Configuring Rational Suite

Product Version	Rational Suite 2000.02.10
Release Date	April 2000
Part Number	800-023317-000

Rational®
the e-development company™

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

IMPORTANT NOTICE

Contents

Preface

1 Welcome to Rational Suite

Rational Suite Editions 13

2 Rational Suite Administration

Installing Rational Suite Software 15
Licensing Rational Suite 15
Before You Plan 15
Developing a Database Plan 17
Creating and Managing a Rational Repository 17
Writing Synchronizer Rules 17
Customizing Extended Help 18

3 Creating a Rational Repository

What is a Rational Repository? 19
Planning the Repository 19
Which Rational Tools Will the Team Use? 20
Which Commercial Databases Will You Use? 20
Will You Use Existing Rational Data? 21
Using the Rational Administrator 21
Setting Up the Repository 22
High-Level Steps for Setting Up a Rational Repository 22
Upgrading Existing Data 22
Installing and Configuring Databases 24
Deciding Where to Locate the Repository 25
Creating the Repository 25
Adding One or More Projects 26

Adding Users, Groups, and Computers 26
Attaching Existing Rose Models 27
Placing Data under Configuration Control 27
Using Rational Suite without a Repository 27

4 Configuring the Integration for ClearQuest and RequisitePro
Overview: Setting Up the Integration 29
Creating a Rational Repository 30
Configuring the ClearQuest Schema Repository Databases 30
Upgrading the ClearQuest Schema Repository Database 31
Upgrading the ClearQuest User Database 32
Configuring the RequisitePro Repository Project 32
Upgrading the RequisitePro Repository Project 33
Collecting Setup Information 34
Updating the ASCQISetup.bat file 35
Running the Batch File 37
Upgrading the Associations 37

5 Rational Synchronizer
Overview of Using the Synchronizer 39
Example 39
Creating and Updating Items 40
In this Chapter 40
Using the Synchronizer 41
Starting the Synchronizer 41
Startup Context: What to Expect 41
Working with the Synchronizer 42
Administration for the Synchronizer 42
Importing Rules 43
Writing Synchronizer Rules 44

6 Rational Suite Out-of-the-Box Tools

RequisitePro Out-of-the-Box Project 56

Creating a Project Based on an Existing Project 56

The Rational Rose RUP Model 57

Tool Mentors 57

Extended Help 58

Viewing Extended Help 58

Customizing Extended Help 58

7 Additional Resources

Online Release Notes 63

Error Message 63

EHRegister 63

Index

Tables

Table 1: Rational Suite Tools 14
Table 2: Commercial Database Choices 20
Table 3: ClearQuest – RequisitePro Configuration 30
Table 4: Sample Rules 51
Table 5: Synchronizer Rule Syntax 52
Table 6: Rational Suite Sample Projects 55
Table 7: RequisitePro Out-of-the-Box Project 56
Table 8: Valid Product Subtool Combinations 60

Preface

This manual describes how to set up Rational Suite for use by a development team. Rational Suite delivers a comprehensive set of integrated tools that embody software engineering best practices and span the entire software development life cycle. Rational Suite's unparalleled level of integration improves communication both within teams and across team boundaries, reducing development time and improving software quality.

Audience

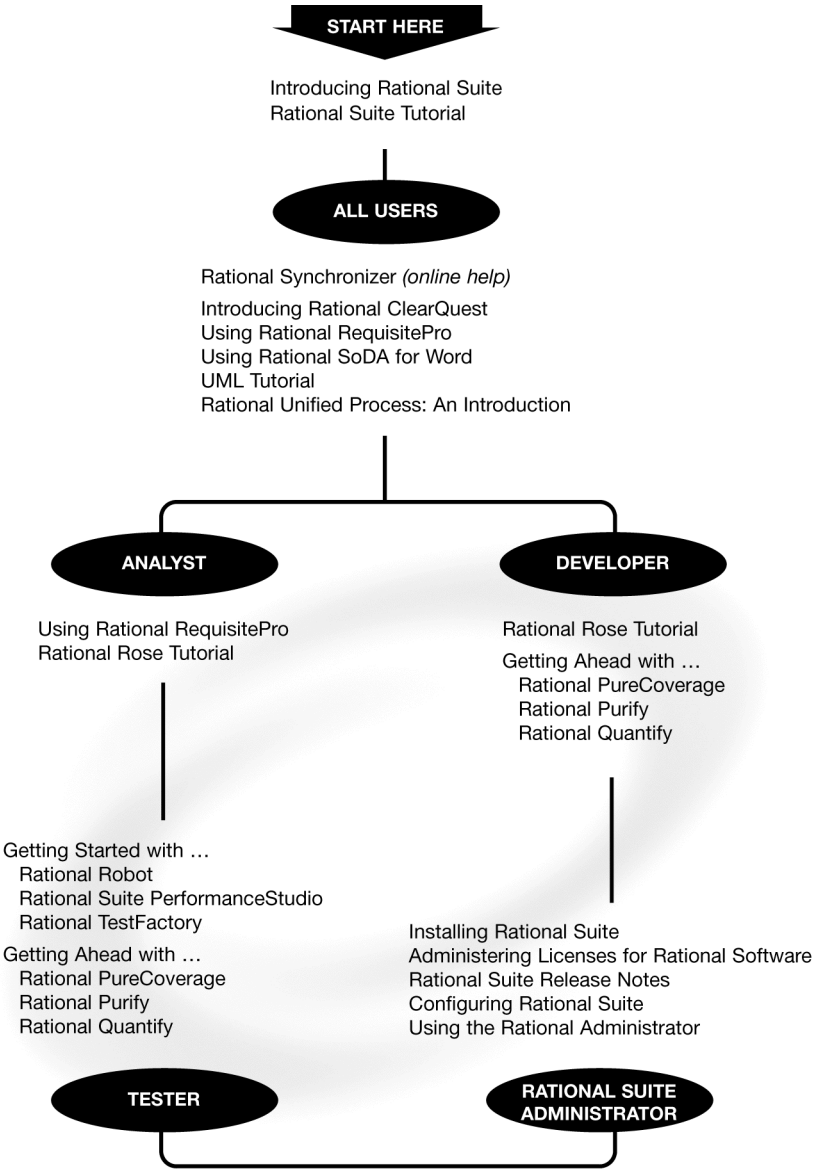
This guide is intended for Rational Suite administrators.

Other Resources

- Online Help is available for Rational Suite.

  From a Suite tool, select an option from the **Help** menu.
- All manuals are available online, either in HTML or PDF format. The online manuals are on the Rational Solutions for Windows Online Documentation CD.
- If you install Rational Suite DevelopmentStudio – RealTime Edition, PDF versions of the manuals for Rose RealTime are installed in `%ROSERT_HOME%\help`. All other manuals are available on the Rational Solutions for Windows Online Documentation CD.
- For more information on training opportunities, see the Rational University Web site: `http://www.rational.com/university`.

Rational Suite Documentation Roadmap



Contacting Rational Technical Publications

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

Contacting Rational Technical Support

If you have questions about installing, using, or maintaining this product, contact Rational Technical Support as follows:

Rational Technical Support Information

Location	Contact Information	Notes
U.S. and Canada	800-433-5444 781-676-2450 support@rational.com	When sending e-mail: – Specify the product name in the subject line, for example, “Rational Suite”. – For existing issues, include your case ID in the subject line.
Europe	+31 (0) 20 4546 200 support@europe.rational.com	
Asia Pacific	+61-2-9419-0111 support@apac.rational.com	

1 Welcome to Rational Suite

Rational Suite is an integrated set of software development products. It combines best-in-class tools with the industry's best practices encapsulated by the Rational Unified Process.

This book shows you how to configure Rational Suite, including how to set up databases for Rational Suite products, how to use and customize the Synchronizer and Extended Help, and how to use predefined projects to get started quickly.

Additional information for each Rational Suite product is located in each product's online Help and in printed documentation. See also *Tool Mentors* on page 57, which describes features to help you use the Rational Unified Process with Rational Suite.

