

# Using Rational TestManager

Rational ClearQuest/TeamTest Edition  
Release 7.5

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

**Rational**  
software

## Using Rational TestManager

Copyright © 1999 Rational Software Corporation. All rights reserved. The contents of this manual and the associated software are the property of Rational Software Corporation and are copyrighted. Any reproduction in whole or in part is strictly prohibited. For additional copies of this manual or software, please contact Rational Software Corporation.

Rational, the Rational logo, PerformanceStudio, SiteCheck, TestFactory, TestStudio, Object-Oriented Recording, and Object Testing are trademarks or registered trademarks of Rational Software Corporation in the United States and in other countries. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All other names are used for identification purposes only and are trademarks or registered trademarks of their respective companies.

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

Revised 10/1999

This manual prepared by:  
Rational Software Corporation  
20 Maguire Road  
Lexington, MA 02421  
U.S.A.

Phone:  
800-433-5444  
408-863-4000

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

# ▶▶▶ Contents

About This Manual . . . . .	v
Contacting Rational Technical Support . . . . .	vi
Documentation. . . . .	viii
Using Help . . . . .	x
Typographical Conventions. . . . .	xii
<b>1 Introduction to Rational TestManager</b>	
Starting Rational TestManager . . . . .	1-2
<b>2 Planning Your Tests</b>	
Working with Test Plans and Other Test Documents . . . . .	2-1
Defining Test Requirements . . . . .	2-4
Planning Scripts . . . . .	2-9
Managing Requirements . . . . .	2-17
Customizing Scripts. . . . .	2-18
<b>3 Managing Builds, Log Folders, and Logs</b>	
Overview. . . . .	3-1
Displaying Builds in the Asset Browser . . . . .	3-3
Creating a New Build . . . . .	3-5
Copying, Renaming, and Deleting Builds . . . . .	3-6
Renaming and Deleting Logs and Log Folders. . . . .	3-8
Displaying Log Properties . . . . .	3-9
Viewing Logs. . . . .	3-9
Working with Build States. . . . .	3-10
<b>4 Querying the Rational Repository</b>	
Overview. . . . .	4-1
Running Queries . . . . .	4-2
Editing Existing Queries . . . . .	4-9
Setting Query Options. . . . .	4-9
Configuring the Query Window. . . . .	4-10

## **5 Running TestManager Reports**

Types of Reports . . . . .	5-1
Listing Reports . . . . .	5-2
Selecting Reports to Use . . . . .	5-3
Working with Listing Reports . . . . .	5-4
Working with Coverage Reports. . . . .	5-6
Working with the Test Results Progress Report . . . . .	5-16
Copying, Renaming, and Deleting Reports . . . . .	5-18

## ▶ ▶ ▶ Preface

The information in this *Using Rational TestManager* manual is intended for all members of your development team — development engineers, test engineers, project leaders, and managers. You'll find out how to use TestManager to plan tests, manage builds, create queries for use with the Rational repository, and run reports.

## About This Manual

---

This manual contains the following chapters:

**Chapter 1: Introduction to Rational TestManager**

**Chapter 2: Planning Your Tests** – Describes the planning phase of your testing effort. It includes information about writing test plans and other documents, defining and managing test requirements, and planning and customizing scripts.

**Chapter 3: Managing Builds, Logs, and Log Folders** – Explains the use of builds in functional testing, ways of displaying builds in TestManager, and how to perform various actions on builds, logs, and log folders.

**Chapter 4: Querying the Rational Repository** – Explains how to create and run queries. It includes information about configuring the query window and setting query and filter options.

**Chapter 5: Running TestManager Reports** – Explains how to run and create various kinds of reports.

## Contacting Rational Technical Support

---

If your site has a designated, on-site support person, please try to contact that person before contacting Rational Technical Support. If you still need assistance, you can use the following information to contact Rational Technical Support.

### When Contacting Rational Technical Support

When contacting Rational Technical Support, please be prepared to supply the following information:

- ▶ Your name, telephone number, and company name
- ▶ Computer make and model
- ▶ Operating system and version number
- ▶ Product release number
- ▶ Your Case ID number (if you are calling about a previously reported problem)

### How To Contact Rational Technical Support

Rational Technical Support can provide information and assistance by:

- ▶ Telephone
- ▶ Electronic Mail
- ▶ Fax
- ▶ World Wide Web

#### Telephone and E-Mail

Telephone and e-mail support is available Monday through Friday (except holidays) in four major Rational Technical Support call centers around the world. Specific call center contact information is located in the following section, *Rational Technical Support Call Center Information*, as well as on our Web site at <http://www.rational.com/support/contact>.

When contacting Technical Support via e-mail, please include the information listed in the previous section, *When Contacting Rational Technical Support*, along with a detailed description of your problem. Upon receipt of your request, Rational Technical Support will send you an electronic response with your Case ID number and the point of contact for your issue. When sending e-mail concerning a previously reported problem, please include in the subject field “re: Case XXXXX,” substituting your assigned support Case ID for XXXXX.

## **Fax**

Rational Technical Support Engineers will sometimes ask you to fax information to help them diagnose problems. Please mark faxes “Attention: Technical Support” and add your fax number to the information listed in the previous section.

## **World Wide Web**

You can also contact Rational Technical Support through our Web site at <http://www.rational.com/support/contact>.

## **Rational Technical Support Call Center Information**

You can use the following addresses and telephone numbers to contact a Rational Technical Support call center in your area.

- ▶ **North America**  
18880 Homestead Road  
Cupertino, CA 95014  
  
20 Maguire Road  
Lexington, MA 02421  
  
Telephone: 800-433-5444 or 408-863-4000  
E-mail: [support@rational.com](mailto:support@rational.com)
- ▶ **Europe**  
Beechavenue 30  
NL-1119 PV Schiphol-Rijk  
The Netherlands  
  
Telephone: +31-20-4546-200  
E-mail: [support@europe.rational.com](mailto:support@europe.rational.com)
- ▶ **Asia Pacific**  
Level 13, 821 Pacific Highway  
Chatswood NSW 2067  
Sydney, Australia  
  
Telephone: +61-2-9419-0111  
E-mail: [support@apac.rational.com](mailto:support@apac.rational.com)

## Documentation

---

This product includes printed manuals, online manuals, and online Help. The documentation that you receive depends on the products that you have purchased.

### Printed Manuals

This product includes some or all of the following printed manuals:

**New Features** – Describes the new features in this release.

**Using the Rational Administrator** – Provides information about the Rational repository and explains how to use the Rational Administrator.

**Getting Started with Rational Robot** – Introduces you to Rational Robot and includes a tutorial that shows you how to use Robot for your own testing projects.

**Using Rational Robot** – Describes the concepts and procedures you need to use Rational Robot to plan tests, develop and play back GUI scripts, analyze your results, and run reports and queries.

**Getting Started with Rational TestFactory** – Introduces you to Rational TestFactory and includes a tutorial that shows you how to use TestFactory for your own testing projects.

**Using Rational TestFactory** – Describes how to use Rational TestFactory to map the application-under-test and automatically generate scripts that test your product. Also describes how to use Rational TestAccelerator to conduct remote distributed testing.

**Getting Started with Rational Suite PerformanceStudio** – Introduces you to Rational Suite PerformanceStudio and includes installation and licensing instructions.

**Using Rational LoadTest** – Describes the concepts and procedures you need to plan performance tests and distributed functional tests, record sessions, design workloads, and analyze your results.

**Installing Rational TeamTest and Rational Robot** – Describes the hardware and software requirements for Rational TeamTest and Rational Robot and includes complete installation instructions.

**SQABasic Language Reference** – Describes the commands and conventions of the SQABasic scripting language.

**VU Language Reference** – Describes the commands and conventions of the VU scripting language.



**Rational Robot: Try it! with Rational SiteCheck** – Describes how to use Rational SiteCheck to test the structural integrity of your intranet or World Wide Web site.

**Rational Robot: Try it! with HTML** – Shows you how to use Rational Robot to test your HTML pages.

**Rational Robot: Try it! with Java** – Shows you how to use Rational Robot to test Java applets and applications.

**Rational Robot: Try it! with Oracle Forms** – Shows you how to use Rational Robot to test objects in your Oracle applications.

**Rational Robot: Try it! with PowerBuilder** – Shows you how to use Rational Robot to test the controls in your PowerBuilder applications.

**Rational Robot: Try it! with Visual Basic** – Shows you how to use Rational Robot to test the controls in your Visual Basic applications.

## Online Manuals

All of the manuals for this product are available online as Portable Document Format (PDF) files. You can use the Adobe Acrobat Reader to read, search, and print these files.

### Installing the Adobe Acrobat Reader

This product includes the Adobe Acrobat Reader. To install the Acrobat Reader from the Rational Online Documentation CD:

1. Insert the Online Documentation CD into your CD drive.
2. If a Web browser does not open, double-click `autorun.htm` on the CD to open the Online Documentation page.
3. Click the Acrobat Reader link at the beginning of the page.
4. Do one of the following:
  - To install the Acrobat Reader directly from the CD, click **Run this program from its current location** and click **OK**. Follow the on-screen instructions to complete the installation.
  - To save the Acrobat Reader installation file to your hard disk, click **Save this program to disk**, click **OK**, and save the file. Then locate the file `arnnn.exe` (where *nnn* is the version of the Acrobat Reader) on your hard disk. Double-click the file and follow the on-screen instructions to complete the installation.

**NOTE:** You can also download the reader from the Adobe Web site:  
<http://www.adobe.com>.

## Opening a PDF File

To open a PDF file on the Online Documentation CD:

1. Install the Adobe Acrobat Reader. For instructions, see the previous section, *Installing the Adobe Acrobat Reader*.
2. Insert the Online Documentation CD into your CD drive.
3. If a Web browser does not open, double-click `autorun.htm` on the CD to open the Online Documentation page.
4. Locate the Rational Test 7 section on the page.
5. Double-click the name of the document you want to open.

You can also copy the online manuals to your hard disk or to a network, and then open a manual by double-clicking its file name.

## Using Help

---

You can view Help whenever you need assistance. Complete online information is available to help you perform tasks, understand concepts, and work with the user interface.

### Using the Help Contents and Searching for Information

To display the Help Contents and locate information:



- ▶ Click the **Help Contents** button on the toolbar of any product or component, or click **Help** → **Contents and Index**. The Help Topics dialog box contains three tabs:

**Contents** – A table of contents showing the organization of the information in the Help system.

**Index** – Use to search for topics by index entry.

**Find** – Use to do a full text search on any words in the Help.

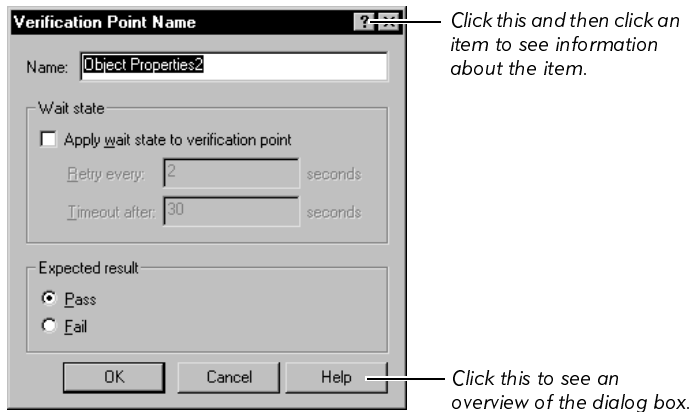
For information about performing standard Windows Help operations, see your Windows documentation.

## Help for Dialog Boxes, Menus, and Toolbars

Context-sensitive Help is available for dialog boxes, menu commands, and toolbars.

### Dialog Box Help

Most dialog box Help includes overviews and detailed item information.



### Menu Command Help



For menu command Help, highlight the command and press F1, or click the **Help Pointer** on the toolbar and select the command. A brief description of the command also appears in the status bar.

### Toolbar Button Help



For toolbar button Help, pause the pointer over the button. A yellow ToolTip appears below the button, and a brief description appears in the status bar. For more detailed information, click the **Help Pointer** on the toolbar and select the button.

## Typographical Conventions

---

This manual uses the following typographical conventions:

<b>Convention</b>	<b>Meaning</b>
<b>Bold Sans Serif Type</b>	Menu commands. Dialog box options. New terms. User-entered text.
<i>Italicized Type</i>	Variable text that you define. Emphasized text. Referenced sections and manual titles.
Monospace Type	Sample code.
SMALL CAPS	Keyboard key names.
+ between key names	Hold down the first key while pressing the next key. For example, CTRL+N means hold down the CTRL key while pressing N.
→ between menu commands	Choose each command in sequence. For example, <b>File</b> → <b>Save As</b> means choose <b>File</b> from the menu bar, and then choose <b>Save As</b> from the menu.

## ▶▶▶ CHAPTER 1

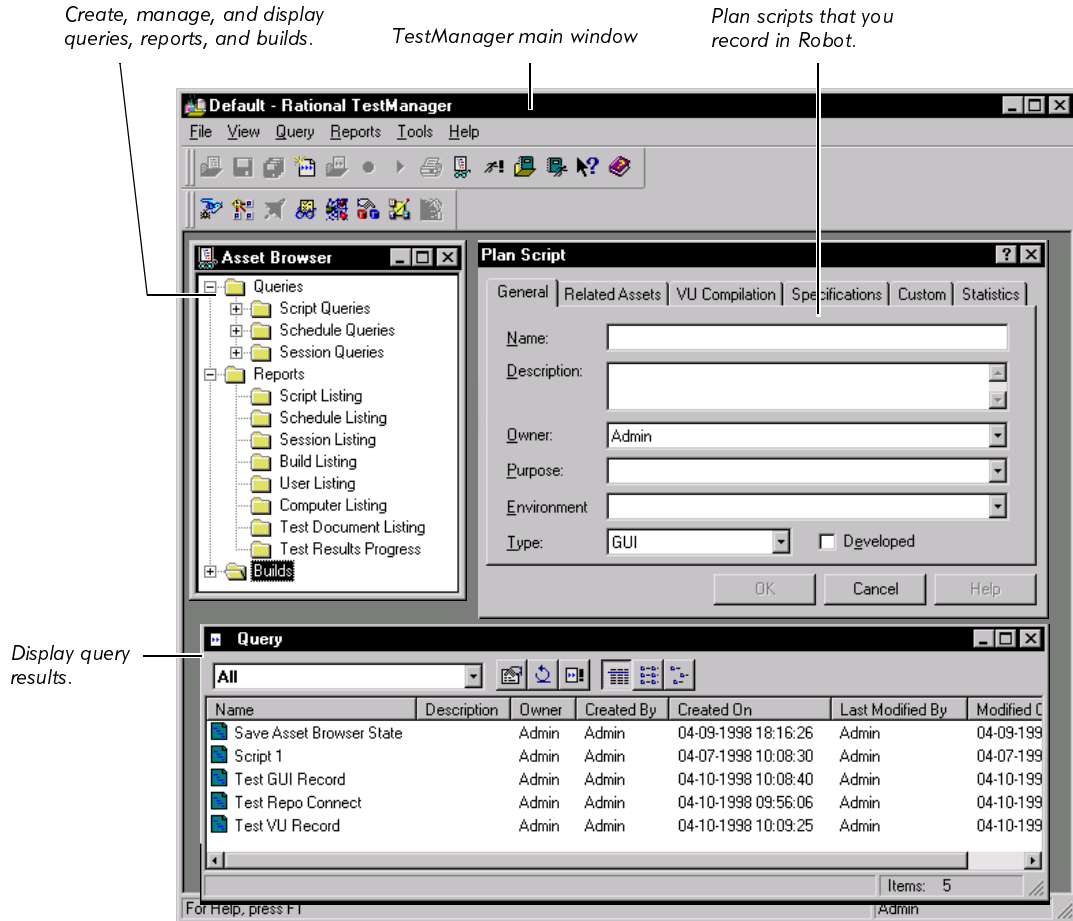
# Introduction to Rational TestManager

Rational TestManager lets you plan tests, manage your test assets, and run queries and reports.

