

# Installation Guide

RATIONAL ROSE® REALTIME

VERSION: 2003.06.00

PART NUMBER: 800-026105-000

WINDOWS/UNIX



## **Legal Notices**

©1993-2003, Rational Software Corporation. All rights reserved.

Part Number: 800-026105-000

Version Number: 2003.06.00

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

Rational, Rational Software Corporation, the Rational logo, Rational Developer Network, AnalystStudio, ClearCase, ClearCase Attache, ClearCase MultiSite, ClearDDTS, ClearGuide, ClearQuest, ClearTrack, Connexis, e-Development Accelerators, DDTS, Object Testing, Object-Oriented Recording, ObjecTime, ObjecTime Design Logo, Objectory, PerformanceStudio, PureCoverage, PureDDTS, PureLink, Purify, Quantify, Rational Apex, Rational CRC, Rational Process Workbench, Rational Rose, Rational Suite, Rational Suite ContentStudio, Rational Summit, Rational Visual Test, Rational Unified Process, RUP, RequisitePro, ScriptAssure, SiteCheck, SiteLoad, SoDA, TestFactory, TestFoundation, TestStudio, TestMate, VADS, and XDE, among others, are trademarks or registered trademarks of Rational Software Corporation in the United States and/or in other countries. All other names are used for identification purposes only, and are trademarks or registered trademarks of their respective companies.

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

U.S. GOVERNMENT RIGHTS. All Rational software products provided to the U.S. Government are provided and licensed as commercial software, subject to the applicable license agreement. All such products provided to the U.S. Government pursuant to solicitations issued prior to December 1, 1995 are provided with "Restricted Rights" as provided for in FAR, 48 CFR 52.227-14 (JUNE 1987) or DFARS, 48 CFR 252.227-7013 (OCT 1988), as applicable.

WARRANTY DISCLAIMER. This document and its associated software may be used as stated in the underlying license agreement. Except as explicitly stated otherwise in such license agreement, and except to the extent prohibited or limited by law from jurisdiction to jurisdiction, Rational Software Corporation expressly disclaims all other warranties, express or implied, with respect to the media and software product and its documentation, including without limitation, the warranties of merchantability, non-infringement, title or fitness for a particular purpose or arising

from a course of dealing, usage or trade practice, and any warranty against interference with Licensee's quiet enjoyment of the product.

**Third Party Notices, Code, Licenses, and Acknowledgements**

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

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

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

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

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

BasicScript is a registered trademark of Summit Software, Inc.

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

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

# Contents

- Preface . . . . . xv**
- Audience . . . . . xv
- Other Resources . . . . . xvi
- Rational Rose RealTime Integrations With Other Rational Products . . . . . xvii
- Contacting Rational Customer Service . . . . . xviii
- 1 Introduction . . . . . 1**
- Welcome to Rational Rose RealTime . . . . . 1
- Release Notes . . . . . 1
- Installation Guide Updates . . . . . 2
- What's New? . . . . . 2
- Improved UML Support . . . . . 2
- New Features to Improve Usability . . . . . 2
- Improved Rose Compatibility . . . . . 3
- Build and Target Enhancements . . . . . 4
- Improved Code Generation . . . . . 5
- Enhanced Configuration Management Integration and Faster Loading . . . . . 5
- Installation and Platform Support . . . . . 6
- Overview of Rational Rose RealTime Capabilities . . . . . 6
- Rational Connexis . . . . . 7
- Run-Time Connectivity Viewing . . . . . 7
- Rational Quality Architect - RealTime Edition . . . . . 8
- Allow Sub-Capsule Instances to be Drivers . . . . . 8
- Verification Mode Changes to Allow more Control in Manual Mode . . . . . 8
- Allow Data Qualifier in Data Field of Send Message . . . . . 9
- Allow Drivers to Model Timing Services . . . . . 9
- Capsule Interface Generation for Instances on Sequence Diagram Without Role Specified . . . . . 9
- Java Language Support . . . . . 9
- How to Get Help . . . . . 9
- Contacting Rational Customer Service Using the Help Menu . . . . . 9
- Contacting Rational Customer Service by Email or Telephone . . . . . 9
- Evaluation and Ordering Information . . . . . 10
- Rational Web Site . . . . . 10

- Directory Contents . . . . .10
- Accessing the Online Help System . . . . .12
- 2 Referenced Configurations and Toolchain Requirements . . . . . 13**
  - Referenced Configurations . . . . .13
    - Requirements for Windows NT . . . . . 13
    - Requirements for Windows 2000 . . . . . 14
    - Requirements for Windows XP Pro . . . . . 14
    - Requirements for UNIX. . . . . 15
  - Toolchain Requirements . . . . .15
    - Help Viewer (Windows Only) . . . . . 15
    - Compiler . . . . . 16
    - Real-time Operating System . . . . . 16
  - Referenced Host Configurations . . . . .16
    - Creating Executables for Hosts Without Toolset Support . . . . . 18
      - Generating an Executable Without a Common File System . . . . .19
  - Adding a Printer on UNIX. . . . .20
- 3 Installing Rational Rose RealTime on Windows . . . . . 23**
  - Removing Previous Releases of Rational Products. . . . .23
  - Installing Mixed Versions of Rational Products . . . . .24
  - Using the Rational Installation Program. . . . .24
  - Using the Rational Setup Wizard . . . . .25
    - Rational\_Install Log . . . . . 26
    - Before You Start the Rational Setup Wizard. . . . . 26
  - Before You Install . . . . .27
    - Preparing for a Rational Rose RealTime Installation . . . . . 28
  - Upgrade Information . . . . .28
  - Specifying the Rational License Server . . . . .29
  - Performing a Client Installation . . . . .29
  - Post-Installation Tasks . . . . .32
    - Licensing. . . . . 32
    - Canceling an Installation from CD-ROM. . . . . 32
    - Reinstalling Rational Rose RealTime from CD-ROM (Modify, Repair, Remove) 32

Creating a Release Area .....	33
Using the Rational Setup Wizard to Create a Release Area .....	34
Installing Rational Rose RealTime on Your Computer .....	35
Post-Installation Tasks .....	37
Licensing .....	37
Running the Site Preparation Wizard to Create Multiple Sitedef Files .....	37
Installing Rational Rose RealTime from a Release Area .....	38
Using a Standard Configuration .....	39
Customizing Your Own Configuration .....	40
Post-Installation Tasks .....	41
Licensing .....	41
Canceling a Product Installation From a Release Area .....	42
Reinstalling Rational Rose RealTime From a Release Area (Modify, Repair, Remove) .....	42
Using Silent Installation Commands .....	43
Silent Installation Overview .....	43
Running a Silent Installation on your Desktop .....	44
Licensing Your Rational Product .....	44
Setting Up Silent Installations of Rational Rose RealTime from a Release Area 45	
Running a Silent Installation .....	45
Canceling a Silent Installation .....	45
Command Line Syntax to Run Silent Install .....	46
Syntax .....	46
After You Install .....	46
Updating Batch Files .....	46
Configuring Your Environment .....	47
ClearCase Workstation Setup .....	47
Command Line Access to the Source Control Tool .....	48
Element type setup: type manager .....	48
ClearCase Options .....	48
Configuring the ClearCase Repository .....	48
Testing your Environment .....	49

<b>4</b>	<b>Installing Rational Rose RealTime on UNIX. . . . .</b>	<b>51</b>
	Before You Install . . . . .	51
	Installing in Secure Environments . . . . .	52
	Installing Multiple OS Versions of Rational Suite DevelopmentStudio RealTime (UnIX) . . . . .	52
	Stopping and Restarting an Installation . . . . .	52
	Upgrade Information . . . . .	52
	Upgrading to New Version Only (Uninstalling Earlier Version) . . . . .	53
	Upgrading to 2003.06.00 While Maintaining an Earlier Version . . . . .	54
	Installation Instructions . . . . .	54
	After You Install . . . . .	58
	Sourcing to the Setup Script. . . . .	58
	Unmounting the CD-ROM Drive . . . . .	58
	ClearCase Workstation Setup . . . . .	58
	Command Line Access to the Source Control Tool. . . . .	59
	Element type setup: type manager . . . . .	59
	ClearCase Options . . . . .	59
	ClearCase Repository Setup . . . . .	60
	Setting the TORNADO 2.0 Debugger Environment Variable . . . . .	60
	Setting Other TORNADO Environment Variables. . . . .	60
	Setting the Connexis Variable. . . . .	61
	Verifying the Connexis Installation . . . . .	61
	Verifying your Installation using BasicTest . . . . .	61
	Host Configuration Installation Verification . . . . .	61
	BasicTest Server Output . . . . .	63
	BasicTest Client Output. . . . .	64
	Starting Rational Rose RealTime (UNIX). . . . .	66
<b>5</b>	<b>Converting Connexis Models . . . . .</b>	<b>67</b>
	Converting Connexis version 2000.02.10 Models to Connexis Version 2003.06.00 Models. . . . .	67
	Verifying Component Compatibility . . . . .	70
<b>6</b>	<b>Understanding Rational Rose RealTime Licenses . . . . .</b>	<b>73</b>
	How Licenses Work . . . . .	73
	Types of Licenses . . . . .	74
	Node-Locked Licenses . . . . .	74
	Floating Licenses . . . . .	74
	Permanent Licenses and Temporary License Keys . . . . .	75