Rational Suite Editions

Rational Suite is available in these editions, tailored for each member of your team:

- Rational Suite AnalystStudio — optimized for the analyst
- Rational Suite DevelopmentStudio — optimized for the developer
- Rational Suite DevelopmentStudio – RealTime Edition — optimized for the developer of real-time, embedded systems.
- Rational Suite TestStudio — optimized for the quality engineer
- Rational Suite PerformanceStudio — optimized for the performance test engineer
- Rational Suite Enterprise — complete set of software development tools

The following table lists the products in each Rational Suite edition.

Table 1: Rational Suite Tools

Rational Tool	Analyst Studio	Development Studio (Windows/Unix)	Development Studio - RealTime Edition	Test Studio	Performance Studio	Enterprise
Rational Unified Process	X	X	X	X	X	X
RequisitePro	X	X	X	X	X	X
ClearQuest	X	X	X	X	X	X
SoDA	X	X	X	X	X	X
Rose	Modeler Edition	Enterprise Edition	RealTime Edition		Enterprise Edition	Enterprise Edition
Robot				X	X	X
TestFactory				X	X	X
PureCoverage		X	X	X	X	X
Purify		X	X	X	X	X
Quantify		X	X	X	X	X
LoadTest					X	
Performance Architect					X	

2 Rational Suite Administration

This chapter provides an overview of the tasks typically performed by a Rational Suite administrator.

Installing Rational Suite Software

The Rational Software Setup program installs Rational Suite and other Rational Software products, using a consistent and familiar installation wizard.

You can find additional details about Rational Software installation in *Installing Rational Suite*, included with your Rational Suite media kit. See the Rational Windows Setup online Help for assistance while using the program.

Licensing Rational Suite

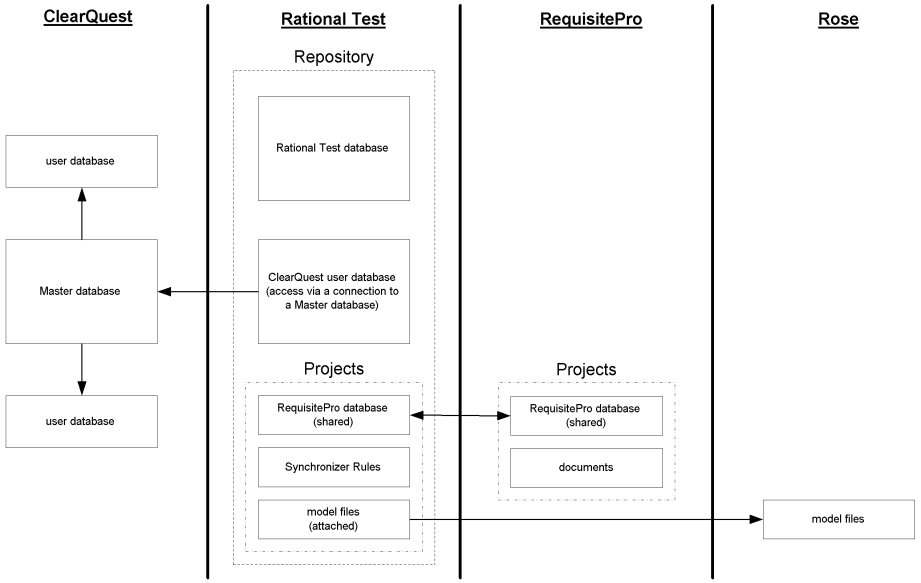
Rational Suite uses FLEXlm licensing services from GLOBEtrotter, Inc. For more information about licensing, please refer to one of the following sources:

- *Administering Licenses for Rational Suite*, included with your Rational Suite media kit.
- Online Help for the Rational License Key Administrator. This provides step-by-step instructions for installing startup license keys and requesting permanent client and server license keys.

Before You Plan

Before you set up your data stores and repositories, you should plan. Before you can plan, though, you need to understand logical relationships among the databases used by Rational Suite products, as shown in the following figure.

Rational Suite Database Relationships



Some notes concerning the figure:

- The arrows between the Master database and the two user databases indicate logical connections; information in the user databases are subsets of information in the Master database.
- The arrow between the Master database and the ClearQuest user database indicates access to the user databases through the Master database.
- The arrow between the two RequisitePro databases indicate that both databases are actually the same database.
- The arrow between the two model files indicate that both files are actually the same file.

Developing a Database Plan

As an administrator, you need to decide which database management system(s) (DBMS) to use with Rational Suite products.

"Creating a Rational Repository" on page 19 provides more information about planning for and installing databases.

Installing Rational Suite, included with your Rational Suite media kit, describes the installation, configuration, and administration of different database management systems, particularly for use with ClearQuest.

Creating and Managing a Rational Repository

A Rational repository is a logical collection of data stores that associates the data you use on a Rational Suite project. Once you create a repository, you can take advantage of integrations among Rational Suite tools.

"Creating a Rational Repository" on page 19 provides more information about creating and configuring a repository.

The *Using Rational Administrator* manual, included with your Rational Suite media kit, describes how to manage a repository.

Writing Synchronizer Rules

The Rational Synchronizer helps maintain project consistency, resulting in improved team communication. The Synchronizer works by creating and updating a project's related items based on the existence of other items in your project.

As an administrator, you may be responsible for analyzing your team's development process and using the results of the analysis to develop Synchronizer rules.

For more information about this process, please see "Rational Synchronizer" on page 39.

Customizing Extended Help

Extended Help provides a connection between Rational Suite tools and the Rational Unified Process. You can augment Extended Help with your own team's process descriptions.

For more information about customizing Extended Help, please see “Rational Suite Out-of-the-Box Tools” on page 55.

3 Creating a Rational Repository

This chapter describes how to create a Rational repository for use with Rational Suite.

What is a Rational Repository?

A Rational repository is a logical collection of data stores that associates the data you use on a Rational Suite project. A Rational repository consists of:

- One Rational Test database
- Zero or one ClearQuest databases
- One or more projects containing:
  - One Rational Suite Synchronizer rules file
  - One RequisitePro database
  - Zero, one, or more Rose model files

Once you create a Rational repository, you can take advantage of product integrations. For example, a repository allows you to:

- Use Test Manager to view test requirements stored in RequisitePro.
- From Rational LogViewer, generate a defect in ClearQuest.
- Work with the Rational Synchronizer to create or update project data based on the existence of other data.

The amount of integration you can achieve often depends on the ClearQuest database schema you use; see the book, *Administering Rational ClearQuest* for more information.

Planning the Repository

Before you create a repository, you need to do some initial planning, as described in the remainder of this section.

Which Rational Tools Will the Team Use?

Your team may want to use all the products in Rational Suite or just a subset of the products. Determine which Suite products your group plans to use.

Which Commercial Databases Will You Use?

For each Suite product you are using, decide which commercial database to use. You can use different databases for different Suite products. The following table summarizes the choices based on the size of the user group planned for each product; please refer to product documentation for more details about each choice.

Table 2: Commercial Database Choices

	ClearQuest	RequisitePro	Rational Test
Microsoft Access	Small groups	Small or medium groups	Small groups
SYBASE SQL Anywhere Server	Medium or large groups	N/A	Medium or large groups
Microsoft SQL Server	Large groups	Large groups	N/A
Oracle	Large groups	Large groups	N/A

Database performance depends on factors such as: number of users, network response time, number of records in the database, and database tuning.

Microsoft Access Runtime is automatically installed when you install at least one Rational Suite product on your machine. (The Access development environment is not installed.) Also, Sybase SQL Anywhere is available on the product CD, but it is configured to work only with Rational products.

Will You Use Existing Rational Data?