You can use TestManager to:

- ▶ **Plan scripts.** A script is a file that is created when you record in Rational Robot.
- ▶ **Create, manage, and run queries.** The query tools in TestManager help you manage your scripts. You can use the default queries provided with TestManager or create queries of your own.
- ▶ **Create, manage, and run reports.** The reporting tools help you track assets such as scripts, builds, and test documents. They also help you track test coverage and progress.
- ▶ **Create and manage builds, log folders, and logs.** Logs are created when you play back a script in Rational Robot. Log folders are used to organize logs.

The following figure shows the main TestManager window and some of the other windows and dialog boxes that you can use to manage your tests.



## Starting Rational TestManager

Before you start using Rational TestManager, you need to have:

- ▶ Rational TeamTest installed. For information, see *Installing Rational ClearQuest/TeamTest Edition*.
- ▶ A Rational repository and a project within the repository. For information, see the *Using the Rational Administrator* manual.

## Logging In

When you log into TestManager, you provide your user ID and password, which are assigned by your administrator. You also specify the repository and project to log into.

To log in:

- ▶ Make sure you have created a repository. For information, see the *Using the Rational Administrator* manual.
- ▶ Click **Start** → **Programs** → **Rational ClearQuest/TeamTest Edition** → **Rational TestManager**.

Type your user ID and password.  
If you do not know these, see your administrator.

Select a repository. To change repositories later, exit all Rational components and log in again. (Repositories are created in the *Rational Administrator*.)

Select a project. You can change to another project within the same repository after you log in.

Click **OK** to log in.

## Changing to Another Project

After you are logged into TestManager, you can easily change to another project in the same repository.

To change to another project:

- ▶ Click **File** → **Change Project**.

Select a project in the repository.

Click **OK**.

To change to a project in a different repository, exit all Rational components and then log in again. Select the repository and project in the Rational Repository Login dialog box. (New repositories and projects are created in the Rational Administrator.)

## Opening Rational ClearQuest

You can open Rational ClearQuest from TestManager. But before you do, make sure that you have installed ClearQuest properly and set up the databases that you need. For information, see the *Getting Ahead with ClearQuest* manual.

To open ClearQuest from TestManager:

- ▶ Click **Tools** → **Rational ClearQuest**.



## ▶▶▶ CHAPTER 2

# Planning Your Tests

The first phase of a software testing effort involves test planning. In this phase, you write test plans, define test requirements, and plan test scripts that will validate the test requirements. Rational TestManager is the tool you use to plan your tests.

This chapter describes the planning phase of your testing effort. It includes the following topics:

- ▶ Working with test plans and other test documents
- ▶ Defining test requirements
- ▶ Planning scripts
- ▶ Managing requirements

## Working with Test Plans and Other Test Documents

---

Often, the first part of the test planning phase involves the creation of test plans and other documents. A **test plan** defines a testing project so it can be properly measured and controlled. The test plan usually describes the features and functions you are going to test and how you are going to test them. Often, the test plan discusses resource requirements and defines a project schedule. Large software projects may require several test plans, each one written by a separate quality engineer.

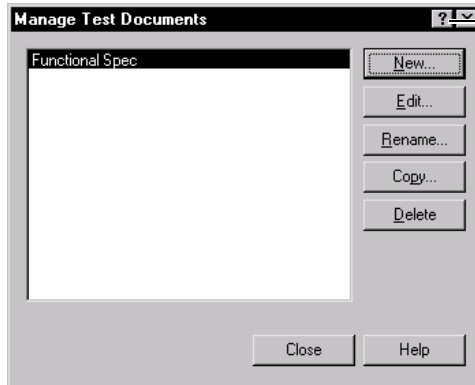
In TestManager, test plans, project schedules, and other related documents are called **test documents**. You develop your test documents using any word processing or scheduling program, and you create references to these documents in TestManager. The references are stored in the Rational repository. If the test documents themselves are modified during the project, you can use TestManager to rename, copy, or delete the references. You can also open the test documents directly within TestManager.

## Creating Test Document References

After you create test plans and other documents, you can use TestManager to create references to these documents. By doing this, you can use TestManager as your “control center” from which you can access any document that is important to the project.

To create a reference to a test document:

1. Click **Tools** → **Manage Test Documents**:



*For detailed information about an item, click the question mark, and then click the item.*

2. Click **New**.
3. In the Test Document Properties dialog box, type a name that refers to an existing test document. For clarity, it is best to use a name that is the same or very similar to the actual document name. The name can contain up to 40 characters.
4. Type a short description of the test document.
5. Type the path to the test document, or click **Browse** to search for the path.
6. Click **OK** to close the Test Document Properties dialog box.
7. Click **Close** to close the Manage Test Documents dialog box.

## Editing Test Document References

If test documents are modified or deleted during a project, you can use TestManager to rename, copy, or delete the reference. You can also edit the reference's name, description, or path.

To edit a reference to a test document:

1. Click **Tools** → **Manage Test Documents**.
2. Select a test document from the list.
3. Click one of the following buttons:

**Edit** – To change the description of the reference or to map the reference to a different source document.

**Rename** – To change the name of the reference.

**Delete** – To remove the reference.

**Copy** – To copy a reference to a test document (not the document itself).

4. Complete the edit, rename, deletion, or copy.
5. Click **Close**.

## Viewing Test Documents

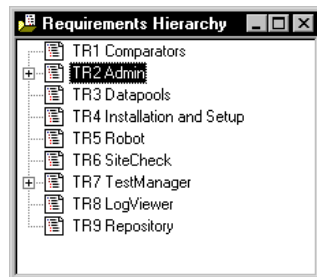
To open a test document using its associated editor:

1. Click **Tools** → **Manage Test Documents**.
2. Select a test document from the list.
3. Click **Edit**.
4. Click **Open** to open the document using its associated editor.

## Defining Test Requirements

---

After you write your test plans and other test documents, the next step in test planning is to define your **test requirements**—the features and functionality you plan to test. You define test requirements using the Requirements Hierarchy in TestManager. The Requirements Hierarchy is a graphical outline of requirements and nested child requirements. You can define a hierarchy of your project's requirements in any way that is appropriate for your project. The following figure shows a sample Requirements Hierarchy:



Requirements in TestManager are stored in a Rational RequisitePro database. RequisitePro is a requirements management tool that helps project teams control the development process by organizing, managing, and tracking the changing requirements of your system.

TestManager includes a baseline version of RequisitePro which you can use to build a test requirements hierarchy and then associate scripts with the requirements in the hierarchy.

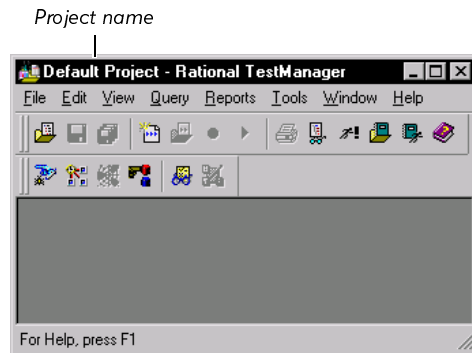
With the full version of RequisitePro, you can customize the requirements database and incorporate other advanced features such as traceability, change notification, and requirements attribute management. With the full version of RequisitePro, you can manage all of your project's requirements—user requirements, test requirements, software and product requirements, and any other requirements you may have.

TestManager lets you use either version of RequisitePro, the baseline version or the full version. The full version of RequisitePro is included as part of Rational Suite TestStudio, or you can purchase it separately.

## Working with Projects

Rational Test categorizes test information within a repository by project. You can use the Rational Administrator to create and manage projects.

The **project** represents the top-level name of a software testing effort. Its name is displayed in the TestManager title bar and in the title bar of the other components of Rational Test.



Projects help you categorize your software testing information and resources for easy tracking. Each project can include information about requirements, test documents, scripts, and verification points.

The number of projects in a repository depends on the complexity of the application-under-test or on the number of ongoing unrelated testing efforts. For example, you could divide a large application into several smaller projects, or you could define a separate project for each unrelated testing effort.

The Requirements Hierarchy in TestManager is project-specific—that is, it is associated with a particular project. When you log into TestManager, you select the repository and project you want to work with. The default is the last project that was used. For information about creating and deleting projects, see the *Using the Rational Administrator* manual.

### Importing Test Assets

TestManager includes an Import Test Assets wizard that you can use to copy or import test scripts and other test assets from one project to another. The destination project may reside within the same repository or in another repository.

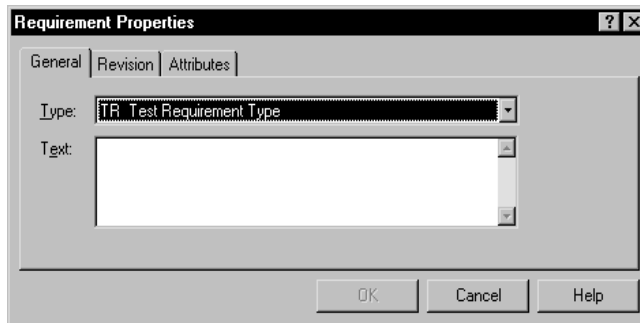
## Building the Requirements Hierarchy

The Requirements Hierarchy is displayed in its own window within TestManager. It displays requirements as well as the scripts that you can associate with those requirements. You build the Requirements Hierarchy as you would build an outline. Each requirement can have multiple levels of nested sub-requirements, called child requirements.

### Inserting a Requirement

To insert a requirement in the hierarchy:

1. Click **File** → **Open Requirements** to open the Requirements Hierarchy.
2. Click **Edit** → **Insert Requirement** to open the Requirement Properties dialog box.



3. Select a requirement type from the list.

Requirement types are defined in RequisitePro. With the base version of RequisitePro that is included with TestManager, there is only one requirement type.

**NOTE:** You cannot change the type of an existing requirement. In addition, a child requirement must be the same type as its parent.

4. Type the requirement name in the **Text** box.
5. Click the **Revision** tab.
6. In the **Description** box, type a description for this version of the requirement.

The screenshot shows the 'Requirement Properties' dialog box with the 'Revision' tab selected. The 'Last Saved' section contains the following fields:

- Revision: [ ] Label: [ ]
- Date: 12/12/99 Time: 0:00AM
- Author: [ ]
- Description: [ ]

Buttons include 'History...', 'OK', 'Cancel', and 'Help'.

For example, you might type *Update priority attribute value from medium to high*.

7. Type a label for the revision—for example, *beta 2*, *alpha 1*, or the build number.
8. Click the **Attributes** tab.

The screenshot shows the 'Requirement Properties' dialog box with the 'Attributes' tab selected. It displays a table of attributes and their values:

Attribute	Value
Priority	Medium
Status	Approved
Cost	
Difficulty	Medium
Stability	Medium
Assigned To	

Buttons include 'OK', 'Cancel', and 'Help'.

Attributes describe the requirements in the Requirements Hierarchy. In TestManager, you can assign values to attributes. To add or modify attributes and their values, you must use RequisitePro.

9. Click in the value column.
  - For a text field, type in a value.
  - For a single-value field, select a value from the list.
  - For a multi-value field, double-click the **Value** column to open the Select Attribute Values dialog box. Move the values from the **Available** list to the **Selected** list and click **OK**.
10. Click **OK**.

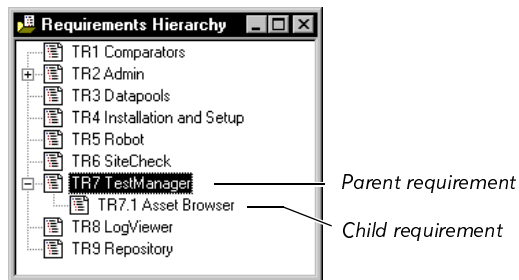
The new requirement appears in the hierarchy.

## Inserting a Child Requirement

To insert a child requirement:

1. Select a requirement in the hierarchy and click **Edit** → **Insert Child Requirement**, or right-click a requirement in the hierarchy and click **Insert Child Requirement**.
2. Fill out the Requirement Properties dialog box as described in *Inserting a Requirement* on page 2-6. Remember that a child requirement must be the same type as its parent.

When you finish, the child requirement appears beneath its parent in the hierarchy.



## Editing Requirement Properties

To change the name of a requirement or to change other properties:

1. Right-click the requirement and click **Properties** or double-click the requirement to open the Requirement Properties dialog box.
2. Make edits to the properties as needed.
3. Click **OK**.

## Expanding and Collapsing the Requirements Hierarchy

For large applications, the Requirements Hierarchy can become quite extensive. While creating or editing a hierarchy, you can simplify navigation through the tree structure by showing or hiding specific levels of the hierarchy. All parent requirements display a plus (+) or minus (-) sign on their left side

- ▶ To view the children, click the plus (+) sign.
- ▶ To hide the children, click the minus (-) sign.



## Planning Scripts

---

After you create your test documents and define your test requirements, the next step in the planning process is to create the scripts that you will use to validate the requirements. There are two ways to do this.

One way is to simply start recording in Robot. When you start to record a script, Robot displays a dialog box in which you supply a script name. After you supply the name and click **OK**, you can start recording.

The preferred usage model, however, is to use TestManager to plan the scripts you will need and to attach these scripts to test requirements. After you complete this process, you can start Robot, select one of the scripts you planned, and start recording.

By following this usage model, you will know early on the size of the testing effort, and you will be able to assign the development of each script to the appropriate individual. This model can also help you develop logical and consistent naming conventions for all of your scripts.

After you start to develop your scripts in Robot, you can always use TestManager to plan additional scripts.

To plan scripts, open the Plan Script dialog box and assign properties, such as a name, description, and owner. For details, see *Assigning a Name and Other Properties* on page 2-10.

Other tasks you can perform with scripts include:

- ▶ Attaching a script to a test requirement
- ▶ Referencing specification files, such as functional specifications or design documents
- ▶ Customizing script properties
- ▶ Displaying script statistics
- ▶ Controlling whether Robot starts after planning a script
- ▶ Deleting scripts
- ▶ Copying scripts

These tasks are described in more detail in the following sections.

## About Scripts

Much of your testing effort involves planning and recording scripts. A **script** has two parts:

- ▶ A file that can be executed by Robot.
- ▶ A set of script properties, such as the type and purpose of the script.

You generate a script file when you record your GUI actions or client/server requests with Robot. Robot translates your actions into SQABasic scripting language commands, and writes them to the script.

Typically, you define the script's properties when you plan the script with TestManager. You can also define script properties in Robot after you record the script.

By playing back scripts in Robot, you can perform functional tests of GUI objects. A **functional test** helps you determine whether a system functions as intended by verifying objects over successive builds of the application-under-test.