Emergency and Evaluation Keys . . . . .	75
Suite Licenses and Point Product Licenses . . . . .	75
Returning License Keys . . . . .	75
Upgrading Licenses . . . . .	76
Requesting License Keys . . . . .	76
Receiving and Importing License Keys . . . . .	77
Requesting License Keys by Fax . . . . .	77
Receiving Permanent License Keys . . . . .	78
Converting a Temporary License to a Permanent License . . . . .	78
Licenses for Windows . . . . .	79
The License Manager - UNIX . . . . .	79
License Manager Commands . . . . .	80
Additional Licensing Commands . . . . .	81
License Manager Daemon (lmgrd) . . . . .	81
Vendor Daemon . . . . .	81
License Key File . . . . .	82
Application Program . . . . .	82
Configuring a UNIX Workstation to Point to a FLEXlm Server . . . . .	82
License Activation Process . . . . .	83
Licensing on UNIX . . . . .	84
Running the lmgrd from a Command Prompt . . . . .	84
Example . . . . .	84
Administration Commands . . . . .	85
The License File . . . . .	85
Format . . . . .	85
UNIX Licenses . . . . .	87
Start-up or Emergency keys . . . . .	87
Node-Locked keys . . . . .	87
Floating keys . . . . .	88
TLA (Temporary License Agreement) . . . . .	88
Frequently Asked Questions . . . . .	88
<b>7 Installing License Keys . . . . .</b>	<b>89</b>
Before You Begin . . . . .	89
Installing a Startup or Permanent License on Windows . . . . .	89
Installing a Permanent License on Windows . . . . .	90
Installing the License Key . . . . .	92
Installing a Floating License Key on a UNIX server . . . . .	92

Installing a Startup or Permanent License on UNIX .....	92
Installing a Startup License on UNIX .....	92
Installing a Permanent License on UNIX .....	93
Installing the License Key .....	95
Integration With Rational Suites Licensing .....	95
Troubleshooting .....	96
Windows .....	96
UNIX server .....	97
UNIX .....	97
<b>8 Migration .....</b>	<b>99</b>
Migrating from Rational Rose .....	99
User Interface Differences .....	99
New Modeling Language Elements .....	101
Code Generation, Building, and Running .....	101
Opening Models from Rational Rose .....	102
List of Importation Log Messages .....	102
Limitations and Restrictions .....	103
Importing Rational Rose Generated Code .....	104
Limitations and Restrictions .....	104
Migrating from ObjecTime Developer 5.2/5.2.1 .....	105
Terminology .....	105
User Interface Differences .....	107
Compilation .....	107
Migrating from Rational Rose RealTime 6.0/6.0.1/6.0.2/6.1 .....	108
File Format Changes .....	108
Source Control Migration .....	108
Migrating Customized CM Scripts .....	109
Language Add-in Changes .....	110
Running Two Different Releases of Rational Rose RealTime .....	110
Workspace Files .....	110
RRTEI Changes .....	110
C Language Migration .....	112
Converting a C++ Model to C .....	112
ObjecTime Developer for C Migration .....	113
Importing Models .....	113
Converting Global Signals to Local Signals .....	114
Timing Service .....	115

C++ Language Migration .....	115
Backwards Compatibility Mode .....	115
Migrating in Two Steps .....	116
What Does Backwards Compatibility Do? .....	116
Compiler Will Find All Errors .....	116
Building a Model in Backwards Compatibility Mode .....	117
Full Migration .....	119
Changes .....	119
C++ UML Services Library .....	119
Code Generation and Compilation .....	120
New Classes for Protocols, Signals, and Ports .....	120
Type Safety Explained .....	120
How Has This Changed? .....	121
API Changes Summary .....	121
Asynchronous Sends .....	122
Synchronous Sends .....	123
Message Reply .....	123
Defer, Recall, and Purge .....	124
Port Indexes .....	125
Discriminating in Code the Signal of a Received Message .....	126
Forwarding .....	126
RTPortRef Operations .....	128
RTTimespec Parameters .....	130
RTSignalNames .....	130
Macros .....	130
External Layer Service (ELS) .....	131
Code Generation .....	131
Components .....	131
Directory Structure .....	132
Parameters Available in Transition Code .....	132
Port Cardinality Cannot be Unspecified .....	133
Makefile Override Changes .....	133
Model Properties .....	133
Advanced property Editors .....	133
<b>9 Integration Notes .....</b>	<b>135</b>
Overview .....	135
Configuration Management (CM) Tools Integration .....	135
ClearCase on a UNIX Server and Clients on both NT and UNIX .....	136
Migrating from Rational Rose and ObjecTime Developer .....	136
Requirements Management Tools Integration .....	137
Rational SoDA for Word .....	137
Rational RequisitePro .....	137

Unit Testing Tools Integration .....	137
Rational Purify .....	138
Adding Options to Purify on UNIX.....	138
Microsoft Development Environment .....	138
Integration with Rational Robot .....	138
Naming Directories .....	139
<b>10 Starting Rational Rose RealTime .....</b>	<b>141</b>
Starting Rational Rose RealTime on Windows .....	141
Starting Rational Rose RealTime on UNIX .....	141
Start-up Options for UNIX.....	142
Rational Rose RealTime for UNIX and the X Window System .....	142
X Clients .....	143
X Servers .....	143
X Window Managers.....	143
Input Focus (Active Window) Policy .....	144
Window Order Policy.....	144
Automating Rational Rose RealTime.....	144
Command Line Options .....	145
<b>11 Add-Ins .....</b>	<b>147</b>
Web Publisher .....	147
Suggested Workflow.....	147
Limitations.....	148
Model Integrator .....	149
Suggested Workflow.....	149
Rose C++ Analyzer .....	150
Suggested Workflow.....	150
Limitations.....	152
<b>12 Uninstalling Rational Rose RealTime.....</b>	<b>153</b>
Windows .....	153
UNIX .....	153

<b>13 Troubleshooting, Known Issues, and Updates</b>	<b>155</b>
Overview	155
Rational Connexis	155
Troubleshooting	156
Transport Integration Framework	156
Turning Off Auditing for a Single Transport is Not Recommended.	156
Signals No Longer Supported	156
Rational Quality Architect - RealTime Edition	157
Target Observability Behavior When the Model is Modified.	157
Running Verify Behavior with Eighty or More Sequence Diagrams (UNIX)	157
Driver Methods for Sending Messages to the Log and Custom Comparison	158
Lost Information in <i>To Port</i> for a Message	158
Do Not Use -runScriptAndQuit When Running RQART From a Script	158
Creation of Container Capsules	159
Converting MSCs in Rational Rose RealTime Using the RQA-RT	159
Creating Messages and Sequence Diagrams	160
Sending Message Specification Data Field Format for Java	160
Customizing a Sequence Diagram Created From a Trace.	160
RQA-RT Limitations	161
<b>14 Technical Support</b>	<b>163</b>
Submitting Problem Reports	163
Submitting Feature Requests.	164
Submitting Support Requests	165
Contacting Rational Customer Service by Email or Telephone	166
License Support Contact Information.	167
<b>Index</b>	<b>169</b>



# Preface

This manual provides the necessary information to install, uninstall, and configure Rational Rose RealTime for your environment.

This manual is organized as follows:

- *Introduction* on page 1
- *Referenced Configurations and Toolchain Requirements* on page 13
- *Installing Rational Rose RealTime on Windows* on page 23
- *Installing Rational Rose RealTime on UNIX* on page 51
- *Converting Connexis Models* on page 67
- *Understanding Rational Rose RealTime Licenses* on page 73
- *Installing License Keys* on page 89
- *Migration* on page 99
- *Integration Notes* on page 135
- *Starting Rational Rose RealTime* on page 141
- *Add-Ins* on page 147
- *Uninstalling Rational Rose RealTime* on page 153
- *Troubleshooting, Known Issues, and Updates* on page 155
- *Technical Support* on page 163

## Audience

---

This guide is intended for all readers, including managers, project leaders, analysts, developers, and testers.

## Other Resources

---

- Online Help is available for Rational Rose RealTime.

Select an option from the **Help** menu.

All manuals are available online, either in HTML or PDF format. To access the online manuals on Windows, click **Rose RealTime Online Documentation** from the **Start** menu.

- To send feedback about documentation for Rational products, please send e-mail to [techpubs@rational.com](mailto:techpubs@rational.com).
- For more information about Rational Software technical publications, see: <http://www.rational.com/documentation>.
- For more information on training opportunities, see the Rational University Web site: <http://www.rational.com/university>.
- For articles, discussion forums, and Web-based training courses on developing software with Rational Suite products, join the Rational Developer Network by selecting **Start > Programs > Rational Suite > Logon to the Rational Developer Network**.