Does your project include existing data created with earlier versions of Rational Software? If so, collect information about which data you want to use and where it's located. For example, if you've used earlier versions of Rational tools, you might want to continue using the following types of data:

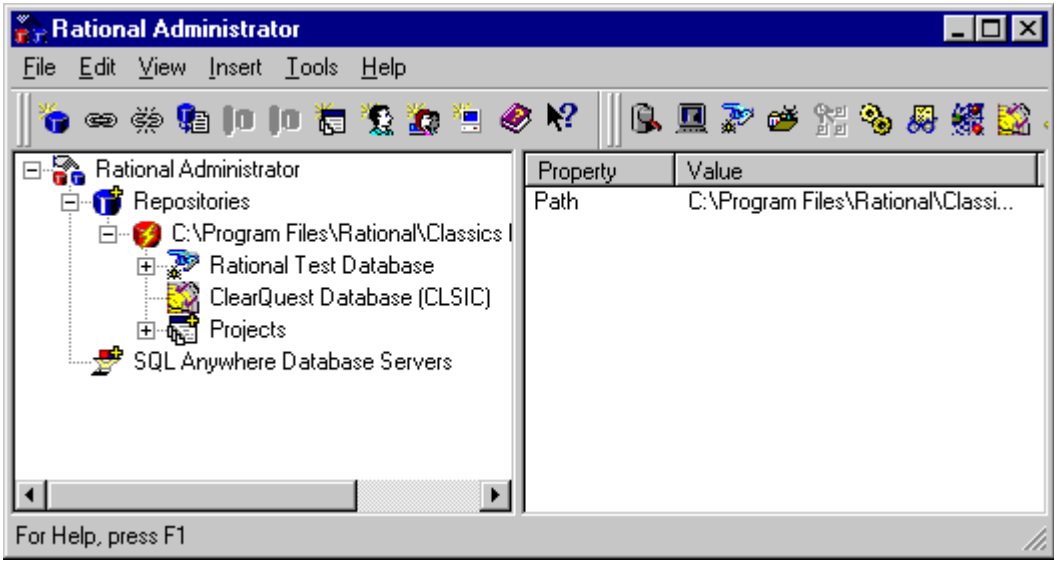
- ClearQuest change requests or schema repositories
- SQA Manager defects
- RequisitePro or SQA Manager requirements
- Rose models

We recommend that you upgrade existing data before you start production work on your project; see *Upgrading Existing Data* on page 22.

Using the Rational Administrator

To create or manage a Rational repository, you use the Rational Administrator. To start the Administrator, click the **Start** button, then click **Programs > Rational Suite > Rational Administrator**.

The Rational Administrator



This chapter does not provide complete information about the Administrator. For more information, refer to the Administrator's online Help or to the book, *Using the Rational Administrator*.

Setting Up the Repository

This section describes, at a high level, how to set up a Rational repository. As you set up a repository, if you need more details, please see the documentation for your commercial database, for an individual Suite product, or for the Rational Administrator.

High-Level Steps for Setting Up a Rational Repository

To set up a Rational repository, perform the following steps, which are described in the remainder of this section:

1. Upgrade any existing data.
2. Install and configure databases.
3. Decide where to locate the repository.
4. Create the repository.
5. Add one or more projects.
6. Add users, groups, and computers.
7. Attach existing Rose models.
8. Place data under configuration control.

Upgrading Existing Data

If you want your group's repository to use data you've created with earlier versions of Rational tools, then you first need to upgrade the existing data. You may also need to upgrade the commercial database you are using. General steps are as follows:

1. Back up existing data. Make sure that you keep a safe copy of your data so that you can revert to it if you run into problems during the upgrade.
2. If necessary, upgrade the databases used for your data. Depending on the database you have decided to use, there may be a newer version available. For your convenience, we recommend that you upgrade your database before you create the repository.

   The remainder of this section describes steps to take for specific products.

Upgrading ClearQuest Data

To continue using data created in an earlier version of ClearQuest, upgrade by using the ClearQuest Maintenance tool. This tool:

- Upgrades the schema repository.
- Upgrades the user database.
- Optionally moves the schema repository to a new database.

To start the ClearQuest Maintenance tool, click the **Start** button, then click **Programs > Rational Suite > Rational ClearQuest > Rational ClearQuest Maintenance Tool**. Refer to *Installing Rational Suite* for additional ClearQuest upgrade and configuration information.

Upgrading RequisitePro Data

To continue using data created in an earlier version of RequisitePro, upgrade by using the RequisitePro upgrade wizard. The wizard:

- Works with existing databases from RequisitePro 2.*x* and later.
- Makes a backup copy of the original database.
- Upgrades in place, overwriting the existing database (but not the backup copy).

The wizard starts when you open a RequisitePro project that contains data created with an older version of RequisitePro.

Upgrading Rational Test Data

SQA Suite 6.*x* preceded Rational Test. To continue using data created in SQA 6.*x*, ensure that:

- SQA Suite 6.*x* is installed on the machine where you are creating the repository.
- The SQA Suite repository is accessible from the machine where you are creating the repository.

When you create the repository, the Administrator converts the SQA 6.*x* data to its new format.

After you create the repository, you can remove SQA Suite 6.*x* from the system containing the repository. For more information, pleased see "Upgrading an SQA 6.x Repository" in *Using the Rational Administrator*.

Upgrading Rose Data

To continue using data created in Rose 3.*x*, use the current version of Rose to open the model. Rose converts the model. Make sure that you save the model before closing Rose.

Installing and Configuring Databases

In the planning phase, you chose the commercial databases to work with (see Table 2 on page 20). It is now time to install and set up those databases, following these guidelines:

- This step is mandatory if you are using Sybase SQLAnywhere, Microsoft SQL Server, or Oracle. Install the database, then perform any set-up activities (such as setting privileges and tuning the database) appropriate for that database. For more information, refer to the documentation for the database and to documentation for the product with which you will use the database.
- If you are using Microsoft Access on a server, install it on the server before creating the repository. Alternately, installing Rational Suite on any machine automatically installs Access Runtime. For Access, you do not have to perform database set-up activities before you create the repository.

Deciding Where to Locate the Repository

Determine where the repository will reside, following these guidelines:

- The Test database must reside in the repository directory structure. The disk that will contain the repository must have enough space to contain at least the Test database.
- The repository contains pointers to the other tools' databases. Therefore, these databases can reside anywhere on your network system.
- Before proceeding, create the top-level directory of the repository. Make sure the directory is empty and make it a shared directory. If you do not share the directory, the repository will not be available to members of your team.

Creating the Repository

To create the repository:

1. Ensure that you have Windows NT Administrator privileges.
2. Start the Rational Administrator: click the **Start** button, then click **Programs > Rational Suite > Rational Administrator**.
3. From the Administrator menu, click **File > Create Repository**.
4. In the Create Repository – Path dialog box, enter the name of the top-level directory that you created in *Deciding Where to Locate the Repository* on page 25. Click **Next**.
5. Configure the repository by specifying:
   - The database you plan to use for Rational Test
   - Whether to create a new Rational Test database or whether to use existing data
   - Whether to use ClearQuest, and if so, whether to create a new ClearQuest database or attach to an existing one.

Adding One or More Projects

When you create a repository, the Rational Administrator creates a default project (named *Default Project*) containing:

- A reference to an empty RequisitePro database
- A reference to a Synchronizer rules file. (The rules file does not actually get created until you run the Synchronizer, import rules, and save the rules file.)

We recommend that you use a project with a more meaningful name. To create a new project:

1. From the Administrator menu, click **Insert > Project**.
2. Follow the prompts. You are asked to:
   - Supply the project name.
   - Indicate whether to create a new RequisitePro database or use an existing one.
   - Choose the Rose model or models to include. You can attach Rose models later.
3. Optionally, delete *Default Project*. (A repository requires the existence of at least one project.)

Adding Users, Groups, and Computers

Configure Rational Test, ClearQuest, and RequisitePro as described in the following sections.

Rational Test

Use the Rational Administrator to add users, groups, and computers to the Rational Test database. If you initialized the Test database with data from an existing Test or SQA repository, the users, groups, and computers defined in the existing data appear in the new repository.

ClearQuest

If you are using ClearQuest, use the ClearQuest Designer to set up user and group access to the ClearQuest database. Refer to the ClearQuest Designer online Help for more information.

RequisitePro

If you are using RequisitePro, you can use project security for the RequisitePro database. Refer to the RequisitePro online Help for more information.

Attaching Existing Rose Models

You can attach Rose models to a project after you create the project. This feature allows you to create models after you create the repository.

1. From the Rational Administrator, right-click on a project.
2. On the shortcut menu, click **Attach Rose Model**.
3. Use the File Browse dialog box to select the model to use.

   You cannot share a Rose model among more than one project. You can attach up to seven Rose models to a repository project.

Placing Data under Configuration Control

We recommend that you use a configuration control program such as ClearCase to manage the repository and the databases associated with the repository.

Using Rational Suite without a Repository

We recommend that you use a repository to work with Rational Suite. It is possible to use some editions of Rational Suite, specifically those editions that do not use Rational Test, without a repository. Rational Test requires a repository.