## Assigning a Name and Other Properties

All scripts must have a name and can have several other optional properties. Script properties include:

- ▶ Name, description, owner, purpose, and test environment
- ▶ Related assets such as test requirements
- ▶ Notes and specification files
- ▶ Custom keywords

To assign a name and other properties to a script:

1. If necessary, open the Requirements Hierarchy by clicking **File** → **Open Requirements**.

2. Right-click the requirement you want to attach the script to and click **Plan Script**, or click **File** → **Plan Script**.

3. Type a name for the script, using a logical naming convention. The name can contain up to 40 characters.
4. Type an explanation of the script in the **Description** box.
5. Select the script's developer from the **Owner** list. Owners are defined in Rational Administrator.
6. Select the purpose of the script from the **Purpose** list. For information about adding purposes to the list, see *Customizing the Purpose List* on page 2-19.
7. Select the environment (operating system) in which the script will be used from the **Environment** list. For information about adding environments to the list, see *Customizing the Environment List* on page 2-18.
8. Select the script type:
  - **GUI** scripts are used in Robot for functional testing. With Rational PerformanceStudio, GUI scripts can also be used to distribute your functional tests among different computers so that your tests can run faster.
  - **VU** (virtual user) scripts are used in LoadTest performance tests to measure server response times and to add load to the system. Virtual user scripts require Rational PerformanceStudio.

9. Click **OK** and continue with the following section, *Attaching Scripts to Test Requirements*.

**NOTE:** The **Developed** check box indicates whether or not a script has actually been developed. TestManager considers a script to be developed when it has been recorded or edited. When planning a new script, the check box should be cleared.

## Attaching Scripts to Test Requirements

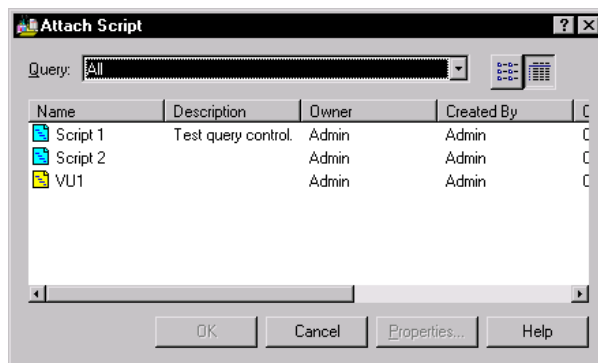
By attaching scripts to test requirements, you can more easily track which areas of the application-under-test have test coverage. When you plan a script from the Requirements Hierarchy, the script is automatically attached to a test requirement. However, if you use **File** → **Plan Script** to plan the script, you will need to explicitly attach the script to a test requirement, as described in the following sections.

**NOTE:** A script can be attached to only one requirement at a time.

### Attaching Scripts from the Requirements Hierarchy

To attach a script to a requirement from the Requirements Hierarchy:

1. Right-click a requirement in the hierarchy and click **Attach Script**, or select a requirement in the hierarchy and click **Edit** → **Attach Script**:

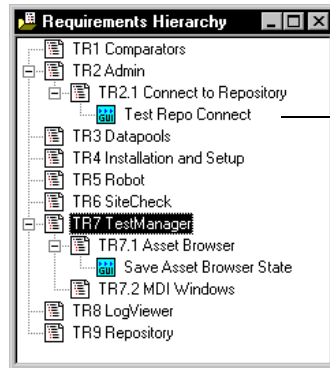


2. Select a query from the Query list.

The query lets you narrow down the list of scripts that are displayed. This is extremely useful in projects with hundreds of scripts. For example, you can query on all scripts, or only some scripts. For information about how to create a query, see *Creating New Queries* on page 4-3.

3. Select a script from the list and click **OK** to attach it to the requirement.

The script appears in the hierarchy beneath the requirement.

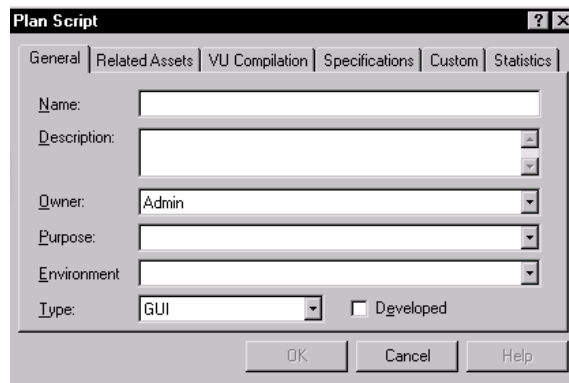


The **Test Repo Connect** script is now attached to the **Connect to Repository** requirement.

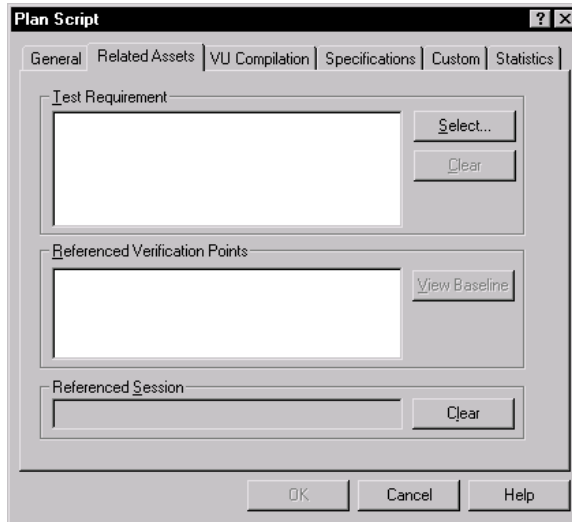
## Attaching Scripts from the Plan Script Dialog Box

To attach a script to a requirement from the Plan Script dialog box:

1. Click **File** → **Plan Script**.



2. Supply a name, description, owner, purpose, environment, and type for the script. For more information, see *Planning Scripts* on page 2-9.
3. Click the **Related Assets** tab.



**NOTE:** If a requirement is selected in the Requirements Hierarchy, the requirement will appear automatically in the Test Requirement box.

4. Click **Select** to open the Requirements Hierarchy.
5. Select the test requirement you want to attach the script to.
6. Click **OK** to close the Requirements Hierarchy.
7. Click **OK** to close the Plan Script dialog box.

**NOTE:** You can also use this procedure to attach a script to a test requirement from within Robot. Simply open the Script Properties dialog box and follow the previous procedure.

### Detaching a Script from a Requirement

When you detach a script from a requirement, you simply eliminate the mapping between them. Afterwards, you can attach the script to another requirement.

To detach a script from a requirement:

- ▶ Right-click the script in the Requirements Hierarchy and click **Detach**, or click **Edit** → **Detach**.

## Moving a Script from One Requirement to Another

To move a script to a different requirement, select the script in the Requirements Hierarchy and drag it to a different requirement, or use the **Attach Script** command to attach the script to the new requirement.

## Referencing Specification Files

You can set up your scripts so that each script references a specification file, such as a functional or design specification, or you can simply add a detailed note describing the script.

To reference a specification file:

1. Right-click in the Requirements Hierarchy and click **Plan Script**, or click **File** → **Plan Script**.
2. Supply a name, description, owner, purpose, environment, and type for the script. For more information, see *Planning Scripts* on page 2-9.
3. Click the **Specifications** tab.
4. Type any additional information about the script in the **Notes** box.
5. Type the path to the specification file, or click **Browse** to locate the file.
6. To open the specification file, click **Open**. Close the file when you have finished viewing it.
7. Click **OK** to close the Plan Script dialog box.

## Customizing Script Properties

When you use the Plan Script dialog box to define a script, you assign several properties, such as a name, description, owner, environment, purpose, and type. With TestManager, you can add your own properties and also customize the values that are used with many of the standard properties. You can define both the property itself (the label) and the values that can be used with that property. For example, you could add a new property called Test Type, and you could add Test Type values, such as UI, functional, and performance.

## Displaying Script Statistics

To display statistics about a script, such as the creation date, modification date, or the name of the developer:

1. Right-click a script in the Requirements Hierarchy and click **Properties**.
2. Click the **Statistics** tab.

## Controlling Whether Robot Starts After Planning a Script

You can control whether or not Robot starts automatically after you plan a new script in TestManager. You can plan all of your scripts before you start recording in Robot or you can plan one script at a time and record each one as you go.

To control whether Robot starts after you plan a new script:

1. Click **Tools** → **Options**.
2. Set the **Launch Robot** option:
  - Select the **Launch Robot after creating scripts** check box to start Robot automatically after you define a new script.
  - Clear the **Launch Robot after creating scripts** check box to prevent Robot from starting automatically after you define a new script.
3. Click **OK**.

## Deleting Scripts

There may be times when you want to delete scripts that are no longer useful.

To delete a script:

- ▶ Right-click a script in the Requirements Hierarchy and click **Delete**.

## Copying Scripts

You can use the Import Test Assets wizard to import scripts from another project in the same repository or from a different repository.

To copy or import a script:

1. Click **File** → **Import Test Assets**.
2. Select **Script** from the list, click one of the **Overwrite** options, and click **Next** to move to the next page.
3. Select the repository and project that contain the scripts you want to copy. Then, type a user ID and password for the project and repository and click **Next**.



4. Select the scripts you plan to import and click **Next**. If necessary, select a query from the list to display just the types of scripts you are interested in. For details about how to work with queries, see Chapter 4, *Querying the Rational Repository*.
5. Review the summary page that is displayed. If you want to change any settings, click **Back**. If you are satisfied with the settings, click **Finish**.

TestManager displays a status window while the scripts are being copied.

## Managing Requirements

---

As a software project evolves, some of your test requirements may need to be changed or deleted altogether. TestManager helps you maintain a requirement throughout the project's life cycle.

### Editing Requirement Properties

To edit requirement properties:

1. Click **File** → **Open Requirements** to open the Requirements Hierarchy.
2. Right-click a requirement in the hierarchy and click **Properties**.
3. If necessary, edit the requirement text.
4. Click the **Revision** tab. Update the description and label as needed. Click the **History** button to view the requirement's history.
5. Click the **Attributes** tab. If necessary, update the attribute values.
6. Click **OK**.

### Deleting Requirements

When you delete a requirement, all of its child requirements are also deleted. In addition, any references to the deleted test requirements in scripts are removed.

To delete a test requirement:

1. From the Requirements Hierarchy, right-click the requirement you want to delete.
2. Click **Delete**.
3. When TestManager asks if you want to delete the requirement, click **Yes**.

## Customizing Scripts

---

With TestManager, you can tailor much of the terminology associated with scripts to the standards and practices used within your organization. For example, you can:

- ▶ Customize the values that are displayed in the Script Properties and Plan Script dialog boxes, such as the list of operating environments and test types.
- ▶ Define custom fields to be used with scripts.
- ▶ Add custom labels to be used with custom fields.

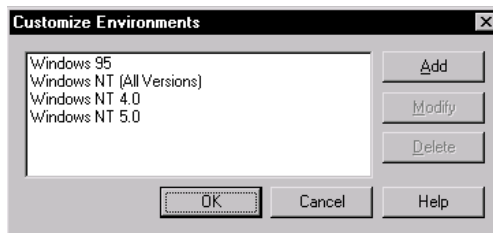
### Customizing the Environment List

One way to customize scripts is to edit the list of operating environments in which the script will run. You can add new environments to the list and modify or delete existing ones.

#### Adding New Environments

To add a value to the environment list:

1. Click **Tools** → **Customize** → **Environments**.



2. Click **Add**. Type an environment, and then click **OK** to add it to the list.
3. Click **OK**.

#### Modifying Existing Environments

To modify an existing environment:

1. Click **Tools** → **Customize** → **Environments**.
2. Select an environment from the list.
3. Click **Modify**. Type a new name for the environment and click **OK**.
4. Click **OK**.

## Deleting an Environment

To delete an existing environment:

1. Click **Tools** → **Customize** → **Environments**.
2. Select an environment from the list.
3. Click **Delete**.

## Customizing the Purpose List

Another way to customize scripts is to edit the purpose list. You assign a purpose to indicate why you would use a script. You can add new purposes to the list and modify or delete existing ones.

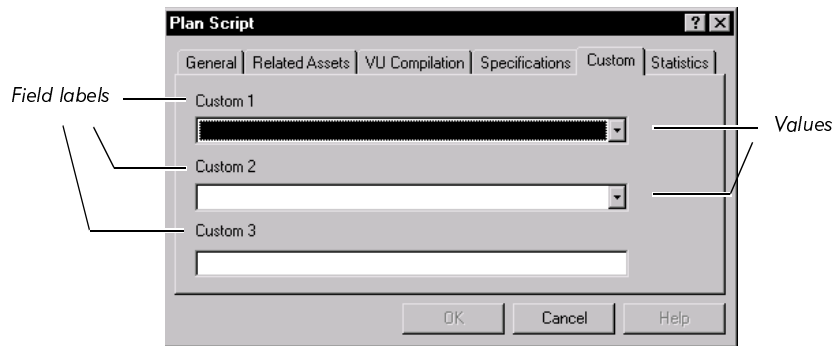
The procedure for adding, modifying, and deleting purposes is the same as the procedure for adding, modifying, and deleting environments. For details, see the previous section, *Customizing the Environment List*.

## Defining Custom Field Labels and Values

When you use the Plan Script dialog box to plan new scripts, you assign a name, description, owner, purpose, environment, and type.

With TestManager, you can add your own values to the purpose, environment, and type lists. You can also customize the field labels and list items that appear on the **Custom** tab of the Plan Script and Script Properties dialog boxes.

For example, you could change **Custom 1** to *Test Type* and add Test Type values, such as *UI*, *functional*, *performance*, and so on. You could change **Custom 2** to something that might indicate the complexity level or whether the script has any dependencies.

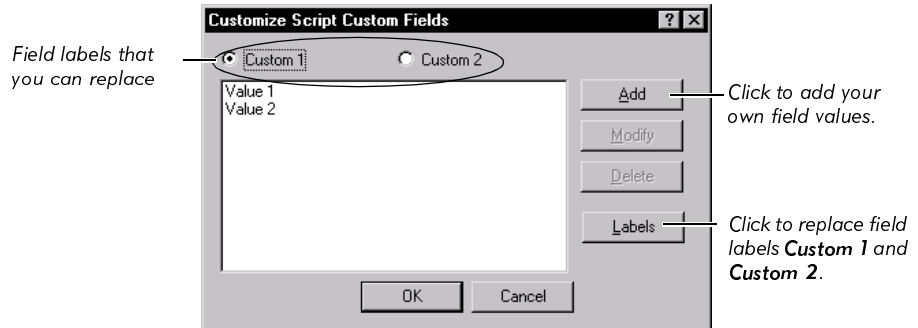


**NOTE:** The examples used in this section show how to customize field labels and values for your scripts.

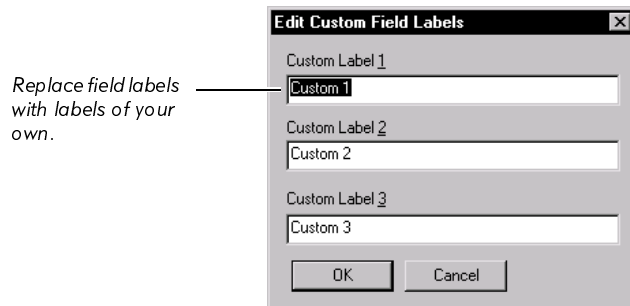
## Defining Custom Field Labels

To define custom field labels for your scripts:

1. Click **Tools** → **Customize** → **Script Custom Fields**.



2. Click **Labels**.



3. Replace **Custom Label 1**, **Custom Label 2**, or **Custom Label 3** with your own field labels and click **OK** to return to the Customize Script Custom Fields dialog box.
4. Click **OK**.