## Rational Rose RealTime Integrations With Other Rational Products

---

Integration	Description	Where it is Documented
Rose RealTime–ClearCase	You can archive Rose RealTime components in ClearCase.	<ul style="list-style-type: none"> <li>▪ <i>Toolset Guide: Rational Rose RealTime</i></li> <li>▪ <i>Guide to Team Development: Rational Rose RealTime</i></li> </ul>
Rose RealTime–UCM	Rose RealTime developers can create baselines of Rose RealTime projects in UCM and create Rose RealTime projects from baselines.	<ul style="list-style-type: none"> <li>▪ <i>Toolset Guide: Rational Rose RealTime</i></li> <li>▪ <i>Guide to Team Development: Rational Rose RealTime</i></li> </ul>
Rose RealTime–Purify	When linking or running a Rose RealTime model with Purify installed on the system, developers can invoke the Purify executable using the <b>Build &gt; Run with Purify</b> command. While the model executes and when it completes, the integration displays a report in a Purify Tab in Rose RealTime.	<ul style="list-style-type: none"> <li>▪ Rational Rose RealTime Help</li> <li>▪ <i>Toolset Guide: Rational Rose RealTime</i></li> <li>▪ <i>Installation Guide: Rational Rose RealTime</i></li> </ul>
Rose RealTime–RequisitePro	You can associate RequisitePro requirements and documents with Rose RealTime elements.	<ul style="list-style-type: none"> <li>▪ <i>Addins, Tools, and Wizards Reference: Rational Rose RealTime</i></li> <li>▪ <i>Using RequisitePro</i></li> <li>▪ <i>Installation Guide: Rational Rose RealTime</i></li> </ul>
Rose RealTime–SoDa	You can create reports that extract information from a Rose RealTime model.	<ul style="list-style-type: none"> <li>▪ <i>Installation Guide: Rational Rose RealTime</i></li> <li>▪ <i>Rational SoDA User's Guide</i></li> <li>▪ SoDA Help</li> </ul>

## Contacting Rational Customer Service

---

If you have questions about installing, using, or maintaining this product, contact Rational Customer Service.

Your Location	Telephone	Facsimile	E-mail
North, Central, and South America	+1 (800) 433-5444 (toll free) +1 (408) 863-4000 Cupertino, CA	+1 (781) 676-2460 Lexington, MA	support@rational.com
Europe, Middle East, Africa	+31 20 4546-200 Netherlands	+31 20 4546-201 Netherlands	support@europe.rational.com
Asia Pacific	+61-2-9419-0111 Australia	+61-2-9419-0123 Australia	support@apac.rational.com

**Note:** When you contact Rational Customer Service, please be prepared to supply the following information:

- Your name, company name, telephone number, and e-mail address
- Your operating system, version number, and any service packs or patches you have applied
- Product name and release number
- Your Service Request number (SR#) if you are following up on a previously reported problem

When sending email concerning a previously-reported problem, please include in the subject field: "[SR#XXXXX]", where XXXXX is the Service Request number of the issue. For example, "[SR#0176528] - New data on Rational Rose RealTime install issue".

## Contents

This chapter is organized as follows:

- *Welcome to Rational Rose RealTime* on page 1
- *Overview of Rational Rose RealTime Capabilities* on page 6
- *What's New?* on page 2
- *How to Get Help* on page 9
- *Directory Contents* on page 10
- *Accessing the Online Help System* on page 12

## Welcome to Rational Rose RealTime

---

Rational Rose RealTime is a comprehensive visual development environment that delivers a powerful combination of notation, processes, and tools to meet the challenges of real-time software development. Through the industry-standard Unified Modeling Language (UML), real-time design constructs, code generation, and model execution capabilities, Rational Rose RealTime addresses the complete lifecycle of a project: from early use case analysis, through to design, implementation, and testing.

Rational Rose RealTime is designed for simple insertion into your software development environment, processes, and workflows. Rational Rose RealTime includes seamless integration with other Rational products and support for a variety of commercial real-time operating systems.

This guide provides the necessary information to install and configure Rational Rose RealTime in your environment.

## Release Notes

See the *Rational Rose RealTime Release Notes* for information on system requirements, known limitations, documentation updates, and troubleshooting information.

## Installation Guide Updates

For the latest documentation updates, please refer to the Rational Rose RealTime web site:

<http://www.rational.com/support/>

Navigate to the **Product Documentation** link.

## What's New?

---

Welcome to Rational Rose RealTime Version 2003.06.00. Based on extensive customer consultation and feedback, this release contains many updates and corrections designed to streamline user workflows and enhance developer productivity. Listed below are some of the more visible changes that you will discover in this release. We hope that you find the following enhancements helpful and we look forward to serving your needs in future releases:

### Improved UML Support

Rose RealTime now supports UML activity diagrams. Users can create and view activity diagrams within Rose RealTime.

### New Features to Improve Usability

Many new features have been added to this release that simplify user workflows associated with editing and navigating models and code. These include:

- Improvements to the find engine to provide easier access and find more things in more places. The engine also now helps find model elements in the navigator and provides better searches of generalizations, messages and sequence diagrams, diagram notes, and connectors.
- Improvements to model navigation and the ability to modify the design from model element specifications without having to use diagrams. This has been extended to:
  - Modifying class relations and relation ends
  - Adding ports, connectors, and triggers, and other elements

You can now navigate models from package specifications with a new tab that lists the contained elements from the package. You can select all references on a component specification using **Ctrl-A**.

- Improvements to state diagram editing and manipulation with better support for copy-paste, improved diagram layout and label positioning, and the addition of a tab listing transitions and their trigger conditions on state specifications.
- Usability improvements for associations. Association ends are now visible in the model navigator as well as in a compartment on classes in class diagrams.
- Usability improvements to aggregations. It is now easier to modify association ends using the aggregation tool as well as from the relations tab on class specifications.
- Improved sequence diagram layout, editing and navigation with better support for copy/paste, additional choices for layout, the ability to navigate sequence diagrams using the keyboard, the ability to create sequence diagrams from the capsule structure navigator, and the ability to add create messages to existing interaction instances.
- Improved support for user defined stereotypes and increased support of built in stereotypes and specialized icons.
- Many other general improvements such as more consistent menus, additional keyboard shortcuts, more interactive diagram layout, and the ability to print user code when printing a model.
- Improved requirements modeling with the addition of the **Show participants in Use Case** report, the ability to create relationships between use cases, and support for finer granularity requirements traceability with Rational Requisite Pro.

## Improved Rose Compatibility

Rational Rose model import has been improved including support for importing C++ code associated with Classic C++ Rose models. Rose model artifacts are preserved and displayed in Rose RealTime, including state, activity, sequence, collaboration, class, component, deployment, and sequence diagrams. Because of differences in the diagram formatting between the two tools, some diagrams may appear differently in Rose RealTime than in Rose, but the information is preserved. Rose RealTime now imports models from Rose v2002 without having to convert to Rose98 format.

For users migrating their Rose models and code to Rose RealTime, improvements have been made to the sharing and interworking between code and model artifacts from Rose C++ within Rose RealTime. You can reference Rose C++ code from Rose RealTime during build, or you can import the model and C++ code into Rose RealTime for further development.

## Build and Target Enhancements

Many new features that enhance target support and code centric workflows have been added:

- It is now easier to edit the generated code with the addition of **Browse Header** and **Browse Body** menus to navigate to the generated source code from the model.
- The generated source code is now easier to read with the ability to turn off the generation of codesync and model tags in the generated code.
- Rose RealTime can include model documentation in the generated C and C++ code, and can code-sync documentation changes back into the model.

Integration with source debuggers has been enhanced with support for adding source breakpoints to operations:

- Source debugging integration without UML debugging is now supported. This permits source debugging on targets that do not have TCP/IP support required for UML debugging.
- It is now possible to create and delete source breakpoint to operations in the UML model and have those appear in an external source debugger.
- Source breakpoints are easier to manage with the addition of a source breakpoint viewer to view model-level source breakpoints.
- The watch window has been enhanced to show the class of a variable, and to better display long class names and data values.
- Improvements to better indicate where a model is halted on a model halt.

Target platform and IDE support, and integration with non-Rose RealTime applications has been enhanced:

- Wind River Tornado 2.2, as well as bare C and C++ environments without an operating system have been added to the list of reference configurations.
- An “External” service API has been added to simplify the integration from non-Rose RealTime threads and Rose RealTime generated capsule-based applications.

There have been numerous other improvements as well:

- The add class dependencies feature has been improved and now supports a finer granularity of choices.
- Report file saves progress is now reported during code generation.
- It is now possible to build a component directly from a component diagram.
- The TargetRTS wizard has been updated.
- Path maps are now available for library scripts.

## Improved Code Generation

Rose RealTime code generation has been extended to support a wider range of target applications. Rose RealTime now includes:

- Support for template code generation from C++ Parameterized classes
- Support for state-machine code generation for C and C++ classes
- Generation of standalone C and C++ libraries, and generation of standalone non-capsule based executables
- Support for C++ exception declarations
- Support for multi-line initial data and header/implementation preface/ending specifications for C and C++ attributes

## Enhanced Configuration Management Integration and Faster Loading

New background synchronization, faster model loading, and faster toolset shutdown improves support for large scale development have been added to this release of Rose RealTime. Model merging has been improved with enhancements such as addition of the ability to work with Rational ClearCase to support merging of code and text in UML models. In addition, sequence diagrams can now be controlled units letting you create simpler models with faster saves and improved model merging.

Model sharing and restructuring has been simplified with the ability to easily change units from shared to owned and back, the addition of a **Save As** command for controlled units to easily modify the storage location of controlled units, the ability to easily create controlled units in uncontrolled units, the ability to view the ClearCase version tree from within Rose RealTime, and support for ClearCase unreserved checkouts. The **Apply Label** command now works recursively by model element instead of file system.

## Installation and Platform Support

Rose RealTime now installs from one CD and does not require a separate install key.

Host platform support has been extended with support for Solaris Version 9.

