file” on page 196.

To import attachments, follow the procedure for “Performing the data import” on page 197, with the following changes:

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

All other steps remain the same.

Importing duplicate records

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

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

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

To import duplicate records:

1. Follow the steps for “Performing the data import” on page 197.
2. In step 3, in the Setup Destination Entity and Data Type to Import dialog box, click **Import Duplicates** and enter the name of the field that contains the original record ID (the ID of the record this is a duplicate of).
3. Run the import. Any duplicate that cannot reference its parent record (because it was imported before the parent) is saved to the Error file.
4. When the import is completed, use the Error file to import the duplicates that did not successfully import the first time.

Importing duplicate records separately

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

Importing records from the error file

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

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

To reimport these problem records:

1. Check the `errlog.txt` file and review the types of errors encountered.
2. Open the error file containing the unimported records.
3. Correct the errors in the records. Be sure the error file uses the import file format. See "Formatting the record import files" on page 193.
4. Perform the import process again, this time specifying the error file as the import file.

Upgrading existing records

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

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

To upgrade existing records, follow the procedure for "Performing the data import" on page 197. In step 3, in the Setup Destination Entity and Data Type to Import dialog box, select **Upgrade existing records**. All other steps remain the same.

A ClearQuest schemas and packages

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

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

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

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

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

Warning: You cannot uninstall or otherwise remove a package from a schema. Be sure to plan carefully before adding a package into a schema.

More information? For instructions on how to add packages to a schema, see "Adding packages to a schema" on page 47. For information on how to customize a schema, see Chapter 4, "Customizing a schema."

ClearQuest predefined schemas

ClearQuest includes the following predefined schemas:

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

Schema	Description	Packages included
Enterprise	For use with Rational EnterpriseStudio. Contains fields and hooks that work with most Rational Suite products. This schema is UCM-enabled.	AMStateTypes, Attachments, BaseCMActivity, Email, EnhancementRequest, History, Notes, PQC, Resolution, Repository, RequisitePro, TeamTest, UCMPolicyScripts, UnifiedChangeManagement, TeamTest
UnifiedChange Management (UCM)	Provides support for integration with UCM and sets up ClearQuest to use the predefined ClearCase policies.	AMStateTypes, Attachments, BaseCMActivity, Email, History, Notes, Resolution, UCMPolicyScripts, UnifiedChangeManagement

ClearQuest packages

ClearQuest includes the following packages. Some packages are read-only; their functionality cannot be changed. This section describes the latest version of each package. Earlier versions may create different record types and fields.

Package	Description	Record type(s) added/modified	Field(s)
AMBaseActivity	Provides additional support for Rational ClearQuest Project Tracker.	Adds the Main tab to the enabled record type.	**Fields 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
AMWorkActivitySchedule	Provides scheduling attributes needed to integrate Rational ClearQuest and Microsoft Project 2000 using Rational ClearQuest Project Tracker. With the AMWorkActivitySchedule record type family, you can query records being created and updated with Rational ClearQuest Project Tracker.	Defines and adds the AMSchedule record type family to the enabled schema. Record types being enabled with this package will be added to this record type family. Adds the Schedule tab to the enabled record type.	**Fields added to the enabled AMSchedule record type:** am_planned_start_date am_planned_end_date am_planned_work am_planned_rem_work am_planned_duration am_planned_rem_duration am_actual_start_date am_actual_end_date am_actual_work
Attachments (read-only)	Allows you to add and remove attachments related to a record.	Adds an Attachments tab to the enabled record type.	**Fields added to enabled record type:** Attachments

Package	Description	Record type(s) added/modified	Field(s)
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 ClearCase*.	Adds the BaseCMActivity record type.	**Fields included in BaseCMActivity record type:** Owner Description Headline
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	Record type(s) added/modified	Field(s)
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/change tracking system.	Adds a Customer stateless record type. Also adds reference fields for customer information to the record type you choose.	**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	Record type(s) added/modified	Field(s)
Email (read-only)	Supports automatic e-mail notification when records are modified.	Creates an Email_Rule stateless record type. Adds a base action to the enabled record type called "Send_Email_Notif." This base action runs the e-mail rule whenever any action is invoked.	**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

Package	Description	Record type(s) added/modified	Field(s)
Enhancement Request	Supports an additional record type for product enhancement requests.	Adds the EnhancementRequest record type.	**Fields included in EnhancementRequest record type:** Customer_Company Customer_Email Customer_Name Customer_Phone Customer_Priority Description Headline Keywords Owner Priority Product Product_Area Request_Type Submit_Date Submitter Target_Release
History (read-only)	Allows you to keep a historical account of all actions taken on a record.	Adds a History tab to the enabled record type.	No fields added.
Notes	Allows you to keep a historical account of all notes that have been entered on a record, according to date and user.	Adds a Notes tab to the enabled record type Also adds a base action called “Init_Note_Entry” in the enabled record type, which deletes the Note_Entry value.	**Fields added to enabled record type:** 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

Package	Description	Record type(s) added/modified	Field(s)
Project	Allows you to track records according to project. (**Note**: No relation to the UCM package “Project” concept.)	Creates a Project stateless record type.	**Fields included in Project record type:** 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	Record type(s) added/modified	Field(s)
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	Record type(s) added/modified	Field(s)
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
UCMPolicyScripts	Provides support for the UnifiedChangeManagement (UCM) package by adding three global scripts.	Does not add any record types.	