## Modifying Custom Field Values

To modify values for custom fields:

1. Click **Tools** → **Customize** → **Script Custom Fields**.
2. Click either **Custom 1** or **Custom 2**.
3. Click one of the default values, **Value 1** or **Value 2**.
4. Click **Modify**, type the new name for the value, and click **OK**.
5. Repeat steps 2 – 4 to modify other custom values.

## Adding Custom Field Values

To add a value to a custom field:

1. Click **Tools** → **Customize** → **Script Custom Fields**.
2. Click either **Custom 1** or **Custom 2**.
3. Click **Add**, type the name for the new value, and click **OK**.
4. Repeat steps 2 and 3.
5. Click **OK**.

## Planning Your Tests

## ▶▶▶ CHAPTER 3

# Managing Builds, Log Folders, and Logs

This chapter explains how to manage builds, log folders, and logs. It includes the following topics:

- ▶ Overview
- ▶ Displaying builds in the Asset Browser
- ▶ Creating a new build
- ▶ Copying, renaming, and deleting builds
- ▶ Renaming and deleting logs and log folders
- ▶ Displaying log properties
- ▶ Viewing logs
- ▶ Working with build states

## Overview

---

After you complete the test planning phase of your project, you can begin the test development and analysis phases. In the test planning phase, described in Chapter 1, *Introduction to Rational TestManager*, you create test documents, define test requirements, and plan test scripts.

In the test development phase, you record the scripts that you planned in the planning phase. In the analysis phase, you play back the scripts and analyze the results.

Before you start the test development and analysis phases, you should know how to use TestManager to organize both the software builds and the test logs that are generated during the test analysis phase.

## Using Builds in Functional Testing

As described in Chapter 1, *Introduction to Rational TestManager*, functional tests compare the way an application-under-test *actually* behaves in the current build against its *expected* behavior as recorded and validated in a previous build. Typically, as software developers add new features or fix defects, a new build is generated. Because this process happens on a continual basis during application development, hundreds of builds may be generated.

You use TestManager to define and manage builds. You can assign properties such as a name, description, owner, and build state. Afterwards, you can copy, rename, or delete a particular build as necessary.

## Generating Log Files

Your test team may develop hundreds of scripts to validate your test requirements. When you play back a script, Robot runs the script and generates a log file that reports the results. A **log** file contains the record of events that occurred during playback of a script. Each log file clearly specifies the pass/fail results of each verification point in the script, as shown in the following figure:

The screenshot shows the Rational LogViewer interface for a project named 'Default Project - Rational LogViewer - [OPtest4]'. The window has a menu bar (File, View, Defect, Tools, Window, Help) and a toolbar. Below the toolbar is a table with the following data:

Log Event	Result	Date	Time
- Script Start (OPtest4)	Fail	04/15/98	02:12:49 PM
Verification Point (Object Properties) - Object Properties	Fail	04/15/98	02:12:59 PM
Script End (OPtest4)		04/15/98	02:13:02 PM

The status bar at the bottom of the window displays 'For Help, press F1' on the left and 'Admin' on the right.

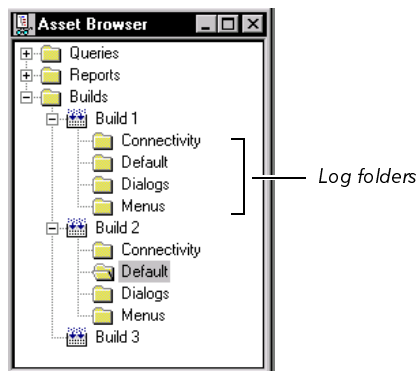
Because you normally play back scripts against each build of the application-under-test, you can easily generate thousands of log files. You can use these log files to generate reports that will help you determine the quality of each build and the progress of your testing effort. For example, one report, the Execution Coverage report, tells you the percentage and number of tests that have passed or failed for a specific build. For more information about these and other reports, see Chapter 5, *Running TestManager Reports*.



## Organizing Log Folders

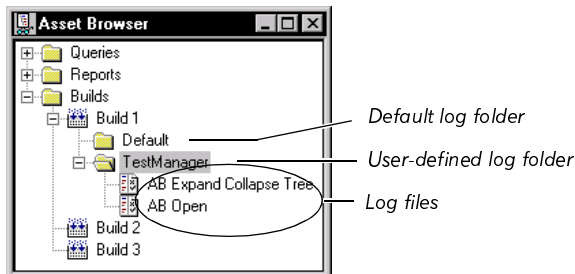
To more easily manage your log files, you can create log folders. Each build folder in the Asset Browser contains one default log folder, named **Default**. You can store all of your log files in the Default folder, or you can create your own log folders. For information about adding log folders, see *Creating a New Build* on page 3-5.

You can name and organize your log folders in any way that makes sense for your project. One logical way to organize your log folders is to mirror the top-level organization of your Requirements Hierarchy. If your Requirements Hierarchy is organized according to functional area, for example, you could create log folders for each of those functional areas.



## Displaying Builds in the Asset Browser

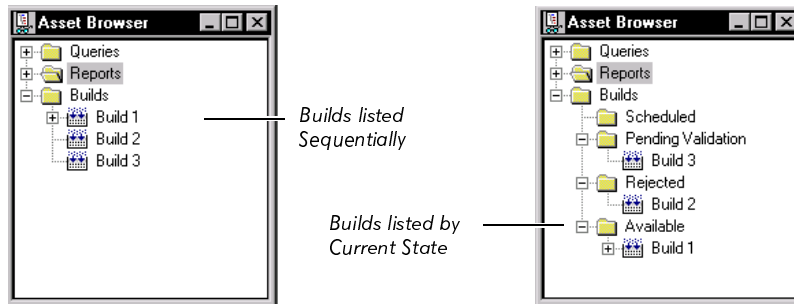
Builds are displayed as one of the top-level folders in the Asset Browser. The **Asset Browser** is the TestManager window that you can use to create, manage, and display builds, queries, and reports, as shown in the following figure:



You can organize the way builds are displayed in the Asset Browser in one of two ways:

- ▶ **Sequentially** – Organized in alphanumeric order.
- ▶ **By State** – Organized within the build state to which they are assigned.  
For information about build states, see *Working with Build States* on page 3-10.

The following figure displays the build portion of the Asset Browser and compares the two ways to organize the folder:



To toggle between the two views:

1. Right-click a build folder or a specific build in the Asset Browser.
2. Click **List Builds Sequentially** or **List Builds by State**.

To define the default view:

1. Click **Tools** → **Options**.
2. Click the **Miscellaneous** tab.
3. Click **List Builds Sequentially** or **List Builds by State**.

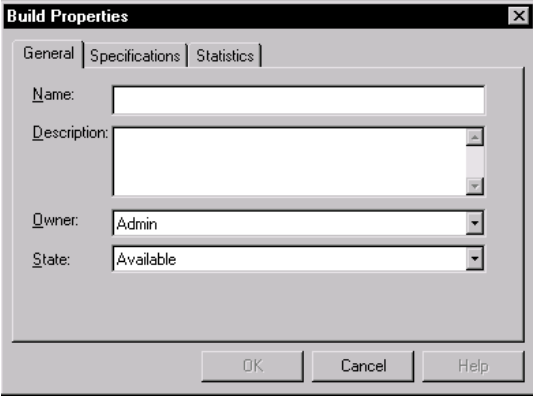
## Creating a New Build

---

With TestManager, you can create new builds and copy, rename, or delete existing ones. When you create a new build, you can add important information about the build in the form of notes. For example, you can include release note items or warnings to testers.

To create a new build:

1. Do one of the following:
  - Right-click the Builds folder or any build states folder in the Asset Browser and click **New Build**. (For information about build states, see *Working with Build States* on page 3-10.)
  - Click **Tools** → **Manage Builds**, and then click **New**.



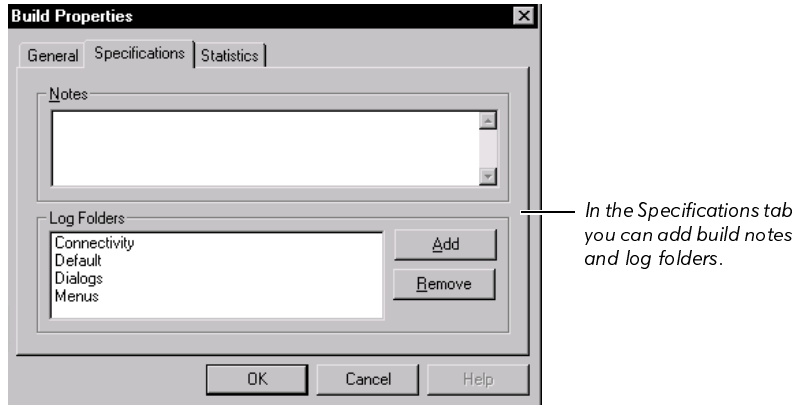
The screenshot shows a dialog box titled "Build Properties" with a close button (X) in the top right corner. It has three tabs: "General", "Specifications", and "Statistics". The "General" tab is selected. The dialog contains the following fields:

- Name:** A text input field.
- Description:** A text area with a scroll bar.
- Owner:** A dropdown menu with "Admin" selected.
- State:** A dropdown menu with "Available" selected.

At the bottom of the dialog are three buttons: "OK", "Cancel", and "Help".

2. Type a name for the build. The name can contain up to 40 characters.
3. Type a description of the build. The description can contain up to 255 characters.
4. Select the owner responsible for the build. Owners are defined in Rational Administrator. For details, see the *Using the Rational Administrator* manual.
5. Select the build state.

6. Click the **Specifications** tab.



7. Type any notes that are relevant to this particular build.
8. To add a log folder, click **Add**.
9. Type the name of the new log folder and click **OK** to accept the name.
10. Click **OK** again to close the Build Properties dialog box.

**NOTE:** There are two other ways to create log folders. You can right-click a build in the Asset Browser and click **New Log Folder**, or you can create a new log folder from the dialog box that appears when you play back a script in Robot.

## Copying, Renaming, and Deleting Builds

After you create a build, organize your log folders, and play back your scripts, you can copy the entire build folder in preparation for testing the next build.

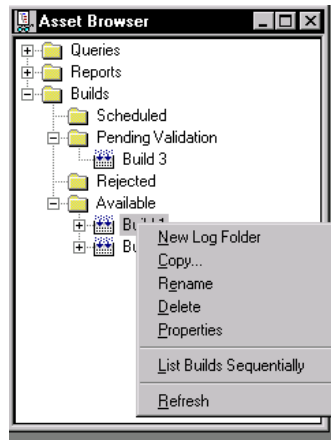
When you copy a build, you also copy the log folders that reside within the build folder. The logs themselves are not copied because they represent the test results for only one particular build.

In addition, you can rename or delete builds. For example, if a particular build has been approved as the beta release, you might want to rename it **Beta**. After the project is finished, you might want to archive your alpha, beta, and final builds and delete all of the other ones. When you delete a build, you delete everything contained within the build folder, including log folders and logs.

You can copy, rename, and delete builds from the Asset Browser or from the Manage Builds dialog box.

To copy, rename, or delete builds from the Asset Browser:

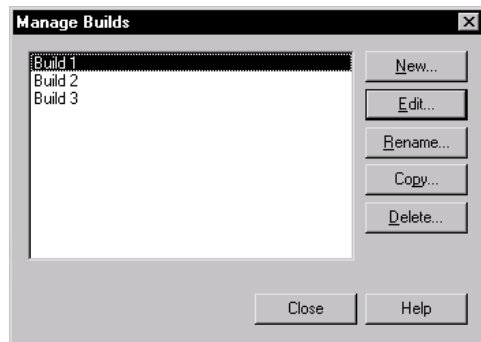
1. Right-click a build in the Asset Browser.



2. Do one of the following:
  - Click **Copy**, type a name for the new build, and click **OK**.
  - Click **Rename**, type the new name for the build, and press ENTER.
  - Click **Delete**, and then click **Yes** to delete the build.

To copy, rename, or delete builds from the Manage Builds dialog box:

1. Click **Tools** → **Manage Builds**.



2. Select a build from the list and do one of the following:
  - Click **Rename**, type the new name for the build, and click **OK**.
  - Click **Copy**, type a name for the new build, and click **OK**.
  - Click **Delete** and then click **Yes** to delete the build.
3. Click **Close**.

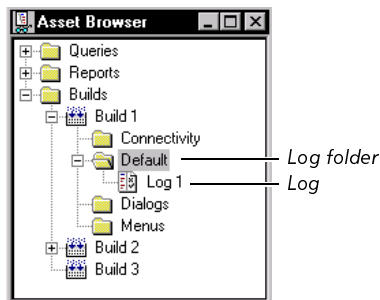
## Renaming and Deleting Logs and Log Folders

---

When you play back a script in Robot, you assign a name to the log file that is created. By default, a log has the same name as the script. Later, you can rename the log or log folder from the Asset Browser in TestManager. You can also delete logs and log folders from the Asset Browser.

To rename or delete logs and log folders:

1. Right-click a log or log folder in the Asset Browser.



2. Do one of the following:
  - Click **Rename**, type the new name for the log or log folder, and press ENTER.
  - Click **Delete**, and then click **Yes** to delete the log or log folder.

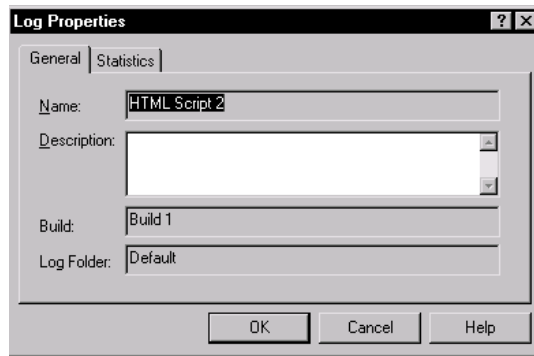
## Displaying Log Properties

---

Log properties include a name and description, as well as the build and log folder to which the log belongs.

To display log properties:

1. Right-click a log in the Asset Browser.
2. Click **Properties**.



3. Optionally, you can edit the log's Description.

## Viewing Logs

---

To view a log in the LogViewer:

- ▶ Double-click a log in the Asset Browser or right-click a log in the Asset Browser and click **Open**.

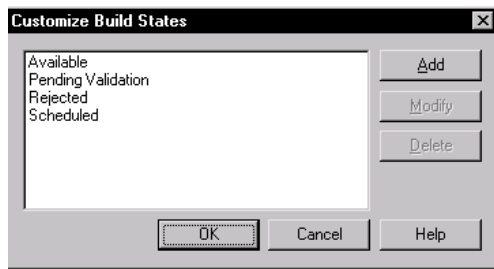
## Working with Build States

---

You can use TestManager to organize the builds in the Asset Browser sequentially or by build state. **Build states** provide a way to organize builds within a build hierarchy. When you organize by build state, you can use the default build states that are provided (Available, Pending Validation, Rejected, and Scheduled), or you can create your own. You can also modify or delete the existing build states.