## Overview of Rational Rose RealTime Capabilities

---

### Modeling:

- Use Case Modeling
- Class Modeling
- Collaboration (role) Modeling
- Interaction Modeling (sequence diagrams)
- Component Modeling
- Deployment Modeling

### Application Generation:

- C++ Language Support
- Java Language Support
- C Language Support
- Data Class Code Generation

### Visual Execution:

- Host Execution
- Target Execution
- Model Visualization (Animation)
- Model Debugging (Tracing, Injection, Inspection)

### Tools Interworking:

- Rational ClearCase
- Microsoft Visual SourceSafe (Windows only)
- SCCS (UNIX only)
- RCS (UNIX only)
- PVSC (UNIX only)
- Rational SoDA (requires Rational Rose RealTime domain)
- Rational RequisitePro
- Rational Purify



### **Model Documentation:**

- Report Generation (Windows only)
- Web Publisher

## **Rational Connexis**

Connexis simplifies the construction of reliable, distributed applications using Rational Rose RealTime. You can easily establish communication paths and send messages between capsules in separate processes whether they reside on the same node or on separate nodes. Also, you can monitor connections, trace messages, and collect communication metrics.

With Rational Connexis, you can distribute applications built with Rational Rose RealTime version 2003.06.00. Connexis also provides features to help you build reliable distributed applications.

Connexis extends the asynchronous messaging used between capsules in Rational Rose RealTime so that it can be used between capsules located in different processes or different nodes in a network. Connexis allows you to use unwired ports, based on protocols you define, to establish these connections. Since the same mechanisms are used for local and remote messaging, it is easy to make your application distributed.

Using a publish and subscribe pattern, Connexis can connect capsules whether they are on the same processor, distributed on a backplane, across a network connection, or through some other channel. To implement these connections, Connexis uses the underlying TCP/IP stack of the operating system.

Connexis also includes several features to provide fault-tolerance capabilities to the applications you build with Rational Rose RealTime. The Connexis Locator Service, which can be used to find connection destinations, supports the use of a backup locator to automatically take over should the primary locator fail.

## **Run-Time Connectivity Viewing**

Determining connectivity in a distributed system can be tedious and time-consuming. To make this easier, Connexis includes a connectivity trace tool, the Connexis Viewer, that lets you examine and monitor connection status in real-time. You can easily determine the services that have been published and who has subscribed to them, wherever the publishers and subscribers may be on your network.

The Connexis Viewer also lets you trace messages between publishers and subscribers. You can apply filters to restrict the trace information being captured and you can also specify the number and size of the memory buffers allocated for tracing. This allows you to carefully control the allocation and use of tracing resources on target.

Together, these features provide easy, convenient, and fault-tolerant connectivity. Instead of building your own distributed communications infrastructure, you can build additional revenue-generating features into your product. The result is that your product ships sooner with more features.

For additional information on Connexis, see the *User Guide - Rational Rose RealTime Connexis*.

## **Rational Quality Architect - RealTime Edition**

Rational Quality Architect automates scenario-based unit testing. You can create sequence diagrams that specify how any set of capsules interact. With a single command, Rational Quality Architect generates and builds test harnesses along with any requested stubs, runs the tests on the specified target, collects the results at run-time, compares those results with the original sequence diagrams, and highlights any differences between them along with any detected race conditions.

Using Rational Quality Architect - RealTime Edition (RQA-RT), you can extend Rational Rose RealTime's design automation capabilities to model, debug and test. By automatically generating complete unit and integration test harnesses directly from sequence diagram specifications, manual coding of stubs and drivers for debugging and testing is eliminated.

RQA-RT automatically verifies designs against sequence diagram specifications both analytically and during execution. Application generation and automatic testing of fully or partially complete designs, plus animated visual and symbolic debuggers, encourages early and continuous design refinement and validation.

For detailed information on RQA-RT, see the *User's Guide - Rational Quality Architect RealTime Edition*.

### **Allow Sub-Capsule Instances to be Drivers**

This functionality provides a way to implement "stub" generation for contained capsule instances. Providing this functionality RQA-RT enables more robust testing and more complex interactions.

### **Verification Mode Changes to Allow more Control in Manual Mode**

This functionality enables RQA-RT to operate in manual mode and provides the functionality to run the harness on a custom node.

## **Allow Data Qualifier in Data Field of Send Message**

This functionality enables full composite data types to be passed as the signal data, allowing the same range of data flexibility or inject capability as in the toolset.

## **Allow Drivers to Model Timing Services**

When a timing service is invoked from the driver interaction instance, it will be generated into the driver capsule. This provides more robust support for converting trace diagrams into specifications.

## **Capsule Interface Generation for Instances on Sequence Diagram Without Role Specified**

Required test wrappers are automatically generated based on the user supplied sequence diagrams.

## **Java Language Support**

Enables RQA-RT functionality on a model based on the Java Language add-in.

# **How to Get Help**

---

This section describes procedures for interacting with Rational Customer Service.

## **Contacting Rational Customer Service Using the Help Menu**

With Rational Rose RealTime, you can email problem reports, feature requests, or support requests to the Rational Customer Service department that services your location, directly from the Rational Rose RealTime application's Help menu.

For details on how to use this feature, see *Technical Support* on page 163.

## **Contacting Rational Customer Service by Email or Telephone**

When contacting Rational Customer Service by email or by telephone, please be prepared to supply the following information:

- Name, telephone number, and company name
- Product name and version number
- Operating system and version number (for example, Windows NT 4.0, Windows 2000, Windows XP Pro, Solaris 2.6, 2.7, 2.8, and 2.9)

- Computer make and model
- Your service request id (if you're calling about a previously reported problem)
- A summary description of the problem, related errors, and how it was made to occur

For details on contacting Rational Customer Service by email or telephone, see *Contacting Rational Customer Service by Email or Telephone* on page 166.

## Evaluation and Ordering Information

### United States and Canada

Rosebud@rational.com

1-800-728-1212

### Other Worldwide locations

Rosebud@rational.com

+1-408-863-9900

## Rational Web Site

You can contact Rational Customer Support and obtain the latest product information through our web site at:

<http://www.rational.com/support>

## Directory Contents

---

After installation of the main Rational Rose RealTime files for Windows and UNIX, ensure that the installation directory is \$ROSSERT\_HOME on UNIX (Solaris) and %ROSSERT\_HOME% on Windows (NT, 2000, and XP Pro) and all its associated files are readable, and not writable, by all users of Rational Rose RealTime.

**Note:** For UNIX, the \$ROSSERT\_HOME/Help directory must be Read/Write for all.

The ROSERT\_HOME directory and its sub-directories contain all the individual files that comprise this release of Rational Rose RealTime. Some of the files and directories are:

Directory	Description
ROSSERT_HOME	This is the top level directory.
AddIns	Contains the configuration information required by Rational Rose RealTime Add-ins.
bin	Contains the Rational Rose RealTime executable and various scripts. The bin directory also contains subdirectories for each of the supported workstation platforms ROSERT_HOST.
C++ or C	These directories contain the libraries, header files, scripts relating to code generation, and source files for the Services Library. For more information regarding the Services Library, see the <i>Toolset Guide</i> and the <i>C Reference</i> and <i>C++ Reference</i> .
Connexis	Contains the Connexis files.
Examples	Contains example model files.
Help	Contains the online Help, PDFs, and Viewlets.
Tutorials	Contains model files for different stages of the various tutorials. See the <i>Rational Rose RealTime Tutorial</i> for additional tutorial information.
RQART	Contains the Rational Quality Architect - RealTime Edition files.
RTJava	Contains the classes and scripts relating to code generation in Java, See the <i>Java Reference</i> for more information.
Scripts	Contains various Rational Rose RealTime scripts.
WebPublisher	Contains the Web Publisher files.

**Note:** For the latest integration information and referenced configurations, see the *Rational Rose RealTime Release Notes*.

## Accessing the Online Help System

---

Online Help and documentation for Rational Rose RealTime is provided in HTML Help format. To access the online Help, from the **Help** menu, click **Contents**.

PDF versions of the documentation are available in the \$ROSERT\_HOME/Help directory, unless specified otherwise.

### Windows Only

The **Help Viewer** requires that Microsoft Internet Explorer (version 3.02 or later) be configured on a user's computer. It is not required that Internet Explorer be used as the system's default browser, or that the Internet Explorer icon be visible on the user's desktop.

# Referenced Configurations and Toolchain Requirements

# 2

## Contents

This chapter is organized as follows:

- *Referenced Configurations* on page 13
- *Toolchain Requirements* on page 15
- *Referenced Host Configurations* on page 16
- *Adding a Printer on UNIX* on page 20

**Note:** Rational Rose RealTime is not supported on Windows 95, Windows 98, or Windows ME.

## Referenced Configurations

---

Before you install on Windows or UNIX, verify that your host configuration meets the minimum system requirements:

- *Requirements for Windows NT* on page 13
- *Requirements for Windows 2000* on page 14
- *Requirements for Windows XP Pro* on page 14
- *Requirements for UNIX* on page 15

## Requirements for Windows NT

The minimum supported configuration for running Rational Rose RealTime on Windows NT is:

- Windows NT 4.0, with service pack 6a and Security Rollup Package (SRP)
- Minimum Pentium II 150 MHz. We recommend 500 MHz or faster CPU
- Minimum 128 MB of RAM. We recommend 256 MB or more of RAM
- Minimum 325 MB of free disk space for the Rational Rose RealTime installation

- Minimum display 1024 X 768. We recommend 1280 X 1024 or higher
- Postscript printer for printing
- Browser requirement - Internet Explorer 5.5 or 6.0 or Netscape Navigator 4.72-4.78 or 7.0. We recommend Internet Explorer 5.5 or 6.0

**Note:** AccoutLink is not accessible using Netscape 4.x browsers.

For additional information regarding requirements for installing Rational Suite DevelopmentStudio, see the book *Installation Guide Rational Suite*.