Package	Description	Record type(s) added/modified	Field(s)
UnifiedChange Management (UCM) (read-only)	Provides supports for the UCM process by enabling integration with Rational ClearCase 4.0 and above; links a ClearCase Project VOB with a ClearQuest user database. Requires the UCMPolicyScripts package. Can be used with the 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

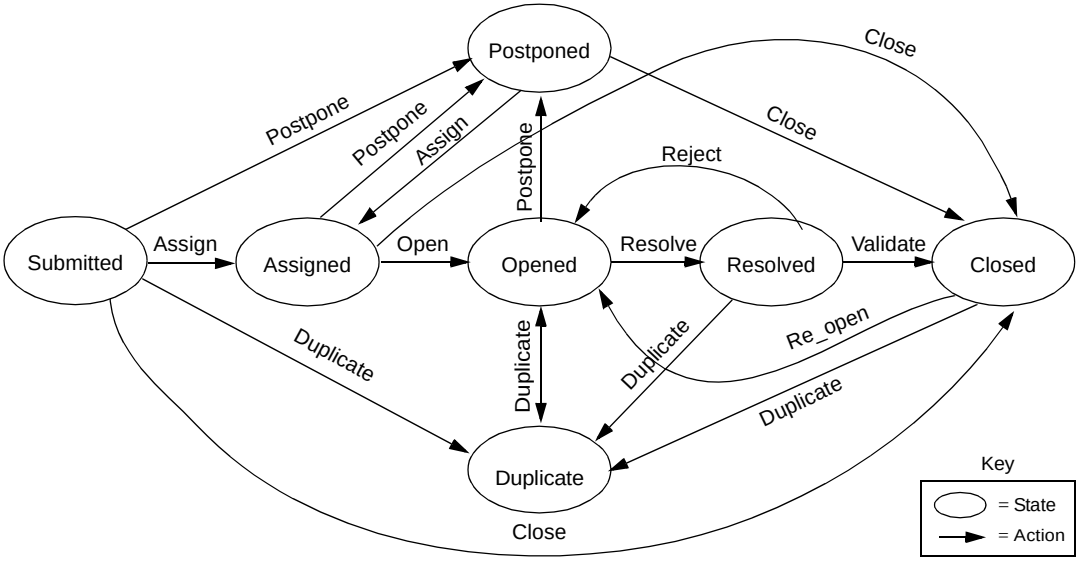
Package	Description	Record type(s) added/modified	Field(s)
Visual SourceSafe (read-only)	Provides support for integration with Microsoft Visual SourceSafe.	Adds SSObject and SCSnapObject stateless record types. Adds the SourceSafe tab to the form of the enabled record type.	**Fields added to enabled 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 Defect record type contains the following states:

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

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



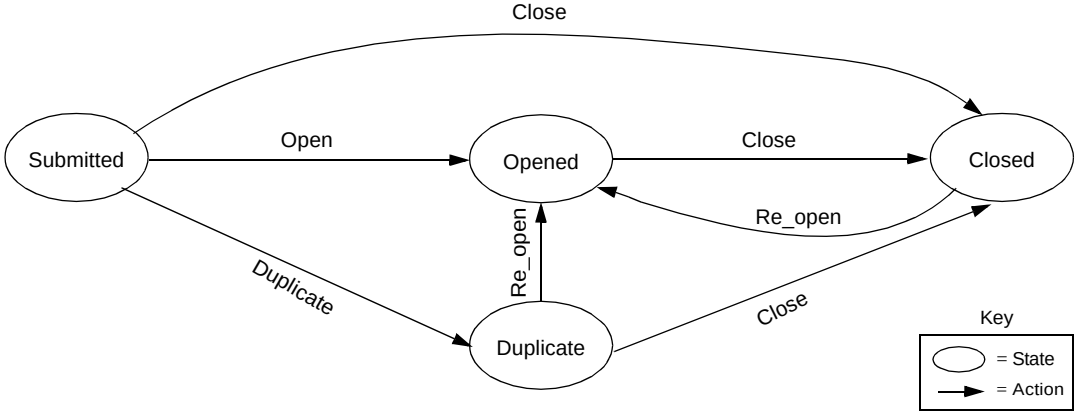
All Duplicate actions can be unduplicated by using the Unduplicate action.

State model for the EnhancementRequest record type

The EnhancementRequest record type contains the following states:

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

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



State-type models for packages

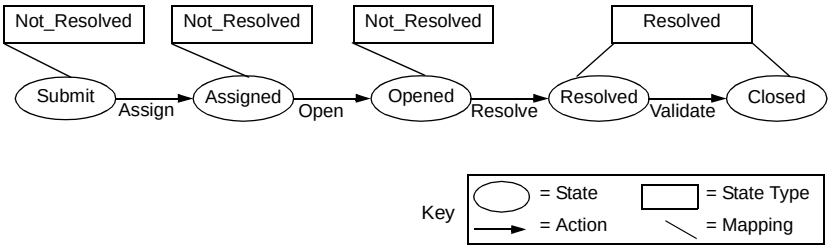
A state type is a label that defines a state's role in your state model. Some ClearQuest schemas have packages that require that each state in your state model be assigned or mapped to a specific state type. For example, the UnifiedChangeManagement schema and package use state types to invoke certain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

Note: When you add a new state to a schema that uses state types, you must map the new state to a state type. See "Mapping state types" on page 76.

Resolution package state type model

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

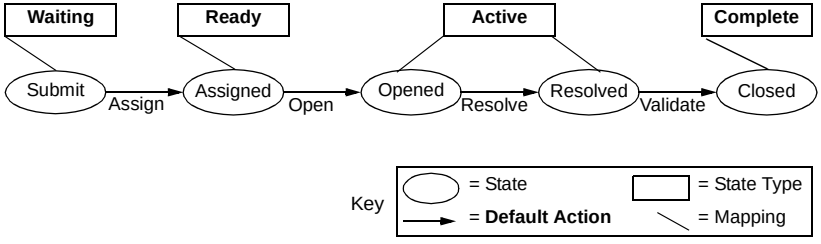
State type	Description
Not_Resolved	The record is not resolved.
Resolved	The record is resolved.



UnifiedChangeManagement package state type model

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

State type	Description	Mapped state in UCM schema
Waiting	Record not yet assigned, triaged, or scheduled	Submitted Postponed
Ready	Record has been assigned. It appears in the assigned user's To Do List query and is ready to be worked on. It may or may not have its ucm_project field set.	Assigned
Active	User has started working on the record, which is now associated with a ClearCase project and can contain ClearCase element information.	Opened
Complete	The record has been either: 1) Worked on and completed, which is associated with a ClearCase project and can contain ClearCase element information, or not worked on and abandoned. OR 2) Postponed or closed, which may or may not be associated with a ClearCase project.	Resolved Closed Duplicate



More information? See Appendix , "Adding ClearQuest Integrations" for more information.

B

Adding ClearQuest Integrations

You can integrate ClearQuest with software configuration management tools to create a complete change management system. For example, ClearQuest relates change requests directly to changes in your evolving code base, allowing you to answer such questions as:

- Which defects have been fixed in one release but not in another?
- Which defects can be attributed to a specific release?