To add a new build state:

1. Click **Tools** → **Customize** → **Build States**.



2. Click **Add**. Type a build state, and then click **OK** to add it to the list.
3. Click **OK**.

To modify or delete an existing build state:

1. Click **Tools** → **Customize** → **Build States**.
2. Select a build state from the list.
3. Do one of the following:
  - Click **Modify**, type a new name for the build state, and then click **OK**.
  - Click **Delete**.
4. Click **OK**.



## ▶▶▶ CHAPTER 4

# Querying the Rational Repository

This chapter explains how to use queries to manage the scripts in your projects. It includes the following topics:

- ▶ Overview
- ▶ Running queries
- ▶ Creating new queries
- ▶ Editing existing queries
- ▶ Setting query options
- ▶ Configuring the Query window

## Overview

---

A **query** is a request for information stored in the repository.

Rational Test provides a powerful set of query tools that help you manage the vast amount of information accumulated during the software testing effort. All of this information is stored in the repository and is accessed with the query tools in TestManager and Robot. You can use the default queries provided with TestManager, or you can create queries of your own.

For information about querying the Rational ClearQuest defect database, see the ClearQuest Help.

## Running Queries

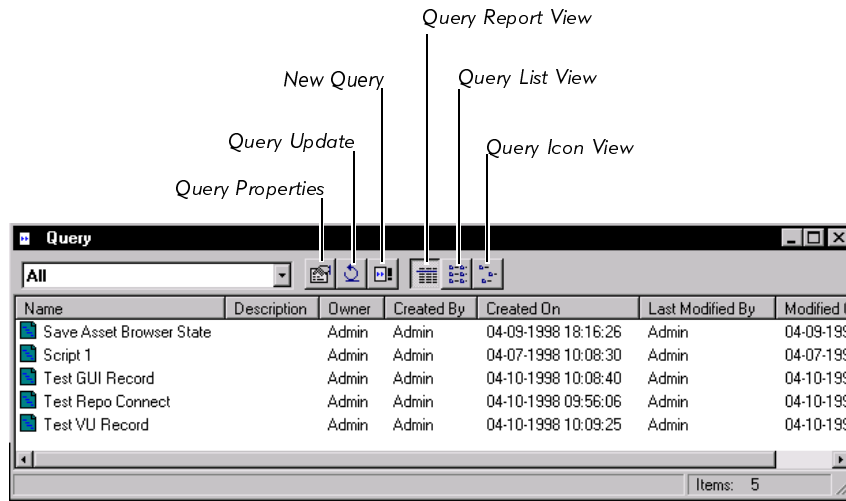
TestManager includes several default queries that you can use. There are queries for retrieving all or just some scripts from the repository.

To run a query, do one of the following:

- ▶ Right-click a script query in the Asset Browser, and then click **Run**.
- ▶ Click **Query** → **Run**, select a query type from the list, select a query, and then click **OK**.

## The Query Window

Query results are displayed in the Query window, as shown in the following figure. This window contains a toolbar for selecting commands and a grid for displaying the results of your query.



You can use the toolbar buttons to display a query’s properties, to retrieve updated data from the repository, to create a new query, and to change how the query results are displayed.

For more information about the toolbar buttons, see the TestManager Help. For information about refreshing the Query window with the most up-to-date information from the repository, see *Setting Query Options* on page 4-9.

You can access the Query window (minus some of the toolbar buttons) from other parts of Rational Test, such as the Robot Record and Playback dialog boxes.

**NOTE:** If no items are listed in the Query window, click the **Query Update** button on the toolbar. To make items appear automatically, click **Automatic** in the Query View tab of the TestManager Options dialog box. For more information about query options, see *Setting Query Options* on page 4-9.

## Deleting Scripts

During your testing effort, you may find scripts that you no longer need. You can delete these scripts from the Query window.

To delete a script:

1. Run a query, as described in *Running Queries* on page 4-2.
2. From the Query window, right-click the item you want to delete and click **Delete Script**.
3. When TestManager asks if you want to delete the item, click **Yes**.

You can also delete scripts from the Requirements Hierarchy in TestManager.

## Creating New Queries

The major tasks involved in creating a new query are:

1. Open the Query Properties dialog box and type a query name.
2. Choose the fields or columns that you want the query to display.
3. Click the **Sort** tab and specify the sort order.
4. Click the **Filters** tab and define one or more filters to narrow down the amount of data to display.

These tasks are described in more detail in the following sections.

## Opening the Query Properties Dialog Box

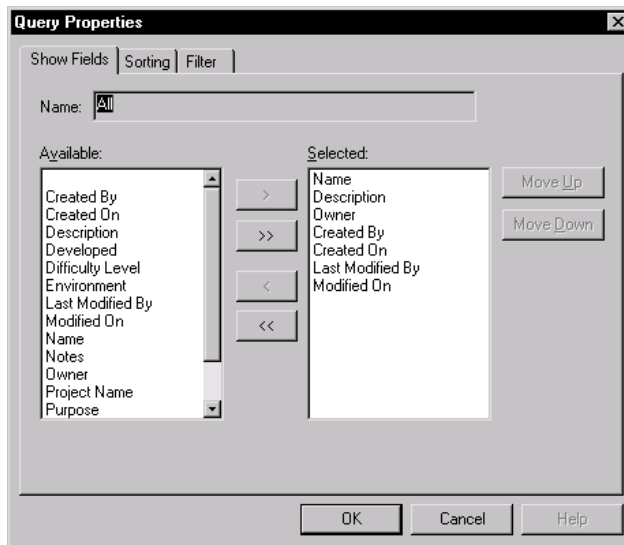
You can open the Query Properties dialog box using menu commands or the Asset Browser.

To open the Query Properties dialog box using menu commands:

- ▶ Click **Query** → **New** → **Script Query**.

To open the Query Properties dialog box using the Asset Browser:

1. Click **View** → **Asset Browser** to display the Asset Browser.
2. Right-click the **Script Queries** folder, and then click **New Query**.



## Choosing Fields to Display

Each query that you create contains several fields or attributes that are predefined in the repository. When you create a query, you can choose to display some or all of these fields. Each field appears as a column in the Query window. Some of the query fields include the query name, description, purpose, owner, environment, and type.

To choose the fields to display:

1. Open the Query Properties dialog box. For details, see *Opening the Query Properties Dialog Box* on page 4-4.
2. Select one or more files in the **Available** list, and then click > or >>.
3. In the **Selected** list, click **Move Up** or **Move Down** to set the display order.
4. Click **OK** to finish defining the query, or click the **Sorting** tab or the **Filters** tab to add additional query properties.

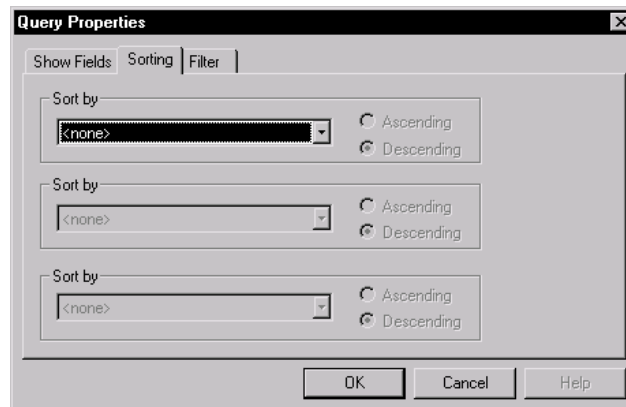
## Specifying the Sort Order

By specifying a sort order, you can control the order in which query items are displayed. TestManager lets you define three sort levels.

You sort the query by the field specified in the first sort box, then by the field specified in the second and third sort boxes. You can sort in either ascending or descending order.

To sort the items in query:

1. Open the Query Properties dialog box. For details, see *Opening the Query Properties Dialog Box* on page 4-4.
2. Click the **Sorting** tab.



3. Select a field from the **Sort by list** and click either **Ascending** or **Descending**.
4. To add a second-level sort, select a field from the second **Sort by list** and click either **Ascending** or **Descending**.
5. To add a third-level sort, select a field from the third **Sort by list** and click either **Ascending** or **Descending**.
6. Click the **Filters** tab to add additional query properties, or click **OK** to finish defining the query.

You can also sort by clicking any column header of the Query window.

## Adding a Filter Statement

A **filter** is the part of the query that specifies the information to be retrieved from the repository. Filter statements help you make queries more specific by narrowing down the information you are searching for. You can build simple filter statements or combine simple filter statements into more complex ones. For detailed information about the options that are available in filter statements, see the TestManager Help.

A Filter consists of two parts:

- ▶ Build filter statement
- ▶ Select where list

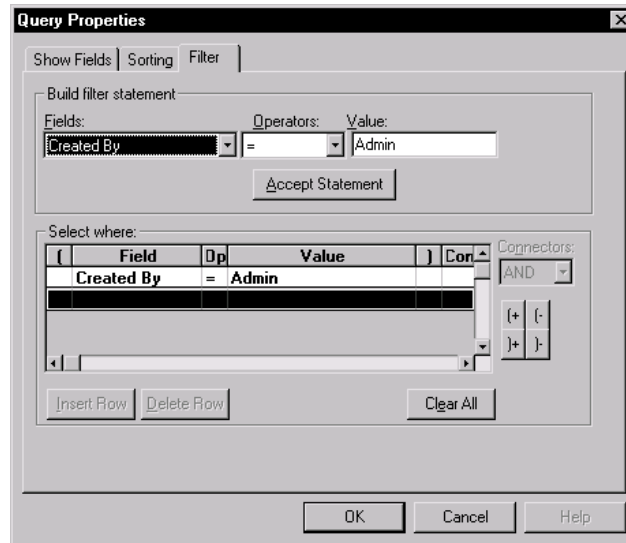
You use the **Build filter statement** to specify the filtering criteria for the query. A sample filter statement for a script could be something like *Created By = Mark*, which means, retrieve all of the scripts created by Mark. You build the filter statement by selecting a field and operator and typing a value. After you define the filter statement, you add it to the **Select where** list.

Each line in the **Select where** list shows one filter statement. You can combine filter statements into complex filters by using the **Connectors** options and the **Parentheses** buttons.

## Adding a Simple Filter Statement

To add a simple filter statement:

1. Open the Query Properties dialog box. For details, see *Opening the Query Properties Dialog Box* on page 4-4.
2. Click the **Filter** tab.



3. Select a field from the **Fields** list. If you select one of the Date fields—such as Created on or Modified on—TestManager inserts Date into the Value field and displays another field into which you can type or select the date.
4. Select an operator from the **Operators** list. The list of available operators varies, depending on the field you choose. For information about which Operators are available, see the TestManager Help.
5. Specify a filter **Value**. For alphanumeric fields, you simply type the value. For Date fields, you can either type in a date, use the spin buttons, or double-click the Date field to display a calendar.
6. Click **Accept Statement** to add the filter statement to the **Select where** list.
7. Repeat steps 3 – 6 to build additional filter statements (if necessary).
8. Click **OK**.

## Building Complex Filter Statements

You can create complex filters based on multiple fields and data values in the repository. You do this by building and accepting several filter statements and using connectors to link the individual statements. The two available connectors are **AND** and **OR**. To combine two or more statements, use parentheses.

To create a complex filter:

1. Build at least two filter statements using the procedure in *Adding a Simple Filter Statement* on page 4-7.
2. Highlight the first filter statement in the **Select where** list, and then click the button next to the **Connectors** list box.

The **Connectors** list box shows the options for linking the filter statements.

3. Select either **AND** or **OR** from the **Connectors** list, or type your choice. The text appears on the line with the filter statement.
4. You can use the parentheses buttons to combine more than two statements.

The following is an example of a complex filtering statement that uses connectors and parentheses:

```
Owner = Eric AND  
(Environment = Windows NT 4.0 OR  
Environment = Windows NT 5.0)
```

This statement retrieves all scripts that are owned by Eric and expected to run on either Windows NT 4.0 or Windows NT 5.0.



## Editing Existing Queries

---

With TestManager, you can edit the properties of a query. You can also copy, rename, and delete queries.

To edit, copy, rename, or delete a query:

1. Click **Tools** → **Manage Queries**.
2. Type or select a query type.
3. Select a query from the list.
4. Click **Edit** to edit the query properties or click **Rename**, **Copy**, or **Delete** to perform one of these actions.
5. Click **OK** twice.

## Viewing Query Properties

In addition to the previous procedure, you can also view a query's properties by doing one of the following:

- ▶ Right-click a query in the Asset Browser, and then click **Properties**.
- ▶ Run a query, and then click the Query properties button.

## Setting Query Options

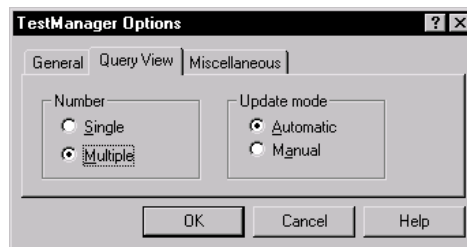
---

TestManager provides two query options that you can set:

- ▶ Number
- ▶ Update Mode

To set query options:

1. Click **Tools** → **Options**.
2. Click the **Query View** tab.



3. In the **Number** area:
  - Click **Single** to indicate that you want only one Query window for each query type to be open at a time. Running successive queries for a particular type replaces the data in the original window.
  - Click **Multiple** to allow more than one open Query window for each query type.
4. In the **Update mode** area:
  - Click **Manual** to update the Query window only when you click the **Query Update** button.
  - Click **Automatic** to update the Query window automatically each time you run a query or change the way you want the query displayed.

**NOTE:** If your repository is relatively small, **Automatic** ensures that the Query window always contains the latest data. However, if your repository contains a large amount of data or if you are in a networked environment, automatic updating can affect the performance of your queries.

## Configuring the Query Window

---

After you choose the fields you want to display in the Query window, you can adjust the column widths. Place the pointer over the column separator line in the title to display the column adjustment cursor, and then click and drag the mouse to set the width.

At any time, you can also:

- ▶ Replace one column with another.
- ▶ Insert a column.
- ▶ Delete a column.

### Replacing a Column

To replace one column with another:

1. Run a query.
2. Right-click the column header that you want to replace.
3. Click **Set Field** and select a replacement field from the list.

## Inserting a Column

To insert a column in a query:

1. Run a query.
2. Right-click on a column header.
3. Click **Insert Field** and select a field from the list.

## Deleting a Column

To delete a column:

1. Run a query.
2. Right-click the column header that you want to replace.
3. Click **Delete Field**.



## ▶▶▶ CHAPTER 5

# Running TestManager Reports

This chapter explains how you can create and run TestManager reports to more easily manage your testing effort. It includes the following topics:

- ▶ Types of reports
- ▶ Selecting reports to use
- ▶ Working with listing reports
- ▶ Working with coverage reports
- ▶ Working with the Test Results Progress report
- ▶ Copying, renaming, and deleting reports

## Types of Reports

---

TestManager provides three types of reports to help you in your testing efforts:

- ▶ Listing reports
- ▶ Coverage reports
- ▶ Progress reports

In addition, there are reports available in Rational ClearQuest. These reports, as well as report formats, queries, and charts, are available to help you manage your defect database. These reports and other items are created for you automatically when you create a repository that contains an associated ClearQuest database.