To use Rational Suite AnalystStudio or Rational Suite DevelopmentStudio without a repository:

1. Set up ClearQuest, RequisitePro, and Rose models as if you were using them in stand-alone mode.
2. Use Rational Synchronizer without a repository, as detailed in the chapter on the Synchronizer.

4 Configuring the Integration for ClearQuest and RequisitePro

The integration between ClearQuest and RequisitePro allows you to associate qualified enhancement requests or defects with related requirements. After you collect stakeholder feedback as enhancement requests or defects, you validate, or *qualify*, the requests you wish to implement. Then use the integration to establish associations between the originating enhancement request or defect and related requirements.

Using the integration provides a seamless combination of ClearQuest and RequisitePro. Working in ClearQuest, you can associate or create new requirements from enhancement requests or defects. Working in RequisitePro, you can view the original enhancement requests or defects on which the requirement is based, as well as establish new associations.

The integration allows you to track stakeholder input during the requirements phase. This ensures that all qualified stakeholder input is addressed.

Overview: Setting Up the Integration

This section provides a summary of the steps for configuring Rational ClearQuest and RequisitePro to enable data association between the two software products.

Note: This guide assumes that you installed Rational software on the C drive. Substitute your actual installation drive name, as needed.

Table 3, ClearQuest – RequisitePro Configuration, summarizes the steps for setting up the ClearQuest and RequisitePro integration. The remainder of this chapter describes these steps in detail.

Table 3: ClearQuest – RequisitePro Configuration

Action	Comment
Create a Rational repository	This repository must contain the ClearQuest database and the RequisitePro project which will be integrated.
Configure the ClearQuest database	New packages have been developed to support data association with RequisitePro.
Configure the RequisitePro project or projects	New attributes are added which are the shared resource between ClearQuest and RequisitePro.
Edit and run the setup file	This file contains a description of your Rational environment, including file locations and names.
Upgrade the Associations	Upgrade the Repository, ClearQuest, and RequisitePro for logon information.

Creating a Rational Repository

The ClearQuest database and the RequisitePro projects must be part of the same Rational repository before the integration is active. For more information on setting up a Rational repository, see Chapter 3, *Creating a Rational Repository*, on page 19.

Configuring the ClearQuest Schema Repository Databases

Each ClearQuest user database containing related data is associated with a schema repository database (also called a master database). Both databases must be configured to access the packages specific to the integration between ClearQuest and RequisitePro.

If you are creating new ClearQuest databases to be used for your development project, Rational provides several out-of-the-box (OOTB) schemas that contain all necessary packages to make the integration available. Those schemas are named AnalystStudio, TestStudio, DevelopmentStudio, and Enterprise schemas. Use any of these schemas to create your new schema and the ClearQuest schema database configuration is complete. For more information on using the out-of-the-box components, see Chapter 6, *Rational Suite Out-of-the-Box Tools*, on page 55.

If you have used the integration in an earlier Suite version, you need only to upgrade your ClearQuest databases to include additional packages.

Upgrading the ClearQuest Schema Repository Database

To upgrade your existing AnalystStudio schema database, begin by consulting the section on upgrading the database in the guide, *Administering Rational ClearQuest*. This document provides detailed recommendations about upgrading your schema database.

Note: Before upgrading, we strongly recommend that you back up the following data:

- ClearQuest Schema Repository database
- ClearQuest User database
- The entire tree of directories and files belonging to the Rational Repository

To upgrade the ClearQuest schema repository database:

1. Use the ClearQuest Designer to log on to your schema repository database with an administrator profile. The Open Schema window appears. Click **Cancel**.
2. From the ClearQuest Designer menu, click **Package > Package Wizard**. The Package Wizard window appears.
3. In the Select a Package list, identify the latest version of the **RequisitePro** package (the package with the highest version number).
4. Click **More Packages**. If there is a **RequisitePro** package with a higher version number than the one you identified in step 3, select it. Click **OK** to return to the Select a Package list.
5. Select the latest version of the **RequisitePro** package. Click **OK**.
6. In the Package Wizard, click **Next**. Select the **AnalystStudio** schema. Click **Next** again**.**
7. Place a check mark in all record types that you will use in the integration. We recommend that you select at least **Defect** and **EnhancementRequest**. Click **Finish**.

8 From the ClearQuest menu, select **File > Checkin** to check this new package into your schema. Click **OK** after verifying your choice. Click **OK** again in the ClearQuest checkin confirmation dialog box.

9 Repeat this procedure to add the latest version of the **RequisitePro Supplement** package.

Upgrading the ClearQuest User Database

To upgrade the ClearQuest user database:

1 Back up your database before upgrading it.

2 From the ClearQuest menu, select **Database > Upgrade Database** and click **Yes** to continue.

3 In the Upgrade Database window, select the database instance that is included in your Rational repository and that has a previous AnalystStudio schema. Click **Next**.

4 Select the latest version of the schema (the schema with the highest version number). Click **Finish**.

5 Click **OK** and close the ClearQuest Designer.

Your ClearQuest databases are now configured to use the ClearQuest – RequisitePro integration.

Configuring the RequisitePro Repository Project

Each RequisitePro project contained in a Rational repository must be configured to share data with ClearQuest's enhancement request and defect records. To enable this data sharing, you need to add two new attributes to the project or projects. These attributes display the ClearQuest record ID associated with the requirement in the RequisitePro database view. You can create new associations from the requirement to a related defect or enhancement request.

If you are creating a new RequisitePro project, Rational provides an out-of-the-box (OOTB) RequisitePro Project Template that contains the necessary attributes to make the integration available. Use this template to create your new project and the RequisitePro project integration configuration is complete. For more information on using the out-of-the-box components, see Chapter 6, *Rational Suite Out-of-the-Box Tools*, on page 55.

Upgrading the RequisitePro Repository Project

To upgrade the RequisitePro project:

1. Back up your RequisitePro project before beginning the upgrade.
2. Launch RequisitePro and open the related Rational repository project.
3. From the RequisitePro Tool Palette click **Project > Properties**.
4. On the Attributes tab, from the Requirement Type list, select the **FEAT** requirement type. Click **Add**.

   Note: If you are not using the OOTB RequisitePro project, you may need to create a FEAT (feature) requirement type. For instructions on completing this task, see the RequisitePro online Help. Alternately, see "Updating the ASCQISetup.bat file" on page 35 to learn about setting up customized requirement types.
5. In the Add Attribute window:
   - In the Label field, enter `EnhancementRequest`.
   - From the Type list, select **ClearQuest integration**.
   - Click **OK**.
6. On the Project Properties window, click **Add** to return to the Add Attribute window.
7. In the Add Attribute window:
   - In the Label field, enter `Defect`.
   - From the Type list, select **ClearQuest integration**.
   - Click **OK**.
8. Save and close the RequisitePro project.

These two new attributes in RequisitePro can now be populated with ClearQuest record IDs when a requirement is associated with a defect or an enhancement request. This makes it possible to track the origin of the associated requirement as well as the realization of the enhancement request or the resolution of the defect.

Collecting Setup Information

Continue configuring the integration by running a batch file containing commands that use the following variables. Use this form to record your project's values. You will use this information to update the batch file.

Batch Command	Description	Your Values
set CQDatabaseName =	The name of your ClearQuest User database	
set H =	The hard drive letter where RequisitePro is installed, followed by a colon (for example C:)	
set U =	The logon name for the ClearQuest database, the RequisitePro project, and the Rational Repository	
set P =	The path to your local Rational installation directory	
set RP =	The name of the RequisitePro project	
set RQS =	The full path to the RequisitePro *.rqs file	
set Help=	The full path to the ClearQuest - RequisitePro Integration help file (Program Files\Rational\ RequisitePro\ Help\CQI.hlp)	

Updating the ASCQISetup.bat file

Now use the information in the preceding table to complete the `ASCQISetup.bat` file. The file is in the directory `Program Files\Rational\Common`.

Begin by copying the `ASCIQSetup.bat` file to a new file such as `MySetup.bat` to preserve the original file. To edit the batch file, locate it in Windows Explorer, right-click the file, and select **Edit** from the context menu. A text editor opens the file. Locate each `set` command and edit it to contain the information in the preceding table.