This appendix lists and describes the integrations available with ClearQuest.

How are other applications integrated with ClearQuest?

ClearQuest is integrated with other software by adding ClearQuest packages to existing schemas. A schema contains the metadata that defines your workflow—record definitions, form and field definitions, record states and actions that can be performed on records, and optional hook code that customizes your workflow. Packages provide add-on functionality for existing schemas. For instance, a package might contain one or more new record types and might modify one or more record types that already exist in your schema. For information on adding packages, see “Adding packages to a schema” on page 47.

The difference between adding a package and adding an integration is:

- Adding a package adds specific functions to your schema, such as enabling attachments.
- Adding an integration not only adds more functions to your schema, it adds functions that allow ClearQuest to work directly with another application, such as Rational ClearCase. One or more packages may be needed for an integration, as well as additional configurations to each application.

Warning: You cannot uninstall or otherwise remove a package from a schema. Be sure to plan carefully before adding/installing a package into a schema.

Available integrations

ClearQuest integrates with many different products that require some kind of change to your existing schema repositories. Some are *independent integrations*, integrations which only require adding the appropriate package. Others are *dependent integrations*, integrations which not only require you to add one or more packages in a specific order, but may need additional configurations to ClearQuest.

Note: The instructions in this appendix assume that you are adding a new integration and that the necessary packages do not already exist in your schema. If you need to upgrade an integration or package, see *Installing Rational ClearQuest*.

Independent integrations

One general process can be used for all independent integrations. The independent integrations include the following:

- Rational Base ClearCase/ClearQuest
  Associates one or more ClearQuest change requests with one or more ClearCase versions.
- Rational Suite ContentStudio/ClearQuest
  Allows you to submit and track content changes with ClearQuest.
- Rational PureCoverage/ClearQuest
  Allows you to submit code coverage data to a ClearQuest database and track it.
- Rational Purify/ClearQuest
  Allows you to submit data to a ClearQuest database and track it.
- Rational Quantify/ClearQuest
  Allows you to submit performance data to a ClearQuest database and track it.

Dependent integrations

Dependent integrations may require multiple packages and configurations. The dependent integrations include the following:

- Rational Administrator/ClearQuest
  Associates Rational projects with ClearQuest databases.
- Rational ClearQuest Project Tracker/ClearQuest
  Allows you to exchange project data between the two systems.
- Rational RequisitePro/ClearQuest
  Associates RequisitePro requirements with ClearQuest records.
- Rational TeamTest/ClearQuest
  Allows you to submit defects found through TeamTest to ClearQuest databases, and to track them.
- Rational Unified Change Management (UCM)/ClearQuest
  Links ClearCase UCM projects and activities to ClearQuest records.
- Microsoft Visual SourceSafe/ClearQuest
  Associates Visual SourceSafe information with ClearQuest records.
- Your e-mail system/ClearQuest
  Enables ClearQuest to communicate with users through their e-mail systems.

Note: An e-mail system integration involves configuring Rational E-mail Reader and adding the e-mail notification package. This integration is described in *Administering ClearQuest E-mail* on page 181.

Upgrade integrations

If you are upgrading a package or an integration, you must first add any prerequisite packages needed to complete the upgrade. For more information on upgrading packages and integrations, see *Installing Rational ClearQuest*.

New record type integrations

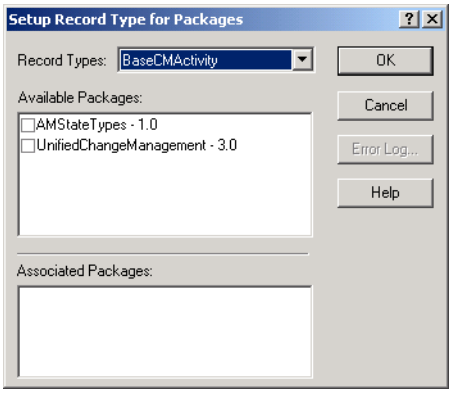
Some packages enable existing record types or add new ones. If you chose not to enable an existing record type, you can still do so later. Also, it is quite possible that you will add additional record types after adding a package. If new record types need functionality from a package, then you need to enable that functionality.

To enable package functionality for a record type:

1 Open your schema (**File > Open Schema**).

2 Select **Package > Setup Record Types for Package**.

The Setup Record Types for Package dialog box appears.



3 Select a record type from the **Record Types** drop-down list. The list contains only those record types that the package can modify.

4 From the **Available Packages** list, click the checkbox of the package(s) that you want the record type to modify.

The Available Packages list displays the installed packages that *have not been* modified by the chosen record type. The Associated Packages box lists the installed packages that *have been* modified by the chosen record type.

Analyzing your current integration

Many packages are used by multiple integrations. Before you add any integrations to your schema repository, you should find out what packages have already been added. If you already have a required package, you may need to upgrade it. See *Installing Rational ClearQuest* for instructions on upgrading packages and integrations.

Different record types can be modified by adding a package, and you may choose to only modify some. New record types can be created after a package is added. For these two reasons, you may need to apply packages to specific record types.

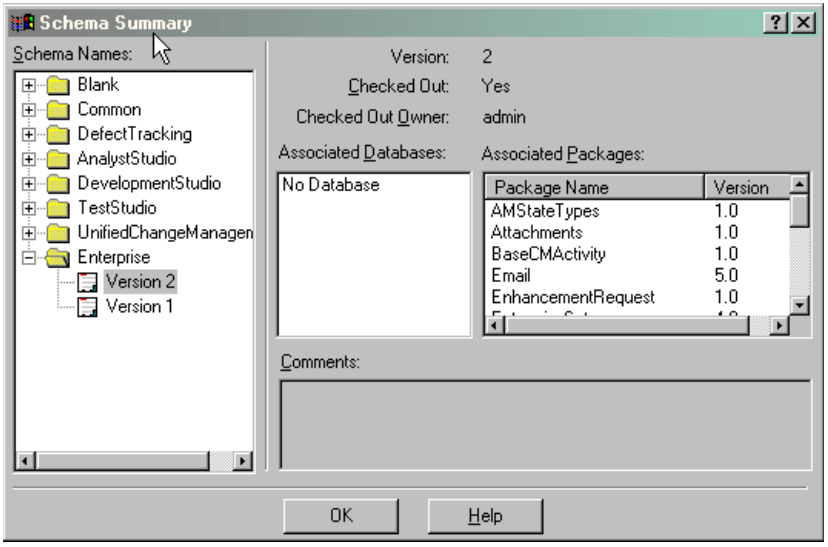
Note: You cannot add a package to the same schema more than once. The schema's metadata must be different than the package's metadata. For example, if you try to add a package containing a "Project" record type to a schema that already has a "Project" record type, it will fail. Also, the schema must have metadata that the package wants to modify. For example, if you try to add a package that modifies the "Project" record type, but the schema does not already have that record type, then it will again fail.