## Requirements for Windows 2000

The minimum supported configuration for running Rational Rose RealTime on Windows 2000 is:

- Windows 2000 Professional, with service pack 2 or 3.
- Minimum Pentium II 150 MHz. We recommend 500 MHz or faster CPU
- Minimum 128 MB of RAM. We recommend 256 MB or more of RAM
- Minimum 325 MB of disk space for the Rational Rose RealTime installation
- Minimum display 1024 X 768. We recommend 1280 X 1024 or higher
- Postscript printer for printing
- Browser requirement - Internet Explorer 5.5 or 6.0 or Netscape Navigator 4.72-4.78 or 7.0. We recommend Internet Explorer 5.5 or 6.0

**Note:** AccoutLink is not accessible using Netscape 4.x browsers.

For additional information regarding requirements for installing Rational Suite DevelopmentStudio, see the book *Installation Guide Rational Suite*.

## Requirements for Windows XP Pro

The minimum supported configuration for running Rational Rose RealTime on Windows XP Pro is:

- Windows XP Pro, or XP Pro with service pack 1
- Minimum Pentium II 300 MHz. We recommend 500 MHz or faster CPU
- Minimum 128 MB of RAM. We recommend 256 MB or more of RAM
- Minimum 325 MB of free disk space for the Rational Rose RealTime installation
- Minimum display 1024 X 768. We recommend 1280 X 1024 or higher



- Postscript printer for printing
- Browser requirement - Internet Explorer 5.5 or 6.0 or Netscape Navigator 4.72-4.78 or 7.0. We recommend Internet Explorer 5.5 or 6.0

**Note:** AccountLink is not accessible using Netscape 4.x browsers.

For additional information regarding requirements for installing Rational Suite DevelopmentStudio, see the book *Installation Guide Rational Suite*.

## Requirements for UNIX

The minimum supported configuration for running Rational Rose RealTime on UNIX is:

- Solaris 2.6, Solaris 2.7, Solaris 2.8, or Solaris 2.9
  - For Solaris operation, the minimum workstation is an UltraSparc 10 with 500 MB of RAM. We recommend an UltraSparc 60 with 600 MB of RAM. We recommend the Solaris 2.8 operating system.
  - Please see the Rational Rose RealTime web site (<http://www.rational.com/support>) for a list of the required UNIX patches applicable to your operating system, or run the **check\_rose\_reqs** script in the \$ROSET\_HOME/bin folder.
- The minimum is 256 MB of RAM. We recommend 512 MB of RAM with approximate three times this amount of swap space.
- Minimum 370 MB of free disk space for the Rational Rose RealTime installation.
- Browser requirement - Netscape Navigator 7.0.

For additional information regarding requirements for installing Rational Suite DevelopmentStudio, see the book *Installing Rational Suite DevelopmentStudio*.

## Toolchain Requirements

---

### Help Viewer (Windows Only)

The Help Viewer requires that Microsoft Internet Explorer (version 5.5 or later) be configured on your computer. For details, see *Accessing the Online Help System* on page 12.

## Compiler

You must have a C or C++ compiler installed on your system if you are working in either C or C++, to make use of the code compilation and execution capabilities for Rational Rose RealTime. Different compilers are required for host workstation and for embedded system targets. For a list of supported compilers and targets, see *Referenced Host Configurations* on page 16.

You do not need a compiler if you are working in Java.

## Real-time Operating System

If you are planning to deploy your model on a real-time operating system, your operating system, hardware and tool line-up must be one of the supported lineups listed in Table 2 on page 17. If you do not have a supported line-up, you may be able to get support for your line-up from a Rational RoseLink partner, or by customizing the Rational Rose RealTime Services Library for your target. For instructions on customizing the Services Library and compiling for new target platforms, see the *C++ Reference*, *C Reference*, or *Java Reference*.

## Referenced Host Configurations

---

Table 1 shows the referenced host configurations for this release of Rational Rose RealTime.

**Table 1** Host Configurations

Toolset Host	Requirements
Solaris 2.6	See <i>Requirements for UNIX</i> on page 15
Solaris 2.7	See <i>Requirements for UNIX</i> on page 15
Solaris 2.8	See <i>Requirements for UNIX</i> on page 15
Solaris 2.9	See <i>Requirements for UNIX</i> on page 15
Windows NT 4.0 (Service Pack SP6a)	See <i>Requirements for Windows NT</i> on page 13
Windows 2000 (Service Packs SP2 and SP3)	See <i>Requirements for Windows 2000</i> on page 14
Windows XP Pro (Service Pack SP1)	See <i>Requirements for Windows XP Pro</i> on page 14

A pre-defined set of the Rational Rose RealTime UML Services Libraries are delivered as part of the Rational Rose RealTime product. The UML Services Library is what allows the execution of standalone executable models on target operating systems. These ports are fully tested by Rational, and are covered by standard Rational Support Agreement. A standard port can be used to facilitate a port to your environment of choice.

**Note:** For a more detailed description of the Services Library, refer to the programmer's guides, or online Help.

A port is based on the following specifications (often called the toolchain line-up):

- OS version
- Processor type
- Compiler version

If you use a configuration other than those tested by Rational and listed in this guide, standard support will cover problems encountered by customers only to the extent that the problem is reproducible for the configurations listed in this guide.

Table 2 shows the referenced configurations and targets.

**Table 2 Referenced Configurations and Targets**

Host Configuration(s)	Target RTOS	Compiler/Processor	RTS Library	Connexis DCS Library
Solaris	Same	Gnu 3.04 SPARC	C & C++	C++
		Sun C++ 5.3, SPARC	C++	C++
		Sun C 5.3, SPARC	C	-
Windows	Same	Visual C++ 6.0, x86	C & C++	C++
		Visual C++ 7.0, x86	C & C++	C++
Solaris, Windows	OSE 4.1.1	Diab 4.3f, ppc	C & C++	C++
		GreenHills 1.8.9, ppc	C	-
		GreenHills 2.0, ppc	C	-
Solaris	OSE 4.1.1 SoftKernel	Gnu 2.95.1, SPARC	C & C++	-
Windows	OSE 4.1.1 SoftKernel	Visual C++ 6.0, x86	C	-

**Table 2 Referenced Configurations and Targets**

Host Configuration(s)	Target RTOS	Compiler/Processor	RTS Library	Connexis DCS Library
Solaris, Windows	Tornado 2.02 (VxWorks 5.4)	Gnu 2.96	C & C++	C++
		GreenHills 1.8.9, ppc	C & C++	C++
		GreenHills 2.0, ppc	C & C++	C++
Solaris Windows NT	Tornado 2.0 Sim	Gnu 2.96, SPARC Gnu 2.96, x86	C++ C++	C++ C++
Solaris, Windows	LYNX 3.1.0a	gnupro-2.9-98r2, ppc	C++	C++
Solaris	LYNX 3.0.1	Cygnus 2.7.97r1, x86	C++	C++
		Cygnus 2.7.97r1, ppc	C++	C++
Solaris	Chorus Classix 4.0	egcs-2.91.66, ppc	C++	-
Windows	Windows CE 3.01	eMbedded Visual C++ 3.0, sh3	C++	C++
Solaris, Windows	eCos ITRON	Gnu 2.95.3, x86	C	-
Solaris, Windows	No RTOS	Gnu 2.8.1, SPARC	C & C++	C & C++
		Visual C 6.0, x86	C & C++	C & C++
Windows	Nucleus 1.1	Diab 4.2b, ppc	C++	-
N/C - native compilation only	Red Hat Linux 7.1	Egcs 2.91.66, x86	C++	C++
N/C - native compilation only	UnixWare 7.0.1	SDK 3.0, x86	C++	C++

## Creating Executables for Hosts Without Toolset Support

For hosts without toolset support, create an executable on the target.

**Note:** The following steps assume that you use a common file system and that paths are equivalent on both machines.

To produce an executable for a host without toolset support:

- 1 Select **Tools > Options** and click the **C++ Compilation** tab. Click **Select** in the **TargetConfiguration** area.
- 2 In the **Target Configuration** dialog, select the appropriate target configuration and click **OK**.

- 3 On the **C++ Generation** tab, ensure that **CodeGenMakeType** and **CodeGenMakeCommand** are appropriately set for the toolset host.
- 4 On the **C++ Compilation** tab, ensure that **CompilationMakeCmd** and **CompilationMakeType** are appropriately set for the compilation host.
- 5 Build the component with a build level set to **Generate**.

This creates the source files and **makefiles**, required for compilation on the target host.

**Note:** If the target computer that you use to compile does not have a common file system with the code generating host, see *Generating an Executable Without a Common File System* on page 19.

- 6 From the build directory on the target host, set the environment variables for the compilation configuration (line-up).
- 7 Invoke the appropriate make command for the line-up.

**Note:** If you generate the source files on Windows (NT, 2000, or XP) and compile on UNIX, see the steps below about converting Windows files to UNIX type.

## Generating an Executable Without a Common File System

If you build the source files on Windows (NT, 2000, or XP) and compile on UNIX, you must convert your files to UNIX type before you compile and link.