The following example of a completed `MySetup.bat` has been edited to work with the integration for the Classics Online project, used throughout *Getting Started with Rational Suite AnalystStudio*. All edited or updated fields are in bold. The REM command precedes a comment.

Note: Text in this font is commentary and is not part of the batch file, itself.

```
REM Before you launch this batch file, please update the following
information:

REM **** INFORMATION TO UPDATE ****

REM Update the name of the ClearQuest database
set CQDatabaseName=CLSIC

REM Update the hard drive letter where your local Rational products are
installed
set H=C:

REM Update the SuperUser login name for ClearQuest database, RequisitePro
and Rational Repository
set U=admin

REM Update the path to your local Rational installation directory
set P=\Program Files\Rational\

REM Update the name of your RequisitePro project
set RP= Order Processing and Fulfillment System
```

```
REM Update the full name to your RequisitePro .rqs file.
REM It should be a UNC path if you want to access your project from
different machines
set RQS=C:\Program Files\Rational\Classics Demo\ClassicsRepo\Project\
ClassicsPOS\requisite\requisite.rqs

REM Update the path to the cqi.hlp file
REM It should be a UNC path if it is on a different machine.
set Help=C:\Program Files\Rational\RequisitePro\help\cqi.hlp

REM **** BEGINNING OF THE SCRIPT ****

%H%
cd %P%\common
```

If you are enabling multiple RequisitePro projects for the integration, copy the line containing “set RP= XXXX” to here. Update this command with the name of the RequisitePro project. Add one “set RP = “ line here for each RequisitePro project that you want to integrate with ClearQuest.

Also copy the line containing “set RQS = XXXX” to here and update it to reflect the full path to your RequisitePro *.RQS file. Add one “set RQS = “ line for each RequisitePro project that you want to integrate with ClearQuest.

Use a UNC pathname for projects that are located on other machines on your network.

```
REM Creation of the association between the 'Defect' ClearQuest record type
and the RequisitePro attribute 'Defect' of the Feature requirement type

ASCQISetup.exe -c %U% %CQDatabaseName% Defect Requirements_List
Defects_list RepoProject Modify "Associate Defects" "%Help%" 2 "%RP%"
"%RQS%" FEAT Defect 3

REM Creation of the association between the 'EnhancementRequest' ClearQuest
record type and the attribute 'EnhancementRequest' of the Feature
requirement type

ASCQISetup.exe -c %U% %CQDatabaseName% EnhancementRequest Requirements_List
EnhancementRequests_list RepoProject Modify "Associate Enhancement
Requests" "%Help%" 2 "%RP%" "%RQS%" FEAT EnhancementRequest 3
```

If you have used RequisitePro requirement types and attributes names other than the ones shown in the above commands, copy the entire `ASCQISetup.exe` command and paste it here. Update the requirement type and the attribute name. Replace FEAT with your requirement type, and Defect or EnhancementRequest with your customized attribute. Copy and update this command here for each requirement type and attribute name to include in the integration.

```
pause
```

The batch file ends with a pause statement.

Running the Batch File

Run your `MySetup.bat` file to create the indicated records and to supply the information specific to your development environment. As the batch file runs, it pauses after each creation of a setup record and prompts you to "Press any key to continue...." Press RETURN until the batch file ends.

Upgrading the Associations

The final step in configuring the ClearQuest – RequisitePro integration is to upgrade the existing associations, the ones created from a previous version of the AnalystStudio schema.

1. Select **Start > Run** from the Windows task bar and browse to `Program Files\Rational\Common\ASCQIUpgrade.exe`.
2. Add the path to your Rational Repository project to the end of this command, enclosing the path in quotes. Click **OK**. (For example: `Program Files\Rational\Common\ASCQIUpgrade.exe` **`"C:\Program Files\Rational\Classics Demo\ClassicsRepo"`**).
3. You may be prompted by each Rational product involved in the integration (Rational Repository, ClearQuest and RequisitePro) for logon information. Log on with an administrator profile.

   Your ClearQuest – RequisitePro integration has now been configured and is ready for use in your development project.

   For more information about using the ClearQuest – RequisitePro integration, refer to the document *Getting Started with Rational Suite AnalystStudio* contained on your documentation CD.

5 Rational Synchronizer

The Rational Synchronizer helps maintain project consistency, resulting in improved team communication. The Synchronizer works by creating and updating a project's related items based on the existence of other items in your project.

For example, the Synchronizer might create a test requirement based on the existence of a feature requirement. Or it might create a test script based on the existence of a test requirement.

The Synchronizer works with the requirements (Rational RequisitePro), modeling (Rational Rose), and test (Rational Test) domains.

Overview of Using the Synchronizer

Generally, when you start the Synchronizer, it displays the Find Wizard. The Find Wizard helps you narrow the scope of items to work with. When you finish using the Find Wizard, the Synchronizer presents a list of items it proposes to create, update, or delete. You select only those items that you want to change. The Synchronizer carries out the changes for the items you chose.

The Synchronizer works by using rules to determine which changes are needed. Rational provides some sample rules to get you started. Your administrator or project leader analyzes your development environment to decide which rules to use, and whether to write new rules or to edit existing ones.

Example

For example, say that you want to ensure that every test requirement in RequisitePro has a corresponding test script in Rational Test. The Synchronizer can enforce this constraint with a rule. When you run the Synchronizer it finds all test requirements for which there is no test script. When you instruct the Synchronizer to proceed, it creates and initializes the test script.

As a result, new developments in the requirements domain are communicated accurately and promptly to the team responsible for testing.

Creating and Updating Items

You can use the Synchronizer to:

- **Create** items in one domain that are related to items in another domain or in the same domain.

  When the Synchronizer creates items, it uses the work of one team to jump-start another team's efforts. For example, the Synchronizer can create test requirements in RequisitePro that correspond to Rose interfaces. As a result, the testing team is automatically notified of new work they need to reflect in test planning.

- **Copy** the values of attributes between items, allowing changes in one application to be visible in another. For example, the Synchronizer can cause a name change in a Rose use case to trigger a corresponding change to a related requirement in RequisitePro.

  Copying attribute values ensures that changes in one item are propagated to related items. This feature eliminates error-prone manual updating.

In this Chapter

The remainder of this chapter discusses:

- Using the Synchronizer (read this section if you plan to use the Synchronizer to create or update items)
- Administration for the Synchronizer (read this section if you plan to set up a project to use the Synchronizer or if you plan to edit Synchronizer rules)
- Comparison with previous integrations (read this section if you have used earlier versions of Rational software)

Using the Synchronizer

You have the Rational Synchronizer if you have installed any edition of Rational Suite or any of the following tools:

- RequisitePro 4.0 or later
- Rose 98i, Rose 2000, or later
- Rational Test 7.1 or later

Starting the Synchronizer

As a user, start the Synchronizer by taking one of the following actions:

- From the RequisitePro or TestManager menu, click **Tools > Rational Synchronizer**.
- From the Rose menu, click **Tools > Synchronize**.
- From the Windows **Start** menu, click **Programs > Rational Suite > Rational Synchronizer**.

As an administrator, you can also start the Synchronizer as follows:

- In the Rational Administrator, select a repository and click **Tools > Rational Synchronizer**.

Startup Context: What to Expect

If you have installed Rational Suite, your project administrator has probably used the Rational Administrator to define a *repository project* for your team. A repository project associates the following items:

- One Rational Test database
- Zero or one ClearQuest databases
- One or more projects containing:
  - One Rational Suite Synchronizer rules file
  - One RequisitePro database
  - Zero, one, or more Rose model files

If your work is part of a repository project, in most cases when you start the Synchronizer, the Find Wizard immediately appears.

In some cases, the Setup Context dialog box appears first, prompting you to identify the repository project to use. Log in to the project, using information supplied by your project administrator. The Find Wizard appears.

If you are not working with a repository project, you cannot permanently identify the set of rules to use when synchronizing. When you start the Synchronizer, the Setup Context dialog box allows you either to use rules that are already defined or to create new rules.

Working with the Synchronizer

To use the Synchronizer, you first run the Find Wizard which helps you identify the items to work with. The Synchronizer displays a list of items based on the selections you make. Each of these items represents an action to create or update part of your project. You review the list to understand the impact of the changes. You select the items you want synchronized. The Synchronizer creates or updates, causing the selected items to become consistent across your project.