For further information about using these defect reports see the ClearQuest Help. For information about creating a repository, see the *Using the Rational Administrator* manual.

For more information about using these defect reports, see the ClearQuest Help. For information about creating a repository, see the *Using the Rational Administrator* manual.

## Listing Reports

---

Listing reports display lists of the different items stored in a project. TestManager includes listing reports for scripts, builds, users, computers, and test documents. For example, you can could a Script Listing report that:

- ▶ Lists all of the properties of all the scripts in your project.
- ▶ Summarizes all of the scripts in your project.
- ▶ Lists or summarizes GUI scripts.

## Coverage Reports

Coverage reports help you track the progress of your test planning, test development, and test execution efforts by showing you whether or not you are validating your test requirements. TestManager includes three types of coverage reports:

**Planning Coverage** reports facilitate test planning by showing you the percentage and number of test requirements that have been assigned scripts. You can use these scripts to validate your test requirements. Planning Coverage reports can be especially useful during the test planning phase of a project. For information about the planning phase, see Chapter 2, *Planning Your Tests*.

**Development Coverage** reports facilitate the test development phase of a project by showing you what percentage and number of test requirements have scripts that are ready to be played back. These reports show you not only whether you have planned a script for each requirement, but also whether you have actually recorded the script. TestManager uses the **Developed** check box in the Plan Script and Script Properties dialog boxes to determine whether or not a script has been developed. TestManager considers a script to be developed when it has been recorded or edited.

**Execution Coverage** reports provide crucial information about the quality of a specific build and the progress of your ability to test that build. These reports tell you the percentage and number of tests that have passed or failed for a specific build. Execution Coverage reports show you not only whether you have planned and developed a script for each requirement, but also whether you have actually played back the script and obtained test results.

## Progress Reports

TestManager includes one progress report—the Test Results Progress report. This report lets you track the success rate of your scripts over multiple builds of an application-under-test.

## Selecting Reports to Use

---

This section provides some basic guidelines for deciding which report to use to find the information you want from the Rational repository.

<b>When you want to retrieve</b>	<b>Use this report type</b>
A list of some or all of the scripts in your project.	Script Listing
A list of the builds in your project.	Build Listing
A list of the users in your project. (Users are defined in Rational Administrator.)	User Listing
A list of the computers in your project. (Computers are defined in Rational Administrator.)	Computer Listing
A list of the test documents in your project.	Test Document Listing
The percentage of requirements that include planned scripts.	Planning Coverage
The percentage of requirements that have scripts ready for testing.	Development Coverage
The percentage of scripts that have passed or failed for a particular build.	Execution Coverage
The success rate of a set of scripts over a particular set of builds.	Test Results Progress

## Working with Listing Reports

---

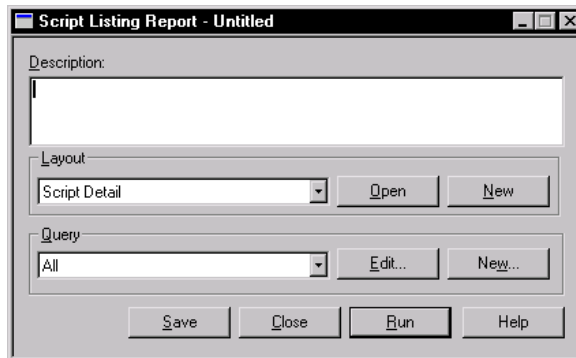
This section includes the following topics:

- ▶ Creating listing reports
- ▶ Running listing reports
- ▶ Opening listing reports

### Creating Listing Reports

To create a listing report, in this case, a Script Listing report:

1. Click **Reports** → **New** → **Script Listing**, or right-click one of the listing report folders in the Asset Browser and click **New Report**.



2. Type a description for the report.
3. Select a report layout from the list. Default layouts include:

**Detail** – Lists all properties. For example, if you are creating a Script Listing report, a detailed layout displays the script name, description, owner, environment, purpose, and so on.

**Summary** – Lists just the name, type, and description.

**NOTE:** You can also create your own report layout with the Report Layout Editor. Click **Open** to open the selected layout and edit it as needed. Click **New** to open a blank page that you can use to design a layout from scratch. For more information about the Report Layout Editor, see the TestManager Help.

4. If you are creating a Script report, select a query from the list.



Use the query to narrow down the number of items to be displayed in the report. Optionally, click **Edit** to modify the selected query, or click **New** to create a new query. For more information about queries, see *Creating New Queries* on page 4-3.

5. Click **Save**, type a name for the report, and click **OK**. The name can contain up to 40 characters.
6. Click **Close**.

## Running Listing Reports

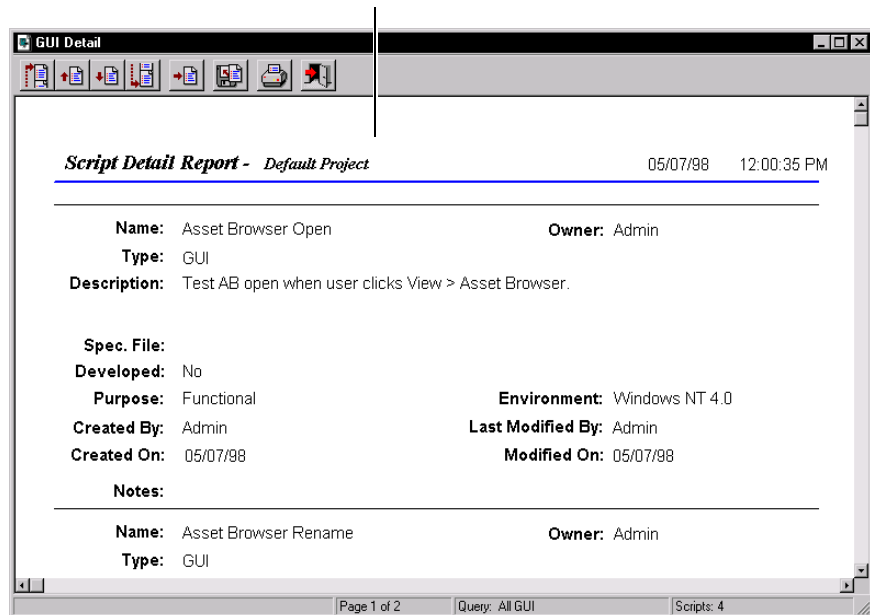
After you create a listing report, you can run it to see the results.

To run a listing report:

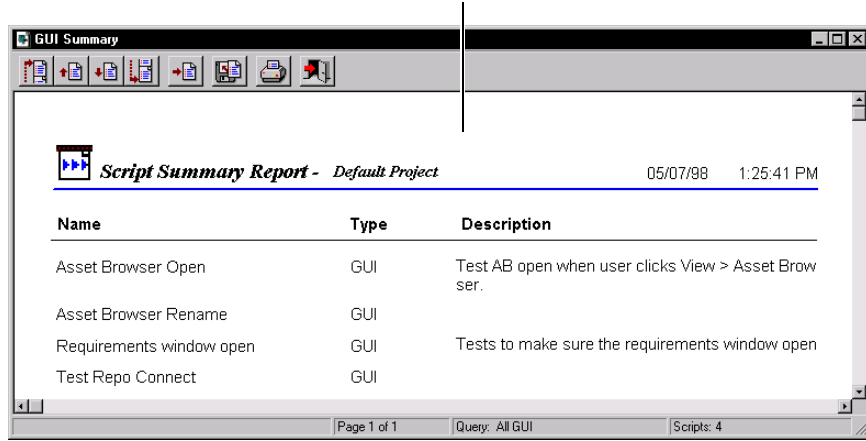
- ▶ Click **Reports** → **Run**, select a report from the list, and click **OK**, or right-click a listing report in the Asset Browser and click **Run**.

The following figures show a Script Detail Listing report and a Script Summary Listing report:

*Script Detail Listing Report*



### Script Summary Listing Report



## Opening Listing Reports

At any point after creating a report, you can open it and, if necessary, make changes to the report definition.

To open a listing report:

- ▶ Click **Reports** → **Open**, select a report from the list, and click **OK**, or right-click a listing report in the Asset Browser and click **Open**.

## Working with Coverage Reports

---

This section includes the following topics:

- ▶ Creating coverage reports
- ▶ Running planning coverage and development coverage reports
- ▶ Running execution coverage reports
- ▶ Opening coverage reports

## Creating Coverage Reports

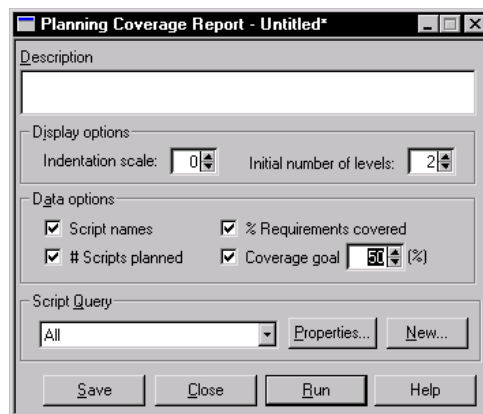
With TestManager, you can create three types of coverage reports. Planning Coverage reports show how many requirements you have planned scripts for. Development Coverage reports show how many requirements you have actually developed scripts for. These are scripts you have recorded with Rational Robot. Execution Coverage reports show how many requirements you have actually developed and run scripts for. These are scripts you have recorded and played back using Robot.

This section includes a procedure for creating each type of coverage report.

### Creating Planning Coverage Reports

To create a Planning Coverage report:

1. Click **Reports** → **New** → **Planning Coverage**, or right-click the **Planning Coverage** folder in the Asset Browser and click **New Report**.



2. Type a description for the report.
3. Set the **Display options**:

**Indentation scale** – Lets you specify the number of spaces to indent each sub-level included in the report. Specify a number from 0 to 10.

**Initial number of levels** – Lets you specify the initial number of levels in the Requirements Hierarchy to be displayed in the report. Later, you can collapse or expand the hierarchy when you run the report.

4. Set the following **Data options**:

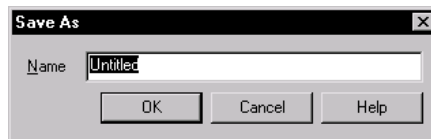
**Script names** – The report includes a column showing the names of the scripts planned for the items in the Requirements Hierarchy.

**# Scripts planned** – The report includes a column showing the number of scripts planned for the items in the Requirements Hierarchy.

**% Requirements covered** – The report includes a column showing the percentage of planned scripts for hierarchy items that have been recorded using Robot.

**Coverage goal** – The report includes a column showing if the number of recorded scripts meets the specified coverage goal.

5. Select a script query from the list. Use the query to narrow down the number of items to be displayed in the report. Optionally, click **Edit** to modify the selected query, or click **New** to create a new query. For more information about queries, see *Creating New Queries* on page 4-3.
6. Click **Save**, type a name for the report, and click **OK**.



7. Click **Close** to close the Planning Coverage Report specification window.

## Creating Development Coverage Reports

To create a Development Coverage report:

1. Click **Reports** → **New** → **Development Coverage**, or right-click the Development Coverage folder in the Asset Browser and click **New Report**.

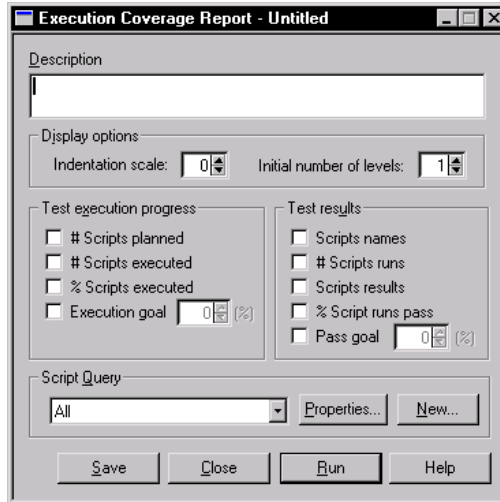


2. Type a description for the report.
3. Set the **Display options**. The display options are the same for all of the coverage reports. For details, see *Creating Planning Coverage Reports* on page 5-7.
4. Set the **Data options**:
  - Script names** – Includes a column in the report showing the names of the scripts planned for the items in the Requirements Hierarchy.
  - # Scripts planned** – Includes a column in the report showing the number of scripts planned for the items in the Requirements Hierarchy.
  - # Scripts developed** – Includes a column in the report showing the number of scripts recorded using Robot.
  - % Scripts developed** – Includes a column in the report showing the percentage of planned scripts that have been recorded using Robot.
  - Developed goal** – Includes a column in the report showing if the number of recorded scripts meets the specified coverage goal.
5. Select a script query from the list. Use the query to narrow down the number of items to be displayed in the report. Optionally, click **Edit** to modify the selected query, or click **New** to create a new query. For more information about queries, see *Creating New Queries* on page 4-3.
6. Click **Save**, type a name for the report, and click **OK**.
7. Click **Close** to close the Development Coverage Report specification window.

## Creating Execution Coverage Reports

To create an Execution Coverage report:

1. Click **Reports** → **New** → **Execution Coverage** or right-click the Execution Coverage folder in the Asset Browser and click **New Report**. This displays the Execution Coverage Report specification window.



2. Type a description for the report.
3. Set the **Display options**. The display options are the same for all of the coverage reports. For details, see *Creating Planning Coverage Reports* on page 5-7.
4. Set the **Test execution progress** options:
  - # Scripts planned** – Includes a column showing the number of scripts planned for the items in the Requirements Hierarchy.
  - # Scripts executed** – Includes a column showing if the scripts planned for each requirement have been played back in Robot.
  - % Scripts executed** – Includes a column showing the percentage of planned scripts for each requirement that have been played back in Robot.
  - Execution goal** – Includes a column showing if the number of executed scripts meets the specified execution coverage goal. You can set the goal by typing a number or by using the spin buttons.

5. Set the **Test results** options:
  - Script names** – Includes a column showing the names of the scripts planned for the items in the Requirements Hierarchy.
  - # Script runs** – Includes a column showing the number of times scripts have been played back for the items in the Requirements Hierarchy.
  - Script results** – Includes additional columns showing the number of scripts that have passed and failed for the items in the Requirements Hierarchy.
  - % Script runs passed** – Includes a column showing the percentage of scripts that have passed for the items in the Requirements Hierarchy.
  - Pass goal** – Includes a column showing whether the number of passed scripts meets the specified coverage goal.
6. Select a script query from the list. Use the query to narrow down the number of items to be displayed in the report. Optionally, click **Edit** to modify the selected query, or click **New** to create a new query. For more information about queries, see *Creating New Queries* on page 4-3.
7. Click **Save**, type a name for the report, and click **OK**.
8. Click **Close** to close the Execution Coverage Report specification window.

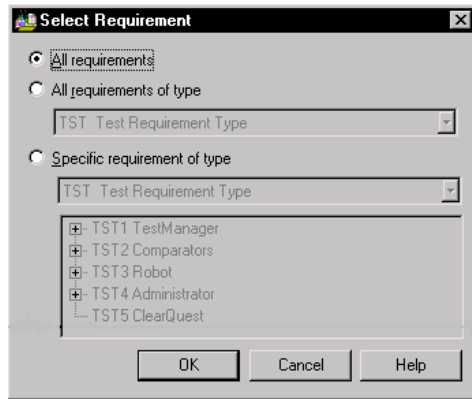
## Running Planning/Development Coverage Reports

After you create a Planning or Development Coverage report, you can run it to see the results.