Viewing the packages in your schema

To view a list of packages already in your schema:

1 Choose **View > Schema Summary.**

2 Open the folder with your schema name and select the latest version. If multiple databases are associated with this schema they will be displayed in the central list display.



The list in the right scroll area will display every package with the name and version that have been added to your schema.

3 If the package(s) you need to add already exist in the **Associated Packages** list, write down the version of the package(s).

Viewing available packages for record types

As discussed in “New record type integrations” on page 229, you may have record types that have not had package functionality applied to them. You can view the available packages for a record type and find out if they have been applied to it.

To view which record types have package functionality enabled:

1. Open your schema (**File > Open Schema**).
2. Select **Package > Setup Record Types for Package**.

   The Setup Record Types for Package dialog box appears. The **Record Types** list contains only those record types that the package can modify. The **Available Packages** list displays the installed packages that *have not been* applied to the selected record type. The **Associated Packages** box lists the installed packages that *have been* applied to the selected record type.

   To enable package functionality for a record type, see “New record type integrations” on page 229.

Conducting a test integration

Once you check in changes to a schema and apply schema changes to your user database, you cannot undo the changes. Therefore, it is highly recommended that you add your integration in a test environment before modifying your production environment.

When performing a test integration, replicate both production environments—both ClearQuest's and that of the application being integrated with ClearQuest. For example, in a ClearQuest/ClearCase integration, you need 1) a test ClearQuest database, 2) a test ClearQuest schema, and 3) a test ClearCase VOB. All must duplicate the production environment. Never use the actual production environment in a test integration.

By performing a test integration, you do not risk permanent damage to your production environments.

More information? To learn more about using test databases, see "Working with a test database" on page 37.

Adding independent integrations

Independent integrations require you to simply add the appropriate package. You must have Super User or Schema Designer user privileges to add packages.

These instructions assume that all changes to your schema have been *checked in* (**File > Check In**). For more information about schemas and packages, see *Overview of working with schemas* on page 36.

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 "Conducting a test integration" on page 233.

Warning: You cannot uninstall or otherwise remove a package from a schema. Be sure to plan carefully before adding/installing a package into a schema.

Note: The instructions in this appendix assume that you are adding a new integration and that the necessary packages do not already exist in your schema. If you need to upgrade an integration or package, see *Installing Rational ClearQuest*.

Adding your integration

See "Independent integrations" on page 227 for a list of independent integrations and a brief description of each.

To add your integration:

1. Find out if the package you need already exists in your schema (see "Analyzing your current integration" on page 230). Then do one of the following:
   - If the package you need is already in your schema, and you only need to apply the package to a new record type, see "New record type integrations" on page 229 for instructions. Then continue to the next step in this section.

- If the package does not exist in your schema, add the latest version of the package listed in the following table for the integration you want (**Package > Package Wizard**). See "Adding packages to a schema" on page 47 for detailed instructions.

Integration	Package to add	Additional documentation
Rational ClearCase (Base ClearCase)	ClearCase	ClearCase online help. Go to the table of contents tab and browse to: ClearCase User Interface > Integrations with other products > Base ClearCase integration with ClearQuest.
Rational Purify	PQC	Rational Purify online help. Look up *ClearQuest*.
Rational Quantify	PQC	Rational Quantify online help. Look up *ClearQuest*.
Rational PureCoverage	PQC	Rational PureCoverage online help. Look up *ClearQuest*.
Rational Suite ContentStudio	ContentStudio	*Installing Rational Suite ContentStudio*

2. Save your changes to the schema by checking it in (**File > Check In**). See "Checking in a schema" on page 43 for detailed instructions.

3. Apply schema changes to the user database (**Database > Upgrade Database**). See "Applying schema changes to a user database" on page 44 for detailed instructions.

4. Configure the integrated application as needed. See the application's documentation for additional configuration needs. The specific documentation is listed in the previous table.

   Note: The ClearCase integration discussed in this section is a Base ClearCase integration; there are no predefined policies. Policies must be set up by the ClearCase administrator. The UCM integration ("Adding a Rational UCM integration" on page 247) sets up predefined ClearCase policies automatically.

Adding dependent integrations

Dependent integrations may require multiple packages and configurations. These integrations require packages to be added or configurations to be made in a specific order. You must have Super User or Schema Designer user privileges to add packages.

If you do not add packages in the specified order, an Error Log dialog box appears displaying the error(s) associated with your action. At that point, you can view the error(s) and/or save them to a file. To avoid errors in the Error Log, read the instructions carefully in these sections and follow them exactly as described. See “Analyzing your current integration” on page 230 for additional error causes.

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 “Conducting a test integration” on page 233.

These instructions assume that all changes to your schema have been *checked in* (**File > Check In**). For more information about schemas and packages, see *Working with ClearQuest Schemas* on page 35.

Warning: You cannot uninstall or otherwise remove a package from a schema. Be sure to plan carefully before adding/installing a package into a schema.

Note: The instructions in this appendix assume that you are adding a new integration and that the necessary packages do not already exist in your schema. If you need to upgrade an integration or package, see *Installing Rational ClearQuest*.

The dependent integrations include:

- Rational Administrator, see page 237
- Rational ClearQuest Project Tracker, see page 239
- Rational RequisitePro, see page 241
- Rational TeamTest, see page 245

- Rational UCM, see page 247
- Microsoft Visual SourceSafe, see page 255

Adding a Rational Administrator integration

Integrating with Rational Administrator associates Rational projects with ClearQuest databases. To add the integration:

- "Step 1: Find out what packages you already have" on page 237
- "Step 2: Add the Repository package" on page 237
- "Step 3: Apply schema changes to your user database" on page 238
- "Step 4: Configure Rational Administrator" on page 238

Step 1: Find out what packages you already have

If you already have an earlier version of the package you need in your schema, you need to use upgrade procedures for adding the latest package.