For more information about working with the Synchronizer, please see online Help for the Synchronizer.

Administration for the Synchronizer

If your team is using Rational Suite, administration tasks for the Rational Synchronizer consist of the following activities:

- Creating a repository project for your team. (For a brief description of a repository project, see page 41. For more detailed information about working with repository projects, see the manual *Using the Rational Administrator*.)
- Importing rules for your project to use.
- Analyzing your project's development process to determine which Synchronizer rules you need.
- Editing existing rules or creating new rules.

If your team is not using Rational Suite, an administrator may identify which rules to use, but users need to import those rules every time they start the Synchronizer.

Importing Rules

How Rules are Stored

Synchronizer rules are text descriptions of Synchronizer actions. You can store one or more rules in a *rule definition file,* a text file with file type `.rsd`. The Synchronizer ships with sample rule definition files, located in:
`<Install Path>\Common\OutOfTheBox\synchronizer rules\`.
For your project, you will probably create new rules which you may base on the sample rules. You can edit a rule definition file with Notepad or another text editor. For more information, see *Writing Synchronizer Rules* on page 44.

To use a rule in the Synchronizer, you first import it into a *Synchronizer Rules Project*, a file with file type `.rsr`. Importing a rule converts it (compiles it) into the internal representation used by the Synchronizer. The Synchronizer Rules Project stores rules in their internal representation. If you edit a rule, you need to re-import it to make your changes effective.

Using Rational Suite: Importing Rules

If your team is working with Rational Suite, we recommend that you start from the Rational Administrator to import rules for your entire project. A placeholder for a Synchronizer Rules Project (a container for imported rules) is created automatically when you create a new project in the Rational Administrator. The file is actually created the first time you import rules and save the file. The Rules Project is named `Rational Synchronizer Rules.rsr`. It is created in the folder *repository name*`\Project\`*current project*`\`, where *repository name* is the Rational repository and *current project* is the project of interest.

To start the Synchronizer from Rational Administrator, click **Tools > Synchronizer**. For each project, the first time you start the Synchronizer from the Rational Administrator, the Synchronizer displays the following prompt: “There are no rules defined. Would you like to import rule definitions?” To import rules, select **Yes**.

Working without Rational Suite: Importing Rules

If you do not have an edition of Rational Suite installed, you can still use the Synchronizer between domains. You first create the Rules Project file, then import rules into it. When you next start the Synchronizer, use the Setup Context dialog box to open the same Rules Project. In more detail:

1. Start the Synchronizer (from the Windows **Start** menu, click **Programs > Rational Suite > Rational Synchronizer**).

   The Rational Synchronizer – Setup Context dialog box appears.

2. Select **Synchronize without using the Rational Repository**, then select **Create a Synchronizer Rules Project**. Click **OK**.

3. In the Create Synchronizer Rules Project dialog box, specify a directory and file name for the Rules Project. Click **Save**.

   The Synchronizer appears. Also, a message appears asking whether you want to import rule definitions. If you click **Yes**, you are prompted to select the rules to import, and then to save the rules project.

   When you next start the Synchronizer, the Setup Context dialog box again appears. To use the same Rules Project, select **Synchronize without using the Rational Repository**, then **Open an existing Synchronizer Rules Project**.

   The online Help for the Synchronizer provides more details about working with rules.

Writing Synchronizer Rules

This section describes how to design and write Synchronizer rules.

Understanding Your Development Process

Before you start to write rules, you need to analyze your development process to understand your group's needs. The following steps provide an overview of the analysis:

1. Write a narrative description of how your group works with items such as requirements, model elements, and test scripts.
2. From the description, identify the item types you work with and the domain they come from.
3. Select only those items that you want to relate to other items.
4. For each pair of items you want to relate, specify the action to take: create or update. Also describe the properties you want to synchronize, the direction of data flow, the timing of the synchronization, and conditions under which to synchronize.

   From this information, you can create tables that serve as specifications for the rules you need to write.

   A white paper, *Using the Rational Synchronizer*, elaborates on these steps and provides an example. This paper is available from your Rational technical representative.

What Can You Specify in a Rule?

You can write rules to perform the following actions:

- *item creation* – These actions create an item (for example, a RequisitePro requirement or Rose use case) that corresponds to another item.
- *data synchronization* – These actions transfer information between two corresponding items. You can use data synchronization actions only for two items that are already related.

You can write a rule that both creates an item and causes data synchronization. You can also write a rule that just creates an item.

This section presents an annotated Synchronizer rule. This rule is part of the Sample rule set supplied with the Synchronizer, described in the next section. For reference, we present the rule first without annotations:

Sample Rule without Annotations

```
# File:       tr_requirement_to_test_script.rsd
# Purpose:    Sample rule to create a test script in a Rational Test domain
#             project that corresponds to a test requirement in a Rational
#             RequisitePro domain project. This rule associates the test
#             script with the requirement when viewed in the Rational
#             TestManager.


begin rule "Create Test Script from Test requirement"
description
{
For each test requirement in the RequisitePro domain that has the attribute
AutoTest set to True, create a test script object in the Rational Test
domain.
}

             projects RequisitePro to TestStudio
             items "Test Requirement Type" to Script
             properties
                          source.Text init target.Name
                          source."Test Notes" to target."Description"
                          source.Object init target.Requirement
             applies when AutoTest = "True"

end rule
```

Sample Rule with Annotations

Note In the annotated rule, text in this font is commentary and is not part of the rule, itself.

```
 # File:      tr_requirement_to_test_script.rsd
# Purpose:    Sample rule to create a test script in a Rational Test domain
#             project that corresponds to a test requirement in a Rational
#             RequisitePro domain project. This rule associates the test
#             script with the requirement when viewed in the Rational
#             TestManager.
```

You can insert comments anywhere in the rule definition file. A comment extends from a # character to the end of the line.

Good housekeeping: Ensure that the file name describes the rule. Include the file name and an English-language description at the beginning of the file.

```
begin rule "Create Test Script from Test requirement"
```

Every rule starts with a begin clause. Give your rule a meaningful name so that it will be easy to remember what it does. If you use spaces in the name, enclose the name in quotes.

```
description
{
For each test requirement in the RequisitePro domain that has the attribute
AutoTest set to True, create a test script object in the Rational Test
domain.
}
```

Optionally provide a description of your rule. The description appears in the Synchronizer session window when you display detailed information about the rule. The description may be any length, but try to keep it short for easy viewing. Describe the purpose of the rule and the data transfer that it causes. Notice that the text of the description is enclosed in braces { }.

```
projects RequisitePro to TestStudio
```

Identify the projects to which this rule applies. In this example, *RequisitePro* is the source project and *TestStudio* is the target project. The projects are from one of the supported domains: modeling (Rational Rose), requirements (Rational RequisitePro), or test (Rational TestStudio).

The source and target project may be the same, for example, if you are synchronizing use case requirements to test requirements within Requisite Pro. If the project name contains spaces, enclose the name in quotes.

```
items "Test Requirement Type" to Script
```

Specify the item types that this rule will synchronize. If an item name contains spaces, enclose the name in quotes. Item types are case sensitive. Spell them exactly as shown in the following table or as they appear in the domain project.

Project Type	Item Types
Rose	"Use Case" Actor Package “Sequence Diagram” Class
Rational Test	Script
RequisitePro	Any requirement type. Type the name of the requirement (not the tag) exactly as shown in the Requirement Tab of the project properties display in RequisitePro.

```
properties
            source.Text init target.Name
            source."Test Notes" to target."Description"
            source.Object init target.Requirement
```

The *properties* statement describes one or more data transfers. There is one properties clause per rule.

Each transfer is defined by a pair of properties, or attributes, one in the source item and one in the target item. The data transfer may be from the source to the target or from the target to the source. In some rules, there may be data transfers in both directions.

For each transfer, the first property is the source of the data (the source item is called the *broadcaster* for this transfer). The second property is the target of the data (the target item is called the *receiver* for this transfer).

Attributes are case-sensitive. Spell them exactly as shown in the following table, including quotes, or as they appear in the domain project.