To run a Planning Coverage or Development Coverage report:

1. Click **Reports** → **Run**, select a Planning Coverage or Development Coverage report from the list, and click **OK**, or right-click a Planning Coverage or Development Coverage report in the Asset Browser and click **Run**.

## Running TestManager Reports



2. Specify the requirements that you want to include.

**All requirements** – Includes all requirements, regardless of type, in the Requirements Hierarchy. With the full version of RequisitePro, you can define multiple requirement types. With the baseline version of RequisitePro that is included with TestManager, there is only one requirement type.

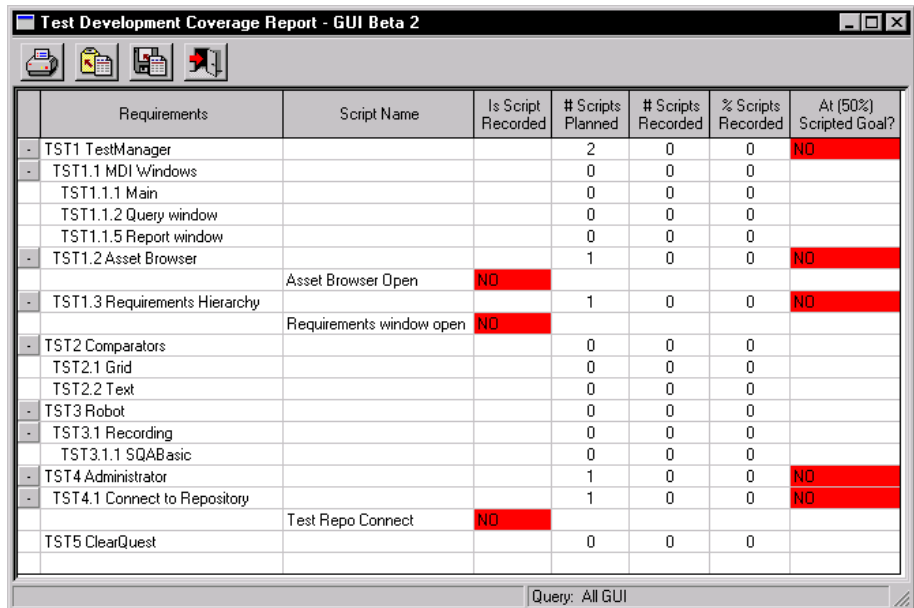
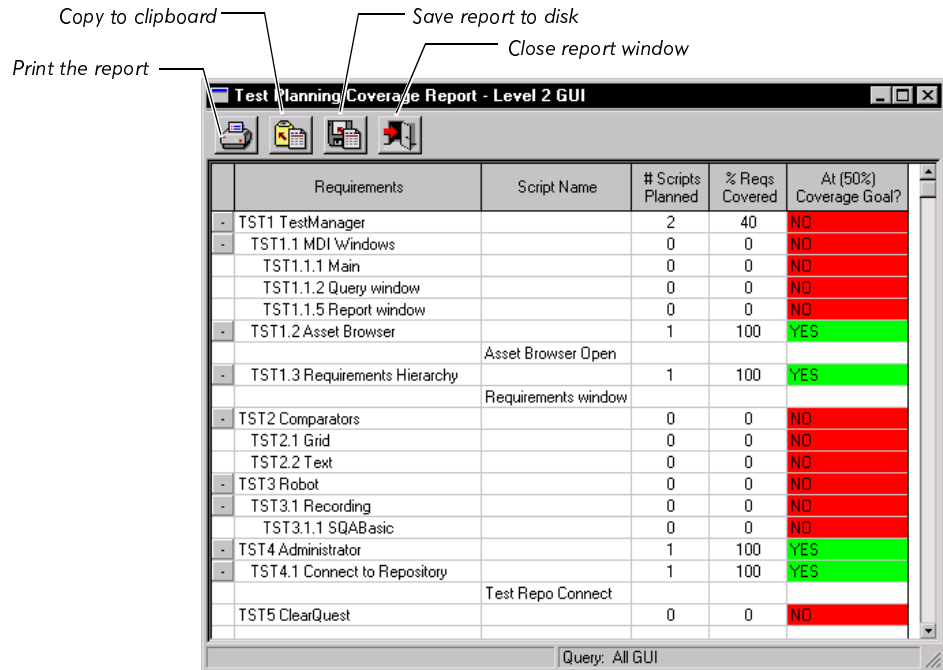
**All requirements of type** – Includes all requirements of the selected requirement type.

**Specific requirement of type** – Includes a single branch of the Requirements Hierarchy for a specific requirement type. In this case, a branch is a single parent requirement and its children.

3. Click **OK** to run the report.



The following figures show a Planning Coverage report and a Development Coverage Report:

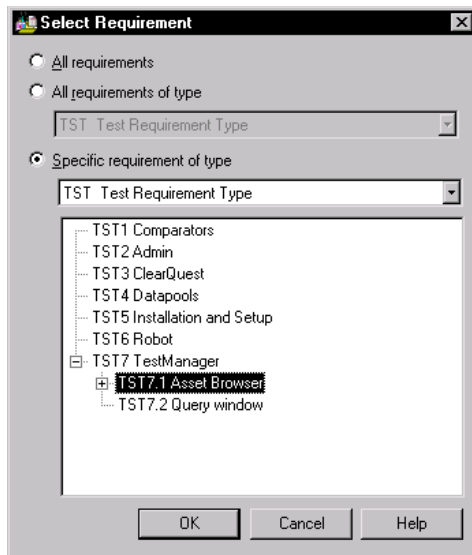


## Running Execution Coverage Reports

After you create an Execution Coverage report, you can run it to see the results. With Execution Coverage reports, you specify the requirement or requirements you want to report on, as well as the set of logs to examine. For more information about logs, see *Generating Log Files* on page 3-2.

To run an Execution Coverage report:

1. Click **Reports** → **Run**, select an Execution Coverage report from the list, and click **OK**, or right-click an Execution Coverage report in the Asset Browser and click **Run**.



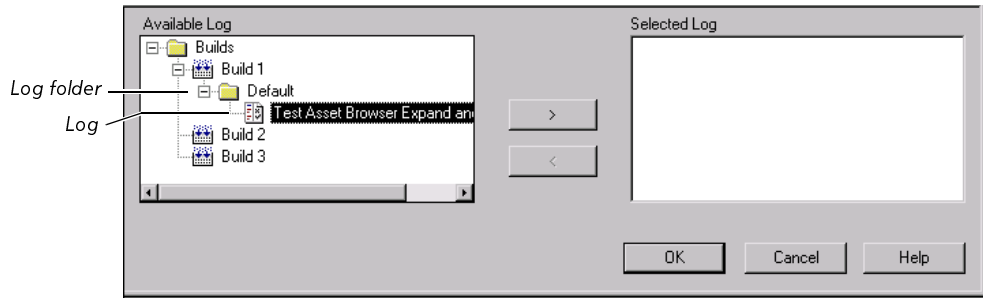
2. Set the requirements that you want to include.

**All requirements** – Includes all requirements, regardless of type, in the Requirements Hierarchy. With the full version of RequisitePro, you can define multiple requirement types. With the baseline version of RequisitePro that is included with TestManager, there is only one requirement type.

**All requirements of type** – Includes all requirements of the selected requirement type.

**Specific requirement of type** – Includes a single branch of the Requirements Hierarchy for a specific requirement type. In this case, a branch is a single parent requirement and its children.

3. Select any number of logs from a build:



4. Click > to move the log into the **Selected Log** list.
5. Click **OK** to run the report.

The following figure shows an Execution Coverage report for a requirement called TST7.1 Asset Browser:

Requirements	# Scripts Planned	# Scripts Executed	% Scripts Executed	At (50%) Executed Goal?	Script Name	# Script Runs	# Script Runs Passed	# Script Runs Failed	% Procs Runs Pass	At (50%) Pass Goal?
- TST7.1 Asset Browser	3	0	0	NO		0	0	0	0	
+ TST7.1.1 Expand/collapse	1	0	0	NO		0	0	0	0	
+ TST7.1.2 Close	1	0	0	NO		0	0	0	0	
+ TST7.1.3 Open	1	0	0	NO		0	0	0	0	

Query: All GUI

## Opening Coverage Reports

At any point after creating a report, you can open it and, if necessary, make changes to the report definition.

To open a coverage report:

- Click **Reports** → **Open**, select a report from the list, and click **OK**, or right-click a coverage report in the Asset Browser and click **Open**.

## Working with the Test Results Progress Report

---

This section includes the following topics:

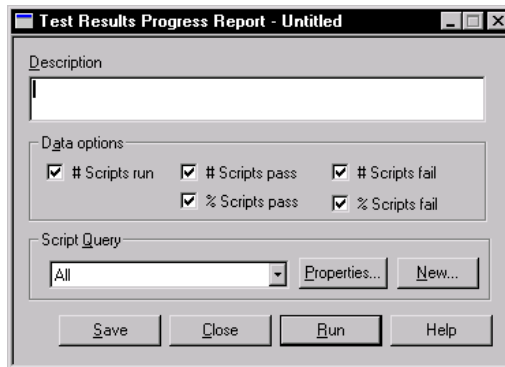
- ▶ Creating a Test Results Progress report
- ▶ Running a Test Results Progress report
- ▶ Opening a Test Results Progress report

### Creating a Test Results Progress Report

With the Test Results Progress report, you can track the success rate of your scripts from build to build. When you create the report, you specify the scripts that you want included. Later, when you run the report, you specify the builds you want to run the report against.

To create a Test Results Progress report:

1. Click **Reports** → **New** → **Test Results Progress**, or right-click the Test Results Progress folder in the Asset Browser and click **New Report**. The Test Results Progress Report specification window appears.



2. Type a description for the report.
3. Set the **Data options**.

**# Script runs** – Includes the total number of script runs covered by the data in the report for all selected builds. For example, if your report covers 50 scripts, and each script is run 10 times, the number is 500.

**# Script passes** – Includes the total number of script runs that resulted in a pass.

**% Script passes** – Includes the percentage of script runs that resulted in a pass.

**# Script failures** – Includes the total number of script runs that resulted in a failure.

**% Script failures** – Includes the percentage of script runs that resulted in a failure.

4. Select a script query from the list. Use the query to narrow down the number of items to be displayed in the report. Optionally, click **Edit** to modify the selected query, or click **New** to create a new query. For more information about queries, see *Creating New Queries* on page 4-3.
5. Click **Save**, type a name for the report, and click **OK**.
6. Click **Close** to close the Test Results Progress Report specification window.

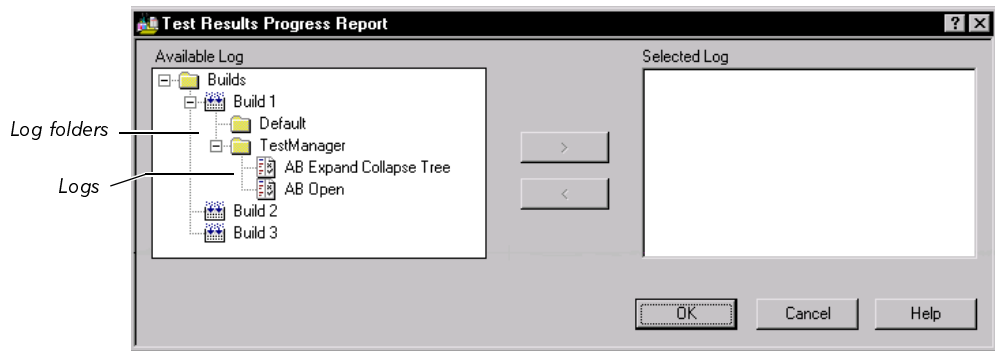
## Running a Test Results Progress Report

After you create a Test Results Progress report, you can run it to see the results. With a Test Results Progress report, you need to indicate the set of logs to examine. For more information about logs, see *Generating Log Files* on page 3-2.

To run a Test Results Progress report:

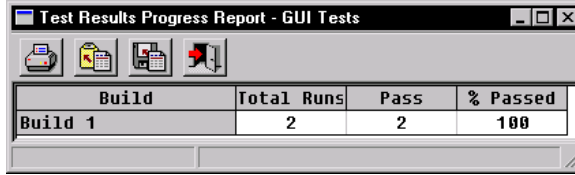
1. Click **Reports** → **Run**, select a Test Results Progress report from the list, and click **OK**, or right-click a Test Results Progress report in the Asset Browser and click **Run**.

The following dialog box appears:



2. Select a log from the list of available logs.
3. Click > to move the logs you want to examine into the **Selected Log** list.
4. Click **OK** to run the report.

The following figure shows a Test Results Progress report:



Build	Total Runs	Pass	% Passed
Build 1	2	2	100

## Opening a Test Results Progress Report

At any point after creating a report, you can open it and, if necessary, make changes to the report definition.

To open a Test Results Progress report:

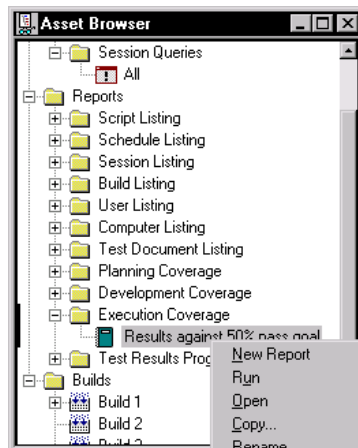
- ▶ Click **Reports** → **Open**, select a report from the list, and click **OK**, or right-click a Test Results Progress report in the Asset Browser and click **Open**.

## Copying, Renaming, and Deleting Reports

---

To copy, rename, or delete a report from the Asset Browser:

1. Right-click a report in the Asset Browser.



2. Do one of the following:
  - Click **Copy**, type a name for the new report, and click **OK**.
  - Click **Delete** and then click **Yes** to delete the report.
  - Click **Rename**, type the new name for the report, and press ENTER.

# Glossary

**ActiveX control** – A reusable software control that takes advantage of Object Linking and Embedding (OLE) and Component Object Modeling (COM) technologies. Developers can use ActiveX controls to add specialized functions to applications, software development tools, and Web pages. Robot can test ActiveX controls in applications.

**actual results** – In a functional test, the outcome of testing an object through a verification point in a GUI script. Actual results that vary from the recorded baseline results are defects or intentional changes in the application. See also *baseline results*.

**Administrator** – See *Rational Administrator*.

**application-under-test** – The software being tested. See also *system-under-test*.

**Asset Browser** – A window that displays testing resources such as builds, queries, scripts, schedules, reports, report output, and logs. The Asset Browser is available in TestManager and LoadTest.

**AUT** – See *application-under-test*.

**automated testing** – A testing technique in which you use software tools to replace repetitive and error-prone manual work. Automated testing saves time and enables a reliable, predictable, and accurate testing process.

**baseline results** – In a functional test, the outcome of testing an object through a verification point in a GUI script. The baseline results become the expected state of the object during playback of the script. Actual test results that vary from the baseline results are defects or intentional changes in the application. See also *actual results*.

**breakpoint** – A feature of the Robot debugger. When you assign a breakpoint to a line of code, and then run the script in the debugger environment, the script stops executing at that line of code. Control returns to you, and the breakpoint line is displayed. From here you can view variables, perform other debugging activities, and continue executing the script.