### To generate an executable without a common file system:

- 1 On the target, a visible copy of the TargetRTS must be available.
- 2 Copy the component directory to the target file system.
- 3 Edit the build/**makefile** so that RTS\_HOME is set to location of the TargetRTS.
- 4 If the source was generated on Windows, convert all files in the component directory to UNIX type, using a utility such as **dos2unix**.

This is very important if the target does not support CRLF (Carriage Return Line Feed) line terminators.

**Note:** It may be necessary to convert files in the TargetRTS directory, particularly if some files were edited on Windows.

- 5 From the build directory, set your environment variables appropriately for the compilation configuration.

6 Invoke the appropriate make command for this line-up.

**Note:** You can access a ClearCase server on UNIX with Rational Rose RealTime clients running on both Windows and UNIX workstations.

## Adding a Printer on UNIX

---

Rational Rose RealTime on UNIX uses MainWin (a MainSoft product that allows Windows applications to run in a UNIX environment). Special printer specification is necessary to support the PSCRIPT.

MainWin uses the PSCRIPT keyword in win.ini to specify PostScript support under UNIX, using syntax similar to the way one would use the PSCRIPT driver in Windows. Below is a typical printer-related section of a win.ini file. The win.ini file is located in your Windows directory in your home directory (ie: ~/windows/).

The win.ini entries are more or less the same for MainWin as they are for Windows. An explanation of each section follows the win.ini file lines.

### [windows]

```
device=Apple LaserWriter II NT,PSCRIPT,LPT1
...
```

The device entry in this win.ini **[windows]** section defines the default printer. It takes the following syntax:

```
device=outputdevicename,devicedriver,portconnection
```

The keyword PSCRIPT is used in place of *devicedriver*.

### [ports]

```
LPT1:=lp -c "%s"
LPT2:=lp -c -dps1700 "%s"
LPT3:=
...
```

The win.ini **[ports]** section lists available communication and printer ports. Under MainWin, the Windows LPTn keywords are mapped to UNIX commands. In this example, LPT1 and LPT2 are mapped to the print command lp. MainWin sends all print job output to a file. The output file is then sent to the printer. The term %s tells the system to substitute the name of the PostScript intermediate output file. The term **-dps1700** in the example refers to a UNIX printer named ps1700. The printer should be defined in the UNIX **printcap** file.

## **[PrinterPorts]**

```
Apple LaserWriter II NT=PSCRIPT,LPT1:,15,90  
Postscript Printer QMS=PSCRIPT,LPT2:,15,90
```

The win.ini **[PrinterPorts]** section is included for compatibility with applications that require this section. Entries are similar to those for the **[Devices]** block listed below. In **[PrinterPorts]**, PostScript **timeout** values are appended after the device name. The **timeout** values are not used by MainWin.

## **[Devices]**

```
Apple LaserWriter II NT=PSCRIPT,LPT1:  
Postscript Printer QMS=PSCRIPT,LPT2:
```

The **[Devices]** block lists the active and inactive output devices accessed by device drivers, and specifies the ports to which these devices are connected. In this example, Apple LaserWriter II NT=PSCRIPT,LPT1: specifies that the printer is connected to the PSCRIPT queue connected to LPT1.





# Installing Rational Rose RealTime on Windows

# 3

## Contents

This chapter is organized as follows:

- *Using the Rational Installation Program* on page 24
- *Using the Rational Setup Wizard* on page 25
- *Before You Install* on page 27
- *Upgrade Information* on page 28
- *Specifying the Rational License Server* on page 29
- *Performing a Client Installation* on page 29
- *Post-Installation Tasks* on page 32
- *Creating a Release Area* on page 33
- *Installing Rational Rose RealTime from a Release Area* on page 38
- *Using Silent Installation Commands* on page 43
- *Setting Up Silent Installations of Rational Rose RealTime from a Release Area* on page 45
- *After You Install* on page 46
- *Testing your Environment* on page 49

## Removing Previous Releases of Rational Products

---

Before installing Rational products from release 2003.06.00, you must completely remove all previous releases of Rational products. For the removal procedure, see *Uninstalling Rational Rose RealTime* on page 153. If the Rational Setup Wizard detects products from previous releases, it will not proceed with the installation.

You or your system administrator must see the Rational Suite Upgrade Guide in the Rational Solutions for Windows Documentation CD-ROM before you try to upgrade your Rational products.

If you are using floating licenses, record the license server name(s) before you remove Rational products from your computer. After you install new Rational products on your computer, you need to reset the hostnames in the Rational License Key Administrator. For more information, see the Removing Rational Products chapter in this installation guide.

## Installing Mixed Versions of Rational Products

---

In most cases, you cannot run mixed versions of Rational products on the same computer. You can install mixed versions in the following cases:

- Rational ClearCase with an earlier version of Rational Suite (that does not include Rational ClearCase LT).
- Rational ClearQuest standalone or as part of Rational Suite with full Rational ClearCase 4.0 - 5.0.
- Rational TeamTest with full Rational ClearCase 4.0 - 5.0.
- Rational XDE with full Rational ClearCase 4.2 (fully patched) or 5.0.

## Using the Rational Installation Program

---

Use the Rational installation program for initial and upgrade installations of Rational Rose RealTime. This chapter describes the Rational installation wizard and how the wizard can help you deploy Rational Rose RealTime directly from the installation CD-ROM, from a release area on a network, and from a silent installation file.

Use Table 3 on page 24 to help you find the correct procedures for the deployment method that you have selected for your users.

**Note:** Ask your administrator whether a release area has been set up for you.

**Table 3** Deployment Methods

Method	See
Install directly from the <i>Rational Solutions for Windows</i> CD-ROM.	1 <i>Using the Rational Setup Wizard</i> on page 25. 2 <i>Before You Install</i> on page 27. 3 <i>Specifying the Rational License Server</i> on page 29. 4 <i>Performing a Client Installation</i> on page 29. 5 <i>Post-Installation Tasks</i> on page 32.
Cancel an installation from CD-ROM.	<i>Canceling an Installation from CD-ROM</i> on page 32.
Reinstall from a CD-ROM.	<i>Reinstalling Rational Rose RealTime from CD-ROM (Modify, Repair, Remove)</i> on page 32.

**Table 3    Deployment Methods**

<b>Method</b>	<b>See</b>
Create a release area using the Rational Setup Wizard.	<ol style="list-style-type: none"> <li>1 <i>Using the Rational Setup Wizard</i> on page 25.</li> <li>2 <i>Specifying the Rational License Server</i> on page 29.</li> <li>3 <i>Using the Rational Setup Wizard to Create a Release Area</i> on page 34.</li> <li>4 <i>Post-Installation Tasks</i> on page 37.</li> </ol>
Create multiple site definition files for a release area.	<ol style="list-style-type: none"> <li>1 Create a release area using the Rational Setup Wizard.</li> <li>2 <i>Running the Site Preparation Wizard to Create Multiple Sitedef Files</i> on page 37.</li> </ol>
Install from a release area on a network (using the standard configuration)	<ol style="list-style-type: none"> <li>1 <i>Using the Rational Setup Wizard</i> on page 25.</li> <li>2 <i>Installing Rational Rose RealTime from a Release Area</i> on page 38</li> </ol>
Install from a release area on a network (customizing the client configuration for your desktop)	<ol style="list-style-type: none"> <li>1 <i>Using the Rational Setup Wizard</i> on page 25.</li> <li>2 <i>Before You Install</i> on page 27.</li> <li>3 <i>Specifying the Rational License Server</i> on page 29.</li> <li>4 <i>Installing Rational Rose RealTime from a Release Area</i> on page 38</li> </ol>
Cancel an installation from a release area.	<i>Canceling a Product Installation From a Release Area</i> on page 42
Reinstall from a release area.	<i>Reinstalling Rational Rose RealTime From a Release Area (Modify, Repair, Remove)</i> on page 42
Run a silent installation from a release area.	<ol style="list-style-type: none"> <li>1 <i>Using the Rational Setup Wizard</i> on page 25.</li> <li>2 <i>Setting Up Silent Installations of Rational Rose RealTime from a Release Area</i> on page 45.</li> </ol>
Run a silent installation from a site defaults file.	<ol style="list-style-type: none"> <li>1 <i>Using the Rational Setup Wizard</i> on page 25.</li> <li>2 <i>Silent Installation Overview</i> on page 43.</li> <li>3 <i>Running a Silent Installation on your Desktop</i> on page 44</li> </ol>

## Using the Rational Setup Wizard

---

The Rational Setup Wizard installs Rational Software products. Your Rational product shipment includes three *Rational Solutions for Windows* CD-ROMs:

- Disc 1-2 provide the Rational products to install.
- Disc 3 provides Rational product documentation.

## Rational\_Install Log

The Rational installer does not display an error summary. To verify that an installation was successful or to understand why it failed, look in the log of installation activities called Rational\_install.log (e.g. TEMP=C:\DOCUME~1\\LOCALS~1\Temp or c:\temp\install.log). The location of the temp directory depends on the temp environment variable set on the computer. To find the location, open a command window and type SET at the prompt.

## Before You Start the Rational Setup Wizard

The following general requirements are necessary to run the Rational Setup Wizard on the system.

- Stop all applications before you begin the installation.
- Make sure you have administrator privileges before installing Rational products.