Item Type	Valid Properties
Rose "Use Case"	Name, Documentation, Stereotype, Rank
Rose Actor	Name, Documentation, Stereotype
Rose Package	Name, Documentation, Stereotype, Global
Rose “Sequence Diagram”	Name, Documentation
Rose Class	Name, Documentation, Stereotype, “Export Control”, Cardinality, Space, Persistence, Concurrency, Abstract
Rational Test Script	Name, “Description” (must be in quotes), Requirement (see Note 1 below)
RequisitePro requirement	Any attribute for the requirement type. Enclose in quotes if the attribute contains spaces. At a minimum, all requirements have the following attributes: Text, Tag, Key. (See Note 2 below.)

Note 1: Use the Requirement property of a TestStudio script to associate the script with a requirement. In this case, use the special property *object* as in the following example, where the source is a requirement and the target is the script.

source.object to target.Requirement

Note 2: For these attributes, *Tag* and *Key* are read-only; they cannot be set by a Synchronizer rule. You can set the *Text* attribute only if the requirement is database-based (not document-based).

Specify properties using the format: *source.<property>* or *target.<property>* where *source* and *target* correspond to the definitions in the *items* clause. To use quotes, enclose just the property in quotes, for example, *target."Planned Iteration"*.

The keyword *to* between the two properties means that on the first application of the rule, the receiving field, will be created and initialized with the source data. On subsequent synchronizations the data will be re-transferred if the values are different.

The equivalent keywords *init* or *initialize* cause the data transfer to occur once, when the target item is initially created.

`applies when AutoTest = "True"`

An optional *applies when* clause constrains the application of the rule. The clause performs an equality test on a single property in the source item. If the test results in a match, then the rule is applied to the source and target items. If the test result is false, the rule is not applied.

You can optionally create a traceability link from the source to the target or from the target to the source if allowed in the appropriate domains. Both the source and target items must be in the requirements domain. In this case, you can include one of these statements in the rule:

trace source to target
trace target to source

`end rule`

Every rule ends with an end clause.

Sample Rules Supplied with the Synchronizer

The Synchronizer includes sample rules which you can use as a starting point for your project. You may be able to use the rules as shipped or you can customize them to support your project's needs.

The sample rules are in the following directory:
`<Install Path>\Common\OutOfTheBox\synchronizer rules\.`

These rules work with the out-of-the-box RequisitePro domain project. If you use a different RequisitePro project schema you will probably have to modify the sample rules or write your own.

Table 4, Sample Rules, describes the rules supplied with the Rational Synchronizer.

Table 4: Sample Rules

File Name	Purpose
`actor_requirement_to_actor.rsd` Create a Rational Rose actor from an actor requirement	Create a Rational Rose actor model element that corresponds to a RequisitePro actor requirement. The name and documentation in the Rational Rose element are initialized from the requirement. After initialization, the name and documentation in the Rational Rose element are the master fields and are synchronized to the requirement on subsequent synchronizations.
`actor_to_actor_requirement.rsd` Create an actor requirement from a Rational Rose actor element	Create an actor requirement in RequisitePro that corresponds to a Rational Rose actor element.
`class_to_tr_requirement.rsd` Create Test Requirement from Rose Class	For each class model element in the modeling (Rose) domain, create a test requirement object in the requirements (RequisitePro) domain.
`supl_requirement_to_tr_requirement.rsd` Create Test requirement from Supplemental requirement	For a supplemental requirement, create a test requirement for use by TestStudio. This rule applies when the Test attribute on the SUPL requirement is True.
`tr_requirement_to_test_script.rsd` Create Test Script from Test requirement	For each test requirement in the requirements (RequisitePro) domain that has the attribute AutoTest set to True, create a test script object in the test (Rational Test) domain.
`uc_requirement_to_tr_requirement.rsd` Create Test requirement from Use Case requirement	For a use case requirement, create a test requirement for use by TestStudio. This rule applies when the Test attribute on the UC requirement is True.

The Synchronizer Rule Template File

A Synchronizer rule template file provides a starting point for writing rules. The template describes rule syntax and gives guidelines for writing rules.

The Synchronizer Rule Template File is located in `<Install Path>\Common\OutOfTheBox\synchronizer rules\synchronizer_rule_template.rsd`.

Synchronizer Rule Grammar

For completeness, Table 5, Synchronizer Rule Syntax, presents the Synchronizer rules grammar. **Bold** text designates terminal symbols; regular text designates non-terminal symbols.

Table 5: Synchronizer Rule Syntax

Left Side	Right Side
RulesFile:	Specifications
Specifications:	Specification Specification Specifications
Specification:	ProjectSpecification RuleSpecification
ProjectSpecification:	Empty ProjectBegin ProjectClauses ProjectEnd
ProjectBegin:	**begin project** ProjectName
ProjectClauses:	ProjectClause ProjectClauses
ProjectClause:	Empty ApplicationSpec PathSpec
ApplicationSpec:	**application** AppName
PathSpec:	**path** PathName
ProjectEnd:	**end project**
ProjectName:	String
AppName:	String
PathName:	String

Table 5: Synchronizer Rule Syntax (Continued)

Left Side	Right Side
RuleSpecification:	Empty RuleBegin RuleClauses RuleEnd
RuleClauses:	RuleClause RuleClauses
RuleClause:	Empty RuleDesc ProjectSpec ItemSpec PropertySpec Applies Trace
Empty:	
RuleBegin:	**begin rule** RuleName
RuleName:	String
RuleDesc:	Empty DescKeyword {Any-text-except-quote}
DescKeyword:	‘desc’ ‘descr’ ‘description’
ProjectSpec:	**projects** ProjectName ProjectName **projects** ProjectName **to** ProjectName
ProjectName:	String
ItemSpec:	**items** ItemType ItemType **items** ItemType **to** ItemType
PropertySpec:	**properties** PropertyTransferList
PropertyTransferList:	PropertyTransfer PropertyTransfer PropertyTransferList
PropertyTransfer:	Property TransferMode Property
Property:	PropertyLocation ‘**.**‘ String
PropertyLocation:	source target

Table 5: Synchronizer Rule Syntax (Continued)

Left Side	Right Side
TransferMode:	init initialize to
Applies:	**applies** SourcePropertyName '**=**' PropertyValue **applies when** SourcePropertyName '**=**' PropertyValue
Trace:	trace source **to** target trace target **to** source
SourcePropertyName:	String
PropertyValue:	String
String:	Identifier "Any-text-except-quote"
Trace:	'trace source to target' 'trace target to source'
RuleEnd:	**end rule**

6 Rational Suite Out-of-the-Box Tools

Rational Suite provides sample projects to help you get started with Rational RequisitePro, Rational Rose, Extended Help, and the Rational Synchronizer.

When you install Rational Suite, the installer places the sample projects in the directory `<Install Path>\Common\OutOfTheBox` where *Install_Path* is by default `c:\Program Files\Rational`.

The `OutOfTheBox` directory contains the sample projects described in Table 6, Rational Suite Sample Projects.

Table 6: Rational Suite Sample Projects

Directory	Contents	For More Information
customer database	Blank Extended Help database	See “Extended Help” on page 58
requirements	Sample RequisitePro project	See “RequisitePro Out-of-the-Box Project” on page 56
rose frameworks	Sample Rational Rose model	See “The Rational Rose RUP Model” on page 57
synchronizer rules	Synchronizer rules	See “Rational Synchronizer” on page 39

The remainder of this chapter describes these items in more detail.

RequisitePro Out-of-the-Box Project

The RequisitePro out-of-the-box project provides a complete RequisitePro project structure that uses the Rational Unified Process. You can use this project as the starting point for new RequisitePro projects.

The project name is *RequisitePro Project Template* and it consists of these files in the `requirements` directory: `rstemplate.mdb and rstemplate.rqs`.

Creating a Project Based on an Existing Project

You may want to base your next RequisitePro project on the out-of-the-box project provided by Rational. The project template provides requirement types, suggested attributes and attribute values (including default values), and it defines related document types. You can use the template as provided, or modify it to meet your project's needs.