To find out if the Repository package already exists in your schema, see "Analyzing your current integration" on page 230.

Step 2: Add the Repository package

If you already have a Repository package in your schema, and you just want to apply it to a new record type, see "New record type integrations" on page 229 for instructions. Then continue to "Step 3: Apply schema changes to your user database" on page 238.

If you do not have a Repository package in your schema:

1. Add the **Repository 0.2** package (**Package > Package Wizard**). See "Adding packages to a schema" on page 47 for detailed instructions.
2. Save your changes to the schema by checking it in (**File > Check In**). See "Checking in a schema" on page 43 for detailed instructions.
3. Add the latest **Repository** package (**Package > Package Wizard**).

4 Save your changes to the schema by checking it in (**File > Check In**).

Step 3: Apply schema changes to your user database

Apply schema changes to the user database (**Database > Upgrade Database**). See “Applying schema changes to a user database” on page 44 for detailed instructions.

Step 4: Configure Rational Administrator

Configure the integrated application as needed. See Rational Administrator online help for additional configuration information.

Adding a Rational ClearQuest Project Tracker integration

Integrating with Rational ClearQuest Project Tracker allows you to exchange project data between Project Tracker and ClearQuest. To add this integration:

Step 1: Find out what packages you already have

If you already have an earlier version of the package you need in your schema, you need to use upgrade procedures for adding the latest package.

To find out if the AMBaseActivity and AMWorkActivitySchedule packages already exist in your schema, see "Analyzing your current integration" on page 230.

Step 2: Add the AMBaseActivity package

If you already have an AMBaseActivity package in your schema, and you just want to apply it to a new record type, see "New record type integrations" on page 229 for instructions. Then continue to "Step 3: Add the AMWorkActivitySchedule package" on page 240.

If you do not have an AMBaseActivity package in your schema:

1. Add the latest **AMBaseActivity** package (**Package > Package Wizard**). See "Adding packages to a schema" on page 47 for detailed instructions.
2. Save your changes to the schema by checking it in (**File > Check In**). See "Checking in a schema" on page 43 for detailed instructions.

Step 3: Add the AMWorkActivitySchedule package

If you already have an AMWorkActivitySchedule package in your schema, and you just want to apply it to a new record type, see "New record type integrations" on page 229 for instructions. Then continue to "Step 4: Apply schema changes to your user database" on page 240.

If you do not have an AMWorkActivitySchedule package in your schema:

1. Add the latest AMWorkActivitySchedule package (**Package > Package Wizard**). See "Adding packages to a schema" on page 47 for detailed instructions.
2. Save your changes to the schema by checking it in (**File > Check In**). See "Checking in a schema" on page 43 for detailed instructions.

Step 4: Apply schema changes to your user database

Apply schema changes to the user database (**Database > Upgrade Database**). See "Applying schema changes to a user database" on page 44 for detailed instructions.

Step 5: Configure Rational ClearQuest Project Tracker

Configure the integrated application as needed. See *Using Rational ClearQuest Project Tracker* for instructions on linking your project plans with ClearQuest databases and other integrated tasks.

Adding a Rational RequisitePro integration

Integrating with Rational RequistePro associates RequisitePro requirements with ClearQuest records. To add this integration:

Step 1: Find out what packages you already have

If you already have an earlier version of the package you need in your schema, you need to use upgrade procedures for adding the latest package.

To find out if the Repository and RequisitePro packages already exist in your schema, see "Analyzing your current integration" on page 230.

Step 2: Add the Repository package

If you already have a Repository package in your schema, and you just want to apply it to a new record type, see "New record type integrations" on page 229 for instructions. Then continue to "Step 3: Add the RequisitePro package" on page 242.

If you do not have a Repository package in your schema:

1. Add the **Repository 0.2** package (**Package > Package Wizard**). See "Adding packages to a schema" on page 47 for detailed instructions.
2. Save your changes to the schema by checking it in (**File > Check In**). See "Checking in a schema" on page 43 for detailed instructions.
3. Add the latest **Repository** package (**Package > Package Wizard**).

4 Save your changes to the schema by checking it in (**File > Check In**).

Step 3: Add the RequisitePro package

If you already have a RequisitePro package in your schema, and you just want to apply it to a new record type, see "New record type integrations" on page 229 for instructions. Then continue to "Step 5: Apply schema changes to your user database" on page 244.

If you do not have a RequisitePro package in your schema:

1 Add the latest **RequisitePro** package (**Package > Package Wizard**). See "Adding packages to a schema" on page 47 for detailed instructions.

2 Save your changes to the schema by checking it in (**File > Check In**). See "Checking in a schema" on page 43 for detailed instructions.

Step 4: Update record types with back reference

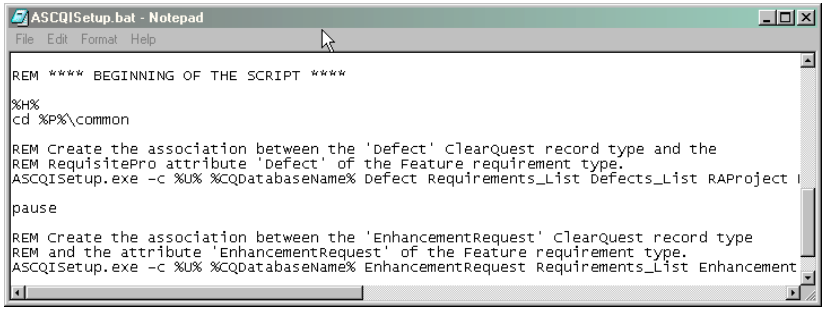
For every record type that you have applied the RequisitePro package, you need to locate the Requirements_List field and add a back reference field to it.