To use the Rational Setup Wizard on a Windows operating system, you must have Windows administrator privileges on the local computer. Log in as one of the following users:

- Local administrator
- Member of the local administrator's group
- Domain administrator who is a member of the local administrator's group
- Turn off all virus protection software. These programs often run in the background and interfere with the install application's performance because the virus protection checks each file that is installed.
- Make certain that the system meets the minimum requirements and the correct operating system.
- The Rational Setup Wizard uses C:\Program Files\Rational as the default install path.

- The Rational Setup Wizard reports the amount of space required on all drives for your installation. To see this information, click the Space button. If your C:\ drive lacks sufficient free disk space, you may either specify another drive during the installation procedure or make space available on the default drive.

**Note:** The Setup Wizard installs Microsoft Core Components and some additional files on the same drive as the operating system (often the C:\ drive), even if you have specified an alternate drive for installation. These files can require 5-15 MB of temporary disk space on your hard drive.

- The Rational Setup Wizard requires that you install all Rational products in the same directory. If you already have Rational products installed on the system, the Setup Wizard installs additional Rational products in the same directory.
- Make sure that you have a current backup of your Registry and system directories.
- Turn off any user interface managers or environments that run on Microsoft Windows.
- Change to a standard VGA video driver while Rational Setup is running, or disable video features such as virtual screens or screen switching.
- Change to a standard mouse driver, or disable special mouse features that perform tasks such as leaving pointer trails or changing pointer sizes.

## Before You Install

---

Before you install Rational Rose RealTime, ensure that you have a supported system configuration. The system requirements are in a table in the section *Referenced Configurations and Toolchain Requirements* on page 13. A setup program is included to facilitate the installation of Rational Rose RealTime on Windows NT, Windows 2000, or Windows XP Pro. You must have administrator privileges to install this software.

**Note:** If you installed the Companion Products for an earlier installation of Rational Rose RealTime, ensure that you also uninstall the Companion Products before installing Rational Rose RealTime 2003.06.00.

For instructions on how to install Rational products, see the *Installation* guides for Rational Desktop Products or Rational Server Products.

There are three types of installations that you can perform:

- Client Installation (a local client installation) - see *Performing a Client Installation* on page 29
- Enterprise Installation - see *Creating a Release Area* on page 33
- Silent Installation from the Network - see *Using Silent Installation Commands* on page 43.

## Preparing for a Rational Rose RealTime Installation

Here is an overview of tasks for installing Rational Rose RealTime as part of your Rational Suite edition.

- To generate and execute C++ code with Rational Rose RealTime, C++ compilers must be installed on your system. For a list of supported compilers and targets, see the *Referenced Configurations and Targets* on page 17.
- To construct and execute UML models, test your Visual C++ environment. To help you determine whether you have correctly installed and configured Visual C++ on your system, see the *Testing your Environment* on page 49.
- To deploy your model on a real-time operating system, see *Referenced Host Configurations* on page 16 for information on referenced configurations.

## Upgrade Information

---

Ensure that past releases of Rational Rose RealTime are removed from your system prior to installation. For details on your specific platform, see *Uninstalling Rational Rose RealTime* on page 153 for your specific platform.

Models created in earlier versions of Rational Rose RealTime can be loaded directly into version 2003.06.00 Rational Rose and ObjecTime Developer models should be converted as described in *Migrating from ObjecTime Developer 5.2/5.2.1* on page 105.

**Note:** Do not attempt to load workspaces created in earlier versions of Rational Rose RealTime, as they are not compatible with the new release.

### Checking the Validity of Your License Keys

If you upgrade to Rational Rose RealTime 2003.06.00 from Rational Rose RealTime releases 6.0, 6.0.1, or 6.0.2, your license keys are not valid. For information on requesting license keys, see *Requesting License Keys* on page 76.

If you upgrade to Rational Rose RealTime 2003.06.00 from Rational Rose RealTime releases 6.1, 2000.02.10, 2001.03.00, 2001A.04.xx, or 2002.05.xx, your license keys are valid.

For more information on license keys, see *Installing License Keys* on page 89.

## Specifying the Rational License Server

---

The Rational License Key Administrator (LKAD) launches at the end of the installation. You can provide a Rational License Server name if you are using floating licenses. Your administrator may have already provided the license server name if you are installing from a release area or running a silent installation script.

## Performing a Client Installation

---

This section describes a typical installation of Rational Rose RealTime from the CD-ROM. The Rational Setup Wizard Program guides you through the software installation. Click **Next** to launch the installation, and to advance through the following screens.

**Note:** Interrupting an installation that is in progress may leave your system in an indeterminate state. If you try to close the Rational Setup Wizard window while the installation is in progress, you are asked to confirm that you want to exit from the incomplete installation.

### To install Rational Rose RealTime from a CD-ROM image:

- 1 Log in as a user with Administrator rights on the local machine on which you want to install Rational Rose RealTime.
- 2 Insert the *Rational Solutions for Windows Disc 1* into the system's CD-ROM drive. The Rational Setup Wizard starts automatically.  
If autorun is disabled on your system, click **Start** > **Run** and enter *cd\_drive: \Setup.exe* where *drive* is the letter of the CD-ROM drive.
- 3 The **Welcome** page to the Rational Setup Wizard appears. Click **Next** to launch the installation, and to advance through the following screens.
- 4 The **Product Selection** page lists all products available for installation. Select Rational Rose RealTime.

- 5 The **Deployment Method** page displays the **Enterprise Deployment** and **Desktop Installation from CD Image** options. Select the **Desktop Installation from CD Image** option.
- 6 Choose to accept or not to accept the Rational Software license agreement in the **License Agreement** page.
  - If you accept the license agreement, the installation Wizard continues.
  - If you do not accept the license agreement, exit the Setup Wizard by clicking **Cancel** and then **Finish**. When you exit from the Wizard, the Setup Wizard makes no visible changes to your system.
- 7 On the **Destination Folder** page, specify the directory where you want to install Rational Rose RealTime. Click **Change** to modify the location.
 

**Note:** The installation Wizard requires that all Rational products be installed in the same directory.
- 8 The **Custom Setup** page provides product feature options for the software installation. You can either accept the default features on the page or you can customize the installation.

If you want to change the features, use the **Help**.

**Note:** You must select the Rational Rose RealTime component. Rational Rose RealTime must be installed in order to install the other addins.

**Table 4 Custom Setup Options**

Option	Description
Rational Rose RealTime	Installs the full install of Rational Rose RealTime including all addins.
Rational Rose RealTime/Connexis	Installs only the Rational Rose RealTime Connexis addin.
Rational Rose RealTime/Rational Quality Architect	Installs only the Rational Quality Architect - RealTime addin.
Rational Rose RealTime/Target Services Library Source	Installs only the TargetRTS run-time services.



- 9** The Wizard informs you of the disk space required to install Rational Rose RealTime and the available disk space on the computer. Click the **Space** on the bottom of the Custom Setup page to display the **Disk Space Requirements** page. Click **OK** to close the page. If the amount of free space is less than the amount required:
- Exit from the installation and make more space available on the specified disk, or
  - Specify an alternate *Install Path*. On the **Destination Folder** page, click **Change**. Enter the new path in the **Folder Name** text field or use the **Folder** icon to select a drive and directory, or
  - Exclude product features in the **Custom Setup** page.
- 10** Click **Install** on the **Ready to Install the Program** page to begin the installation.
- 11** A **Restart Windows** page appears if the Rational Setup Wizard needs to restart your computer. If files required for the installation are in use during the Rational Setup program and if the program needs to install shared components on your system, the Setup Wizard may need to restart your system.
- Select **Restart** or **Don't Restart**. If you select **Don't Restart**, the Wizard reminds you that the installation cannot complete until Windows restarts.
- After Windows restarts, log on *as the same user*. If you do not, the installation does not complete correctly. The second part of the installation process starts automatically after you log on.
- 12** When the Rational Setup Wizard Completed page appears, we recommend that you review the current information related to new features and known issues in the **readme** file. In addition, you can view the **Rational Developer Network Web** pages. Click **Finish** to complete the installation.
- 13** Install the License Key, if required. For information on how to install your license key, see *Installing License Keys* on page 89.

## Post-Installation Tasks

---

### Licensing

Rational products require licenses. If you do not see the License Key Administrator (LKAD) launch at the end of the installation, your product is licensed.

You or your users may see the License Key Administrator (LKAD) launch at the end of a product installation for one of the following reasons:

- You did not provide a Rational license server name when you created the site definitions file (applies to installations from a release area).
- The product requires a desktop license key.

If you see the LKAD, you or your users must perform the following tasks to license the product.

- To configure a floating license key, enter the name of the Rational license server in the LKAD. For more information, see *Installing a Startup or Permanent License on Windows* on page 89.
- To configure a desktop license key, import the node-locked or per user license key file in the LKAD. For more information, see *Installing a Startup or Permanent License on Windows* on page 89

### Canceling an Installation from CD-ROM

If you click Cancel any time during the Setup Wizard or before the installation completes, you will not see any visible changes to the system. The program returns your system to the state it was in before you launched the Rational Setup Wizard.

### Reinstalling Rational Rose RealTime from CD-ROM (Modify, Repair, Remove)

If you have successfully completed a previous installation of Rational Rose RealTime in release 2003.06.00, you can re-run the Rational Setup Wizard **to modify (add or remove product components) an installation. If you have inadvertently removed files from the installation directory, you can re-run Rational Setup Wizard to repair the installation.**

The Rational Setup Wizard guides you through the software installation. In each dialog, click Next to open the next one.