**build** – A version of the application-under-test. Typically, developers add new features or enhancements to each incremental build. As team members test a build, they enter defects against those features that do not behave as expected. You use TestManager to define and manage builds.

**ClearQuest** – See *Rational ClearQuest*.

**client/server** – An architecture for cooperative processing in which the software tasks are split between server tasks and client tasks. The client computer sends requests to the server, and the server responds.

**data test** – A test that captures the data of an object with the Object Data verification point. See also *internal data test* and *user data test*.

**datapool** – A source of test data that GUI scripts and virtual user scripts can draw from during playback. You can automatically generate datapools using TestManager, or you can import datapool data from other sources such as your database.

**distributed architecture** – Architecture in which computer systems work together and communicate with each other across LAN, WAN, or other types of networks. A client/server system is an example of distributed architecture.

**flow control statements** – In the VU and SQABasic languages, statements that let you add conditional execution structures and looping structures to a script.

**functional test** – A test to determine whether a system functions as intended. Functional tests are performed on GUI objects and objects such as hidden DataWindows and Visual Basic hidden controls.

**Grid Comparator** – The Robot component for reviewing, analyzing, and editing data files for text and numeric verification points in grid formats. The Grid Comparator displays the differences between the recorded baseline data and the actual data captured during playback.

**GUI script** – A type of script written in the SQABasic language. It contains GUI actions such as keystrokes and mouse clicks. Typically, a GUI script also contains verification points for testing objects over successive builds of the application-under-test.

**GUI user** – The type of user that is emulated when a GUI script is executed. Only one GUI user at a time can run on a computer.

**hidden object** – An object that is not visible through the user interface. Hidden objects include objects with a visible property of False and objects with no GUI component.

**IDE** – Integrated Development Environment. This environment consists of a set of integrated tools that are used to develop a software application. Examples of IDEs supported by Robot include Oracle Forms, PowerBuilder, and Visual Basic.



**Image Comparator** – The Robot component for reviewing and analyzing bitmap image files for Region Image and Window Image verification points. The Image Comparator displays differences between the recorded baseline image and the actual image captured during playback. The Image Comparator also displays unexpected active windows that appear during playback.

**internal data test** – A data test provided by Robot for determining the data to capture. You use an internal data test with the Object Data verification point. You cannot edit internal data tests. See also *user data test*.

**log** – A repository object that contains the record of events that occur while playing back a script or running a schedule. A log includes the results of all verification points executed as well as performance data that can be used to analyze the system's performance.

**LogViewer** – See *Rational LogViewer*.

**low-level recording** – A recording mode that uses detailed mouse movements and keyboard actions to track screen coordinates and exact timing. During playback, all actions occur in real time, exactly as recorded.

**object** – An object can be a Windows control such as a window, dialog box, check box, label, or command button. In a generic sense, an object is something that has information associated with it and has actions that can be performed on it. For example, a window is an object. Information associated with the window includes its type and size.

**Object-Oriented Recording**<sup>®</sup> – A script recording mode that examines objects in the application-under-test at the Windows layer. Robot uses internal object names to identify objects, instead of using mouse movements or absolute screen coordinates.

**Object Properties Comparator** – The Robot component that you use to review, analyze, and edit the properties of objects captured by an Object Properties verification point. The Object Properties Comparator displays differences between recorded baseline data and the actual data captured during playback.

**Object Scripting commands** – A set of SQABasic commands for accessing an application's objects and object properties. You add Object Scripting commands manually when editing a script.

**Object Testing**<sup>®</sup> – A technology used by Robot to test any object in the application-under-test, including the object's properties and data. Object Testing lets you test standard Windows objects and IDE-specific objects, whether they are visible in the interface or hidden.

**project** – A collection of data, including test assets, defects, requirements, and models, that can facilitate the development and testing of one or more software components.

**query** – A request for information stored in the repository. A query consists of a filter and several visible attributes — the columns of data to display, the width of the column, and the sort order.

**Rational Administrator** – The component for creating and maintaining repositories, projects, users, groups, computers, and SQL Anywhere servers.

**Rational ClearQuest** – The Rational product for tracking and managing defects and change requests throughout the development process. With ClearQuest, you can manage every type of change activity associated with software development, including enhancement requests, defect reports, and documentation modifications.

**Rational LogViewer** – The Robot component for displaying logs, which contain the record of events that occur while playing back a script or running a schedule. Also, the component from which you start the four Comparators.

**Rational repository** – A database that stores application testing information, such as test requirements, scripts, and logs. All Rational Suite TestStudio and Rational PerformanceStudio products and components on your computer update and retrieve data from the same connected repository. A repository can contain either a Microsoft Access or a Sybase SQL Anywhere database.

**Rational RequisitePro** – The Rational product for organizing, managing, and tracking the changing requirements of your system.

**Rational Robot** – The Rational product for recording, playing back, debugging, and editing scripts.

**Rational SiteCheck** – The Robot component for managing your intranet or World Wide Web site. You can use SiteCheck to visualize the structure of your Web site, and you can use it with Robot to automate Web site testing.

**Rational TestManager** – The Robot component for managing the overall testing effort. You use it to define and store information about test documents, requirements, scripts, schedules, and sessions.

**Report Layout Editor** – The TestManager component for customizing the layout of reports.

**repository** – See *Rational repository*.

**RequisitePro** – See *Rational RequisitePro*.

**Robot** – See *Rational Robot*.

**script** – A file of SQABasic or VU commands. A script can have properties associated with it, such as the purpose of the script, user object properties, and requirements for the script. See also *GUI script* and *virtual user script*.

**shell script** – A script that calls or groups several other GUI scripts and plays them back in sequence. Shell scripts provide the ability to create comprehensive tests and then store the results in a single log.

**SiteCheck** – See *Rational SiteCheck*.

**SQABasic** – The Robot scripting language for recording GUI actions and verifying GUI objects. SQABasic contains most of the syntax rules and core commands that are contained in the Microsoft Basic language. In addition, SQABasic has commands that are specifically designed for automated testing. See also *VU*.

**structural test** – A test to determine whether the structure of a Web site is consistent and complete. A structural test ensures that an application's interdependent objects are properly linked together. You perform a structural test using SiteCheck.

**Synchronizer** – See *Rational Synchronizer*.

**system-under-test** – The system being tested. This includes the computers and any software that can generate a load on the system, networks, user interfaces, CPUs, and memory. See also *application-under-test*.

**test assets** – The resources that facilitate the planning or development phases of the testing effort. Examples of test assets include scripts, schedules, sessions, test documents, and test requirements.

**test development** – The process of developing tests to verify the operation of a software application. This includes creating scripts that verify that the application-under-test functions properly. Test development lets you establish the baseline of expected behavior for the application-under-test.

**test documents** – Test plans, project schedules, resource requirements, and any other documents that are important to your project. You develop your test documents using your own word processing or scheduling program; you then reference the name and location of the document in TestManager. This lets members of the test and development team locate documents quickly.

**TestManager** – See *Rational TestManager*.

**Text Comparator** – The Robot component for reviewing, analyzing, and editing data files for text and numeric verification points in any format except grids. The Text Comparator displays the differences between the recorded baseline results and the actual results.

**unexpected active window** – A window that appears during script playback that interrupts the script playback process and prevents the expected window from being active. For example, an error message generated by the application-under-test is an unexpected active window.

**user data test** – A data test used with the Object Data verification point. A user data test uses a specific property of the object, in conjunction with other parameters, to determine the data to capture. Robot provides some user data tests for your immediate use. You can copy and edit these tests as appropriate for your organization and create your own new data tests. See also *internal data test*.

**verification** – The process of comparing the test results from the current build of the software to its baseline results.

**verification point** – A point in an SQABasic script that confirms the state of one or more objects. During recording, a verification point captures object information from the application-under-test and stores it as the baseline. During playback, a verification point recaptures the object information and compares it to the baseline.

**wait state** – In Robot and TestFactory, a delay or timing condition that handles time-dependent activities.

# ▶ ▶ ▶ Index

## A

adding

- build states 3-10

- complex filter statements in queries 4-8

- custom fields to scripts 2-15

adjusting column widths of Query window 4-10

Adobe Acrobat Reader, installing ix

Asset Browser, displaying builds in 3-3

assigning script names 2-11

associating scripts with test requirements 2-12

attaching scripts to test requirements 2-12

## B

Build Listing reports 5-4

build states 3-4, 3-10

builds

- copying 3-6

- creating 3-5

- deleting 3-6

- organizing in Asset Browser 3-4

- renaming 3-6

## C

changing projects 1-3

child requirements, inserting 2-8

choosing query fields to display 4-5

ClearQuest, opening from TestManager 1-4

columns in queries 4-10

- deleting 4-11

- inserting 4-11

- replacing 4-10

Computer Listing reports 5-4

copying

- builds 3-6

- references to test documents 2-3

- reports 5-18

Coverage reports

- definition 5-2

- opening 5-15

creating

- complex query filters 4-8

- Listing reports 5-4

- new builds 3-5

- queries 4-3

- references to test documents 2-2

- simple query filters 4-7

- Test Results Progress reports 5-16

customer support vi

customizing

- field labels and values 2-19

- script properties 2-15

- scripts 2-18

## D

Data options

- in Development Coverage reports 5-9

- in Planning Coverage reports 5-8

- in Test Results Progress reports 5-16

defining

## Index

- custom field labels and values 2-19
- default build view 3-4
- deleting
  - build states 3-10
  - builds 3-6
  - columns in queries 4-11
  - references to test documents 2-3
  - reports 5-18
  - requirements 2-17
  - scripts 2-16, 4-3
- design specifications, referencing 2-15
- detaching scripts from test requirements 2-14
- Development Coverage reports 5-2, 5-8
- display options
  - in Development Coverage reports 5-9
  - in Execution Coverage reports 5-10
  - in Planning Coverage reports 5-7
- displaying
  - builds in Asset Browser 3-3
  - log properties 3-9
  - script statistics 2-16

## E

- editing 2-3
  - build states 3-10
  - properties of queries 4-9
  - references to test documents 2-3
  - requirement properties 2-8, 2-17
- Execution Coverage reports 5-2, 5-10, 5-14

## F

- filters in queries 4-6
  - creating complex 4-8
  - creating simple 4-7
- functional specifications, referencing 2-15

## G

- generating log files 3-2
- GUI scripts
  - deleting 2-16
  - when to use 2-12

## H

- help desk vi
- Help, using x
- hotline support vi

## I

- inserting
  - child requirements 2-8
  - columns in queries 4-11
  - requirements 2-6
- installing
  - Adobe Acrobat Reader ix

## L

- labeling test requirement revisions 2-7
- listing builds sequentially or by state 3-4
- Listing reports
  - creating 5-4
  - opening 5-6
  - overview 5-2
  - running 5-5
- logging into TestManager 1-3
- logs
  - deleting in TestManager 3-8
  - displaying 3-9
  - generating 3-2
  - overview 1-1
- LogViewer, starting from TestManager 3-9

**M**

- mapping scripts with test requirements 2-12
- modifying build states 3-10
- moving scripts to another requirement 2-15

**N**

- naming scripts 2-11

**O**

- opening
  - Coverage reports 5-15
  - Listing reports 5-6
  - Query Properties dialog box 4-4
  - test documents from TestManager 2-3
  - Test Results Progress reports 5-18

**P**

- planning
  - custom fields in scripts 2-19
  - scripts 2-9
- Planning Coverage reports 5-2, 5-7, 5-8
- Progress Reports 5-2
- projects
  - changing 1-3
  - working with 2-5
- properties of scripts, defining in TestManager 2-11

**Q**

- queries
  - choosing fields to display 4-5
  - columns in 4-10
  - copying 4-9
  - creating 4-3
  - deleting 4-9

- deleting columns in 4-11
- editing properties 4-9
- filtering data in 4-3, 4-6, 4-8
- inserting columns in 4-11
- options 4-9
- overview 1-1, 4-1
- properties of 4-9
- renaming 4-9
- running 4-2
- setting options 4-9
- sorting 4-5
- specifying sort order 4-5
- query properties
  - editing 4-9
  - viewing 4-9
- Query Properties dialog box 4-4
- Query Update button 4-3
- Query window 4-2
  - columns in 4-10, 4-11
  - configuring 4-10
  - deleting scripts from 4-3

**R**

- Rational ClearQuest, opening from TestManager 1-4
- Rational technical support vi
- Rational TestManager
  - changing projects 1-3
  - customizing scripts 2-18
  - defining test requirements 2-4
  - displaying builds 3-3
  - editing query properties 4-9
  - filtering queries 4-6
  - logging into 1-3
  - managing builds 3-2
  - overview 1-1
  - reports. *See* reports in TestManager
  - Requirements Hierarchy in 2-6

## Index

- running queries 4-2
- running reports 5-1
- sorting queries 4-5
- starting 1-2
- test documents 2-1
- window 1-2
- referencing
  - specification files 2-15
  - test documents 2-2
- renaming
  - builds 3-6
  - references to test documents 2-3
  - reports 5-18
- Report Layout Editor 5-4
- reports in TestManager
  - copying 5-18
  - Coverage reports 5-6
  - deleting 5-18
  - layouts 5-4
  - Listing reports 5-4
  - overview 1-1
  - renaming 5-18
  - Test Results Progress reports 5-16
  - types of 5-1, 5-3
- requirements
  - attaching scripts to 2-12
  - attributes and revisions of 2-7, 2-17
  - editing 2-8, 2-17
  - labeling revisions of 2-7
  - managing 2-17
  - types of 2-6
- Requirements Hierarchy
  - building 2-6
  - definition 2-4
  - expanding and collapsing 2-8
- RequisitePro 2-4
- revisions, test requirement 2-6

- running
  - Execution Coverage reports 5-14
  - Listing reports 5-5
  - Planning Coverage or Development Coverage reports 5-11
  - queries 4-2
  - Test Results Progress reports 5-17

## S

- Script Listing reports 5-4
- script properties, defining in TestManager 2-11
- scripts
  - attaching to test requirements 2-12
  - customizing properties 2-15
  - deleting 2-16
  - deleting from Query window 4-3
  - detaching from test requirements 2-14
  - displaying statistics for 2-16
  - moving to another requirement 2-15
  - naming 2-11
  - overview 2-10
  - planning 1-1, 2-9
  - properties of 2-10, 2-15
- specification files, referencing 2-15
- starting TestManager 1-2
- statistics, displaying 2-16
- support, technical vi

## T

- technical support vi
- Test Document Listing reports 5-4
- test documents 2-3
  - creating 2-2
  - renaming references to 2-3
- Test execution progress options, in Execution Coverage reports 5-10
- test plans 2-1



- test requirements
  - attributes of 2-7
  - defining 2-4
  - See also* requirements
- Test results options, in Execution Coverage reports 5-11
- Test Results Progress reports 5-2
  - creating 5-16
  - Data options 5-16
  - opening 5-18
  - running 5-17
- TestManager reports. *See* reports in TestManager
- TestManager. *See* Rational TestManager

## U

- usage model for Rational Test 2-9
- User Listing reports 5-4

## V

- viewing
  - logs 3-9
  - query properties 4-9
  - test documents 2-3
- virtual user scripts 2-12

## W

- working with projects 2-5