Naming your back reference

The name you use for the back reference must be consistent with what is found in the `ASCQISetup.bat` file located in:

```
DriveLetter:\[Rational Installation]\common
```

For example, if you look at the following portion of the `ASCQISetup.bat` file, you will see something like this:



The line starting with "ASCQISetup.exe" includes [Defect], [Requirements_List], and [Defects_List]. These are translated as follows:

Defect	=	ClearQuest record type
Requirements_List	=	The RequistePro field added to the Defect record type
Defects_List	=	The back_reference in the Requirement_List field.

Write down the back reference name(s) you need and then continue to "Adding the back reference."

Adding the back reference

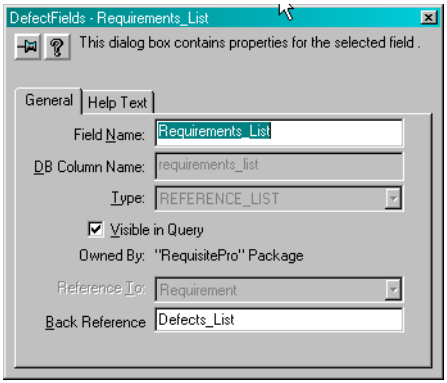
The RequisitePro integration requires you to specify a field as a back reference to the Requirements_List field for each of the record types you selected when you applied the RequisitePro package, such as Defect and Enhancement Request.

To add a back reference field to your schema:

1. In the Workspace, expand the **Record Types** folder for the record type with RequisitePro integration, and expand the **Fields** folder. The list of fields will be displayed in the right panel.
2. Right-click the **Requirements_List** field name and select **Field Properties.** The properties dialog box appears.

3 At the **Back Reference** field, enter the name you identified in “Naming your back reference” on page 242. For some schemas, this field may already contain a value.

See also “Linking records to create a parent/child hierarchy” on page 70 for more information about back references.



4 Save the change by clicking the pushpin button in the properties dialog box. Then close the window.

5 Repeat step 2 through step 4 for each of the record types you selected when you applied the RequisitePro package. Each record should have an entry in the `ASCQISetup.bat` file. See “Naming your back reference” on page 242.

6 From the ClearQuest Designer menu bar, select **File > Checkin** to check these changes into your schema. Click **OK** after verifying your choice. Click **OK** again in the ClearQuest **Checkin Confirmation** dialog box.

Step 5: Apply schema changes to your user database

Apply schema changes to the user database (**Database > Upgrade Database**). See “Applying schema changes to a user database” on page 44 for detailed instructions.

Step 6: Configure Rational RequisitePro

Configure the integrated application as needed. See the Rational RequisitePro online help for additional configuration information.

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. To add this integration:

Step 1: Find out what packages you already have

If you already have an earlier version of the package you need in your schema, you need to use upgrade procedures for adding the latest package.

To find out if the TeamTest and Repository packages already exist in your schema, see "Analyzing your current integration" on page 230.

Step 2: Add the Repository package

If you already have a Repository package in your schema, and you just want to apply it to a new record type, see "New record type integrations" on page 229 for instructions. Then continue to "Step 3: Add the TeamTest package" on page 246.

If you do not have a Repository package in your schema:

1. Add the **Repository 0.2** package (**Package > Package Wizard**). See "Adding packages to a schema" on page 47 for detailed instructions.
2. Save your changes to the schema by checking it in (**File > Check In**). See "Checking in a schema" on page 43 for detailed instructions.
3. Add the latest **Repository** package (**Package > Package Wizard**).

4 Save your changes to the schema by checking it in (**File > Check In**).

Step 3: Add the TeamTest package

If you already have a TeamTest package in your schema, and you just want to apply it to a new record type, see “New record type integrations” on page 229 for instructions. Then continue to “Step 4: Apply schema changes to your user database” on page 246.

If you do not have a TeamTest package in your schema:

1 Add the latest **TeamTest** package (**Package > Package Wizard**). See “Adding packages to a schema” on page 47 for detailed instructions.

2 Save your changes to the schema by checking it in (**File > Check In**). See “Checking in a schema” on page 43 for detailed instructions.

Step 4: Apply schema changes to your user database

Apply schema changes to the user database (**Database > Upgrade Database**). See “Applying schema changes to a user database” on page 44 for detailed instructions.

Step 5: Configure Rational TeamTest

Configure the integrated application as needed. See Rational TeamTest online help for additional configuration information.

Adding a Rational UCM integration

The Unified Change Management (UCM) integration links ClearCase UCM projects and activities to ClearQuest records. This is also known as a UCM-ClearCase integration.

This integration requires the following:

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

ClearQuest provides two predefined schemas that support UCM: the UnifiedChangeManagement schema and the Enterprise schema. The easiest way to implement UCM is to use one of these schemas. For instructions on using schemas, see *Working with ClearQuest Schemas* on page 35.

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 234 for adding a Base ClearCase integration.

Integrating Rational UCM with packages involves the following steps:

- "Step 1: Find out what packages you already have" on page 248
- "Step 2: Add the AMStateTypes package" on page 248
- "Step 3: Set the default actions for UCM" on page 250
- "Step 4: Add the UCMPolicyScripts package" on page 253
- "Step 5: Add the UnifiedChangeManagement package" on page 253
- "Step 6: Add the BaseCMActivity package (optional)" on page 254

- "Step 7: Apply schema changes to your user database" on page 254
- "Step 8: Configure Rational UCM" on page 254

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

Step 1: Find out what packages you already have

If you already have an earlier version of the package you need in your schema, you need to use upgrade procedures for adding the latest package.

To find out if AMStateType, UCMPolicyScripts, UnifiedChangeManagement, and BaseCMActivity already exist in your schema, see "Analyzing your current integration" on page 230. BaseCMActivity is an optional package.

Step 2: Add the AMStateTypes package

If you already have an AMStateType package in your schema, and you just want to apply it to a new record type, see "New record type integrations" on page 229 for instructions. Then continue to "Step 4: Add the UCMPolicyScripts package" on page 253.