The following table describes the structure of the project:

Table 7: RequisitePro Out-of-the-Box Project

Requirement Type	Document Type	Default Template
ACTOR	none	none
FEAT	Vision Document	rup_vision.dot
RMP	Requirements Management Plan	rup_rmpln.dot
STRQ	Stakeholder Requests	rup_stkreq.dot
SUPL	Supplementary Specification	rup_sspec.dot
TC	Test Case Specification	fcnltst.dot
TERM	Glossary	rup_gloss.dot
TPL	Test Plan	rup_tstpln.dot
TR	Test Requirement Type	none

To start a new project using the out-of-the-box project as a model, follow these steps:

1. From the RequisitePro toolbar, click **Project > New**.
2. In the New Project dialog box, select **Use an existing project as a template**.
3. In the cataloged projects list, select **RequisitePro Project Template**. Click **OK**.
4. Follow the normal process for creating projects as described in the RequisitePro documentation.

The Rational Rose RUP Model

The `rose frameworks` directory contains a Rational Rose model, called `rup framework.mdl`. To use this model, copy and rename the model file as follows:

1. Locate the model in the installed directory, `<Install Path>\Common\OutOfTheBox\rose frameworks`.
2. Copy the model into one of your project directories.
3. Rename the file as appropriate.

The model is designed to be self-explanatory. Each diagram and model element is documented in the documentation window, attached notes, or both. The structure of the model is designed to facilitate use of the Rational Unified Process.

Tool Mentors

Tool Mentors provide a link between the Rational Unified Process and Rational tools. A Tool Mentor tells you how to perform a Unified Process activity using a Rational tool. Tool Mentors are part of the Rational Unified Process and are available from both the Unified Process and from Extended Help.

Extended Help

Extended Help provides access from Rational tools to parts of the Rational Unified Process for various software development tasks, depending on your current context. For example, in the Extended Help sections for Rational Robot (included in TestStudio), you can review concepts about the testing lifecycle, as described in the Rational Unified Process.

Viewing Extended Help

Extended Help is available from all Rational Suite tools. From a Rational Suite tool menu, click **Help > Extended Help**.

Customizing Extended Help

You can add your own process descriptions and supporting information to the Extended Help database.

To customize Extended Help, perform these steps, as described in the following sections:

1. Populate the database with information.
2. Register the database and any string substitutions.

Populating the Database

Extended Help information is presented in HTML pages. Pointers to the pages are stored in a Microsoft Access database. You need Microsoft Access to populate the database. You do not need Microsoft Access to view the contents of the Extended Help database. To populate the database:

1. Locate the file `<Install Path>\Common\OutOfTheBox\customer database\custinfo.mdb`.

   Copy and rename `custinfo.mdb` into your project. Make the project directory accessible to all Rational Suite users.

2. Open Microsoft Access. From the Access menu, choose **File > Open** and select the project database you created.

3 On the **Forms** tab, double-click the **Help information entry**. Use this form to add, modify, or delete records from the database. Microsoft Access documentation explains how to perform these tasks.

To cause Extended Help to use a record, provide the following information in each **Help information entry** form:

- **product** – The name of the Rational product from which a query is invoked. The possible values are specified in Table 8, Valid Product Subtool Combinations, on page 60.
- **subtool** – A specific context within the product. The known values are shown in Table 8, Valid Product Subtool Combinations, on page 60.
- **query** – A category for the information pointed to by this entry. There is a set of known values, but you can use any value. Use the name of a folder that groups similar targets in the Extended Help browser. The maximum length of this field is 100 characters, but we recommend that you use shorter values.

  An example of a query value is `Requirements Traceability`. Targets for this query identify specific requirements traceability topics for the customer organization.
- **target** – The URL for the information supplied by this record for the specific product, subtool, and query. It can either be an absolute value or have string substitutes embedded in it that will be replaced at run time. For example, a valid target URL is `<corpinfo>/requirements/traceability.html`. The maximum length of this field is 255 characters.

  Use the EHRegister program to register the *corpinfo* string substitute. For more information, see *Registering the Database and String Substitutions* on page 61.
- **short_description** – The description that appears for the record under the appropriate folder in the Extended Help browser. The maximum length of this field is 50 characters.
- **description** – A longer description that appears in the item's Tool Tip when the cursor is held over the item. The maximum length of this field is 255 characters.

Table 8, Valid Product Subtool Combinations, shows the known values and legal combinations for the product and subtool fields.

Extended Help requires each record to contain a valid combination.

Table 8: Valid Product Subtool Combinations

Product	Subtool
ClearQuest	client
ClearQuest	designer
Purify	Any
RationalTest	Administrator
RationalTest	LoadTest
RationalTest	LogViewer
RationalTest	Robot
RationalTest	TestFactory
RationalTest	TestManager
ReqPro	Toolbar
ReqPro	View
Rose	Activity Diagram
Rose	Class Diagram
Rose	Collaboration Diagram
Rose	Deployment Diagram
Rose	Other Tool
Rose	Package Diagram
Rose	Sequence Diagram
Rose	State Diagram
Rose	Use Case Diagram
VisualPureCoverage	Any
VisualQuantify	Any

Note You can use the value **Any** for the subtool or product, indicating that any context within the product matches the record.

Registering the Database and String Substitutions

This section describes how to register your custom database and string substitutions.

Locating the EHRegister Program

The EHRegister program helps you register your Extended Help information. The EHRegister program is provided in the [`extras`] directory on the *Rational Solutions for Windows* CD.

You can run the EHRegister program directly from the *Rational Solutions for Windows* CD. You can also copy the program to your system as follows:

Copy files from the following locations to a directory on your system, where *drive* is the drive letter of your CD-ROM drive:

- `<drive>\extras\EHRegister.exe` – the program file
- `<drive>\extras\EHRegister.hlp` – the online help file
- `<drive>\extras\EHRegister.cnt` – the contents file for the program's online help

Using EHRegister

To register the custom database and string substations:

1. Click **Start > Run** and type `<Path>\EHRegister.exe`, where *Path* is the directory path of the `EHRegister.exe` program file.
2. Click **Help** to view the procedures for adding or updating Registry information.

7 Additional Resources

Rational Suite provides you with additional resources and assistance, including technical documentation and support information.

Online Release Notes

Release notes for Rational Suite are located in the `<Install Path>\Common\` directory.

Each product within Rational Suite provides its own set of release notes, included in the respective product directory under `<Install Path>\`.

Error Message

EHRegister

```
There was an error opening the registry key
'HKEY_LOCAL_MACHINE\SOFTWARE\Rational Software\Rational
Suite\1.0\Extended Help\databases.' This program is unable to
continue.
```

You do not have Rational Suite installed properly on your system. Review the installation of Rational Suite and retry EHRegister.

Index

C

customizing Extended Help 58

D

database relationships in Rational Suite 16
databases
  commercial 20
  Extended Help 58, 61
DBMS (Database Management System) 17

E

EHRegister 61
error messages 63
Extended Help 58
  customizing 58
  database 61
  requirements 58

I

importing rules 43

M

messages, error 63
Microsoft Access 58

O

One Rational Test database 41
Out-of-the-box
  creating a RequisitePro project 56
  Synchronizer rules 51
  tools 55
Overview of tasks performed by Rational Suite administrator 15

P

planning
  what to do before you plan 15
planning your database 17
projects
  repository 41
  rules 43

R

Rational Administrator
  using 21
Rational repository
  planning the 19
  setting up, high-level 22
  what is 19
Rational Rose RUP model 57
Rational Suite Administrator tasks 15
registering custom information
  EHRegister 61
repository
  adding one or more projects 26
  adding users, groups, and computers 26
  attaching existing Rose models 27
  creating the 25
  installing and configuring databases 24
  setting up, high-level 22
  upgrading ClearQuest data 23
  upgrading existing data 22
  upgrading Rational Rose data 24
  upgrading Rational test data 24
  upgrading RequisitePro data 23
  using Rational Suite without one 27
repository project 41
RequisitePro
  out-of-the-box project 56
resources
  online documentation 63
  online release notes 63
  support information 63

rules
  importing 43
  sample 51
  Synchronizer 43
    template file 52
  writing 44
rules project 43
running the Synchronizer 41

S

Synchronizer
  administration of 42
  out-of-the-box rules 51
  overview of using 39
  Rule template file 52
  rule template file 52
  rules 43
  rules files 51
  rules grammar 52
  running 41
  starting 41
  writing rules 44

T

technical support 63
template
  Synchronizer rule file 52
Tool Mentors 57

U

using Rational Suite without a
    repository 27