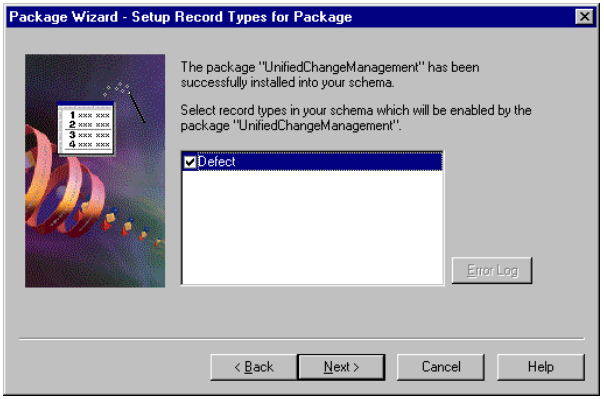
If you do not have an AMStateType package in your schema:

1. Add the latest **AMStateType** package (**Package > Package Wizard**). See "Adding packages to a schema" on page 47 for detailed instructions.

   This package requires you to map state types and define default actions, if they have not already been defined. When you enable record types and move to the next dialog box (Setup State Types dialog box), the Package Wizard will prompt you to map the new state types and define default actions.

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

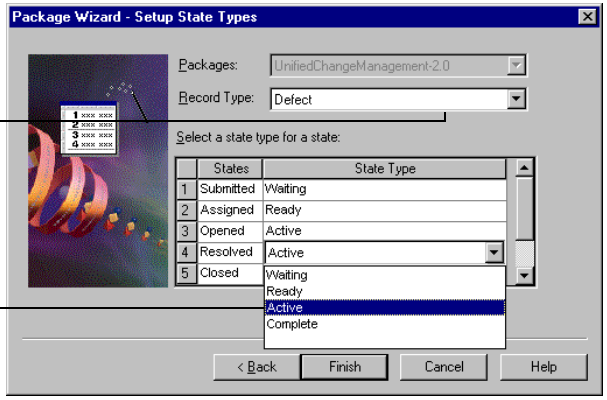
Note: When enabling a record type for UCM packages, a separate submit form is required. (When the BaseCMActivity record is UCM-enabled, a UCM tab is added to the record type.)



3 In the Setup State Types dialog box, map the states in your schema to the UCM state types. Click in the **State Types** field for each state and select the appropriate UCM state type. For more details regarding mapping state types, see “Mapping state types” on page 76.

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

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



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

ClearQuest automatically tries to validate your schema with the new changes. The validation window will indicate that you need to set default actions. Continue to the next step to do so.

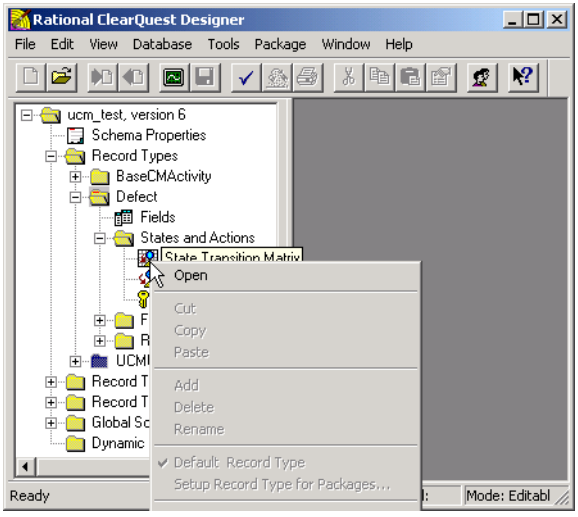
Step 3: Set the default actions for UCM

The State Transition Matrix in your schema must provide at least one path through the state type model for the UnifiedChangeManagement package, from the Waiting state type, to Ready, to Active, to Complete. (See "Using the State Transition Matrix" on page 75 and "UnifiedChangeManagement package state type model" on page 223.)

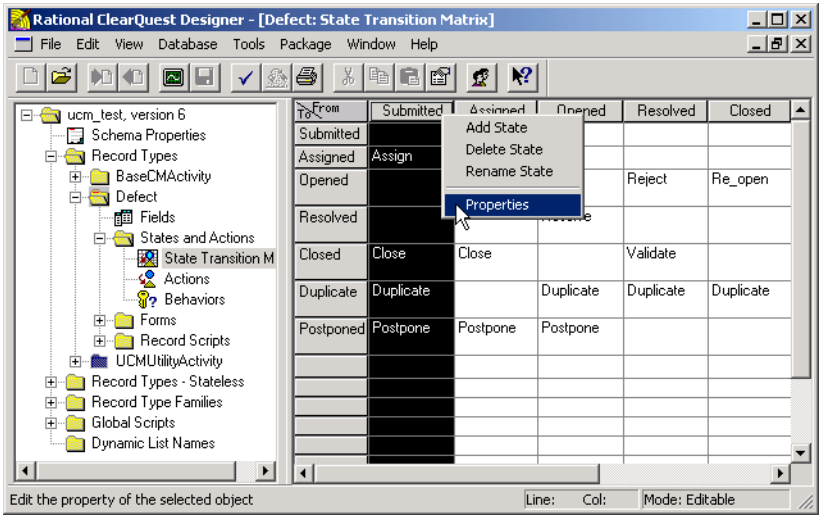
For each state in your schema, except the state mapped to the UCM Complete state, you must assign a default action that moves the record from that state to the next state type in the UCM state type model. See "Using default actions" on page 84.

To assign default actions:

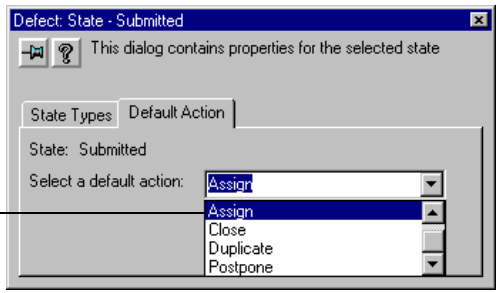
1 Open the **State Transition Matrix** for the record type you enabled for UCM.



2 Right-click a state and select **Properties** to open the Properties dialog 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 Save your changes to the schema by checking it in (**File > Check In**). See “Checking in a schema” on page 43 for detailed instructions.

More information? See “Using default actions” on page 84 and “UnifiedChangeManagement package state type model” on page 223. Look up *actions, default* in the ClearQuest Designer Help index.

Step 4: Add the UCMPolicyScripts package

If you already have a UCMPolicyScripts package in your schema, and you just want to apply it to a new record type, see “New record type integrations” on page 229 for instructions. Then continue to “Step 5: Add the UnifiedChangeManagement package” on page 253.

If you do not have a UCMPolicyScripts package in your schema:

1. Add the latest **UCMPolicyScripts** package (**Package > Package Wizard**). See “Adding packages to a schema” on page 47 for detailed instructions.
2. Save your changes to the schema by checking it in (**File > Check In**). See “Checking in a schema” on page 43 for detailed instructions.

Step 5: Add the UnifiedChangeManagement package

This package requires you to associate state types and set default actions. This section guides you through the Package Wizard configuration.

If you already have a UnifiedChangeManagement package in your schema, and you just want to apply it to a new record type, see “New record type integrations” on page 229 for instructions. Then continue to “Step 6: Add the BaseCMActivity package (optional)” on page 254.

If you do not have a UnifiedChangeManagement package in your schema:

1. Add the latest UnifiedChangeManagement package **(Package > Package Wizard)**.
2. Save your changes to the schema by checking it in (**File > Check In**). See “Checking in a schema” on page 43 for detailed instructions.

Step 6: Add 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 *Using Rational ClearQuest Project Tracker*.

If you do not have a BaseCMActivity package in your schema:

1. Add the latest **BaseCMActivity** package (**Package > Package Wizard**). See "Adding packages to a schema" on page 47 for detailed instructions.
2. Save your changes to the schema by checking it in (**File > Check In**). See "Checking in a schema" on page 43 for detailed instructions.

Step 7: Apply schema changes to your user database

Apply schema changes to the user database (**Database > Upgrade Database**). See "Applying schema changes to a user database" on page 44 for detailed instructions.

Step 8: Configure Rational UCM

Configure the integrated application as needed. See *Managing Software Projects with ClearCase* for additional configuration information.

Adding a Microsoft Visual SourceSafe integration

The Visual SourceSafe integration adds Visual SourceSafe fields to a record type you choose, and the Visual SourceSafe tab to the record type form.

To add this integration:

- “Step 1: Find out what packages you already have” on page 255
- “Step 2: Add the Visual SourceSafe package” on page 256
- “Step 3: Apply schema changes to your user database” on page 256
- “Step 4: Create a query” on page 256
- “Step 5: Set up each client machine” on page 257

For instructions on using the integration after completing these steps, see the Help for the ClearQuest Visual SourceSafe tool.

Step 1: Find out what packages you already have

If you already have an earlier version of the package you need in your schema, you need to use upgrade procedures for adding the latest package.

To find out if the Visual SourceSafe package already exists in your schema, see “Analyzing your current integration” on page 230.

Step 2: Add the Visual SourceSafe package

If you already have a Visual SourceSafe package in your schema, and you just want to apply it to a new record type, see “New record type integrations” on page 229 for instructions. Then continue to “Step 3: Apply schema changes to your user database” on page 256.

If you do not have a Visual SourceSafe package in your schema:

1. Add the latest **Visual SourceSafe** package (**Package > Package Wizard**). See “Adding packages to a schema” on page 47 for detailed instructions.

   Note: You can only apply this package to one record type per schema.

2. Save your changes to the schema by checking it in (**File > Check In**). See “Checking in a schema” on page 43 for detailed instructions.

Step 3: Apply schema changes to your user database

Apply schema changes to the user database (**Database > Upgrade Database**). See “Applying schema changes to a user database” on page 44 for detailed instructions.

Step 4: Create a query

You need to create a query that users can use to find the records you want to associate with Visual SourceSafe projects.

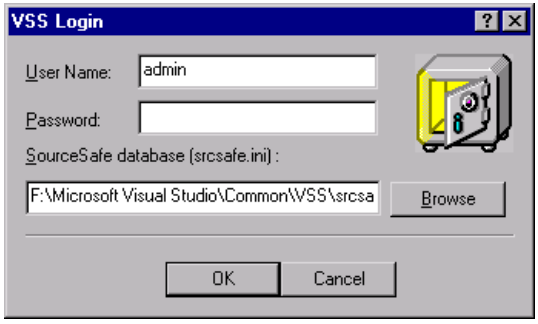
To create a query:

1. Log in to ClearQuest as a user with public folder privileges.
2. Define the query. (See the ClearQuest online help for instructions.)

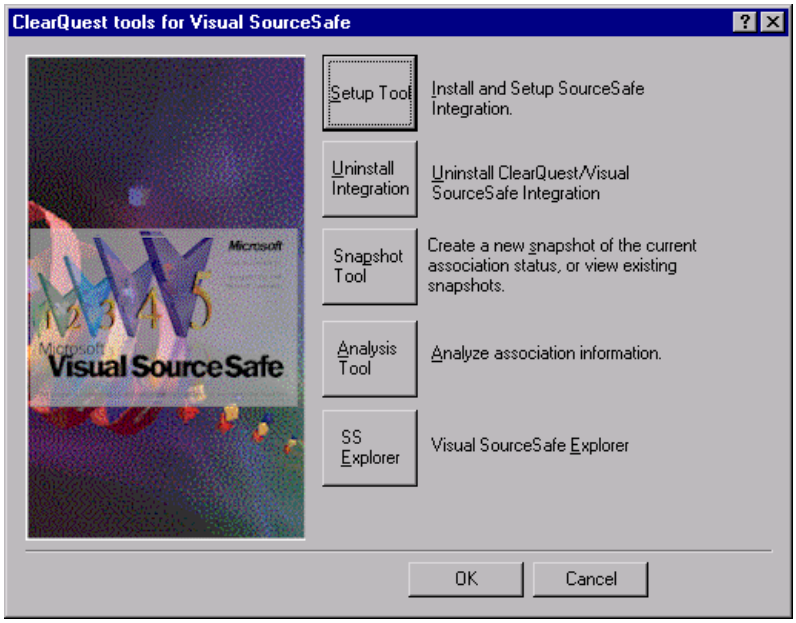
Step 5: Set up each client machine

On each client machine that wants to use the integration:

1. Launch 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. From the ClearQuest tools for Visual SourceSafe dialog box, select **Setup Tool**.



4 In the Visual SourceSafe Integration dialog box, select the ClearQuest query you created in “Step 4: Create a query” on page 256 and click **OK**.

Index

C

D

E

F

G

H

I

K

L

M

N

O

S

T

U

V

W