## To reinstall from the CD-ROM :

- 1 Log in as a user with Administrator rights on the local machine on which you want to install Rational Rose RealTime.
- 2 Insert the *Rational Solutions for Windows Disc 1* into the system's CD-ROM drive. The Rational Setup Wizard starts automatically.  
If autorun is disabled on your system, click **Start** > **Run** and enter *cd\_drive: \Setup.exe* where *drive* is the letter of the CD-ROM drive.
- 3 The **Rational Setup Wizard** provides general information about the software installation. Click **Next** to open the Program Maintenance dialog. This dialog offers three options:
  - **Modify the Existing Installation** – Choosing this option enables you to change which products and product features are installed.  
To modify the existing installation, click **Modify** and then click **Next** to select or clear product features in the Custom Setup dialog. Click **Next** and then click **Install** to begin the installation.
  - **Repair the Existing Installation** – Choosing this option enables you to repair any installation files that may be damaged. This option may repair a damaged registry or replace files that you may have inadvertently deleted. The Repair option does not repair incomplete installations or an unsuccessful installation.  
To repair an installation, click **Repair** and click **Next** and then click **Repair** to begin the repair. At the end of the repair operation, the status of the repair is displayed.
  - **Remove the Existing Installation** – Choosing this option removes all files that you previously installed for this product.  
To remove the existing installation, click **Remove** and then click **Next**. Click **UnInstall** to begin the de-installation.

## Creating a Release Area

---

The release area contains site defaults files and all the files that will be used in subsequent installations. For example, part of setting up a release area is to specify information, such as client software and license servers. This information is stored in a site defaults file and used when a Rational product, such as Rose RealTime, is installed on clients.

There are two methods for creating a release area and populating it with site defaults files. Both methods create an on-disk image of product files in a shareable directory on the network. You can also use these methods to install the product on your computer after creating a site defaults file.

- Run the Rational Setup Wizard on the *Rational Solutions for Windows CD-ROM*. See *Using the Rational Setup Wizard to Create a Release Area* on page 34 for more information.
- Run the Site Preparation Wizard multiple times to create multiple site defaults files. See *Running the Site Preparation Wizard to Create Multiple Sitedef Files* on page 37 for more information.

## Using the Rational Setup Wizard to Create a Release Area

This section explains how you can use the Rational Setup Wizard to create site defaults files for a release area. You can create a meaningful name for the site defaults file. By default, the file is named `sitedefs.dat` if you do not specify a name for it.

You can use this release area to install Rational servers and client software. Client users can use this release area to install Rational products on their desktops.

- 1 Make the release area directory shareable. Even if the drive containing that directory is already shareable, making the directory itself shareable makes it easier to find the product release area.
  - a In Windows Explorer, right-click the network release area to display the directory shortcut menu.
  - b Click **Sharing**. The **Properties** page appears.
  - c On the **Sharing** tab, click **Shared this folder** and supply a meaningful share name, such as **Rose RealTime 2003.06.00 Release Area**.

- 2 Log in as a user with Administrator rights on the local machine.

- 3 Insert the *Rational Solutions for Windows Disc 1* into the system's CD-ROM drive.

The Rational Setup Wizard starts automatically.

If autorun is disabled on your system, click **Start** > **Run** and enter `cd_drive: \Setup.exe` where *drive* is the letter of the CD-ROM drive.

The Rational Setup Wizard Program guides you through the software installation. Click **Next** to open the page.

- 4 The **Product Selection** page lists all products available for installation. Select Rational Rose RealTime. Click **Next**.

- 5 The **Deployment Method** page displays the **Enterprise Deployment** and the **Desktop installation from CD image** options. Select the **Enterprise Deployment** option to create a release area. Click **Next**.
- 6 The custom configuration page displays in the wizard. Use the **Help** to provide instructions in this section of the wizard.

Enter the required information in each page of the wizard. (All required information displays in the left panel of the wizard with a red dot.)

To navigate through the pages, you can either click **Next** to see them sequentially or click on the page title in the left pane to access the page directly and nonsequentially.

- 7 In the **Completion** pages, you are required to fill in the **Description** page and the **Create a Release Area** page. The **Launch Installation** is optional.
  - a In the **Description** page, enter a description for users. When you are done, click **Next**.
  - b In the **Create a Release Area** page, enter the release area location and filename for the site defaults file that will be created.
    - If you want to install software on your computer based on the site defaults that you just entered, click **Next** to go to the **Launch Installation** page. You will save the site defaults information to a file and then proceed with the installation on your computer. For the rest of the instructions, see *Installing Rational Rose RealTime on Your Computer* on page 35.
    - If you only want to create a release area, click **Done** and then **Next** to create the site defaults file.
- 8 When the Rational Setup Wizard Completed page appears, we recommend that you review the current information related to new features and known issues in the **readme** file. In addition, you can view the **Rational Developer Network** Web pages. Click **Finish** to complete the installation of the release area.

For more information about how to install Rational Rose RealTime from this release area, see *Installing Rational Rose RealTime from a Release Area* on page 38.

## Installing Rational Rose RealTime on Your Computer

When you create a release area, you have the option to install Rational Rose RealTime on your desktop.

## To install Rational Rose RealTime on your computer:

- 1 In the **Launch Installation** page, select **Launch installation after saving site defaults information** and then click **Done** and then **Next**.
- 2 When the Rational Setup Wizard Completed page appears, click **Finish** to complete the installation of the release area. The opportunity to see the release notes and visit the Rational Developer Network will appear at the end of the product installation on your computer.
- 3 When the installation begins, click **Next**. Choose to accept or not to accept the Rational Software license agreement in the **License Agreement** page.
  - If you accept the license agreement, the installation Wizard continues.
  - If you do not accept the license agreement, exit the Setup Wizard by clicking **Cancel** and then **Finish**. When you exit from the Wizard, the Setup Wizard makes no visible changes to your system.
- 4 Specify the **Destination folder** using the **Destination Folder** page, then click **Next** or change the destination folder, by clicking **Change**.
- 5 In the Site Default Configuration page, decide whether you want the site defaults that you set installed on your computer. Either click **Use the standard configuration** (These are site defaults that you set.) or **I will create my own custom client configuration**.
  - If you select **I will create my own custom client configuration**:
    - The **Custom Setup** page displays product components to select.
    - When you click **Next**, you may change any of the existing site default values. (Any changes to the site default values apply only to this single installation.) After modifying the values, click **Done** and then click **Install** to begin the installation.
  - If you select **Use the standard configuration**, then the default features for the existing site default values will be used for the installation. Click **Next** and then click **Install** to begin the installation.
- 6 A **Restart Windows** page appears if the Rational Setup Wizard needs to restart your computer. If files required for the installation are in use during the Rational Setup program and if the program needs to install shared components on your system, the Setup Wizard may need to restart your system.

Select **Restart** or **Don't Restart**. If you select **Don't Restart**, the Wizard reminds you that the installation cannot complete until Windows restarts.

After Windows restarts, log on *as the same user*. If you do not, the installation does not complete correctly. The second part of the installation process starts automatically after you log on.

- 7 When the Rational Wizard Completed page appears, we recommend that you review the current information related to new features and known issues in the **readme** file. In addition, you can view the **Rational Developer Network** Web pages. Click **Finish** to complete the installation.

## Post-Installation Tasks

### Licensing

Rational products require licenses. If you do not see the License Key Administrator (LKAD) launch at the end of the installation, your Rational Rose RealTime is licensed.

You or your users may see the License Key Administrator (LKAD) launch at the end of a product installation for one of the following reasons:

- You did not provide a Rational license server name when you created the site definitions file (applies to installations from a release area).
- The product requires a desktop license key or license server name.

If you see the LKAD, you or your users must perform the following tasks to license Rational Rose RealTime.

- To configure a floating license key, enter the name of the Rational license server in the LKAD. For more information, see the *Rational Software License Management Guide*.
- To configure a desktop license key, import the node-locked or per user license key file in the LKAD. For more information, see the *Rational Software License Management Guide*.

## Running the Site Preparation Wizard to Create Multiple Sitedef Files

In some cases, your site may require multiple site defaults files. For example, if two groups both use Rational Rose RealTime but need to work with different default settings, then you can create one site defaults file for each group.

### To create multiple site defaults for a release area:

- 1 Create the initial release area by following the instructions in *Using the Rational Setup Wizard to Create a Release Area* on page 34. Set up this area with the site-specific parameters relevant to the first group of users.
- 2 Rerun the Rational Site Preparation Wizard from the release area that you created in Step 1. Double-click `siteprep.exe` in the release area or run `siteprep.exe` on the command line.
- 3 Follow the instructions to set the product parameters.
  - If you have already created one site defaults file (`sitedefs.dat` for instance), in the initial release area, the Rational Site Preparation Wizard displays the values set in `sitedefs.dat`. Keep the values that apply to both groups and change the ones according to the needs of the second group.
  - If you have previously created multiple site defaults files, you can select a specific site defaults file `*.dat` as a starting point by clicking **File > Open**.
- 4 Click **File > Save as** to save a new site defaults file. You are prompted to enter a file name and folder for the new site defaults file:
  - If you started the site preparation on the command line and specified a file-name argument for the site defaults file, for example, **sitedefs.dat**, the **Folder** and **File name** boxes display this information. You could save the modified site defaults files with a new file name, for example **sitedefs\_cqclient.dat**
  - If no file name was previously specified, the **File name** box is blank. Type a file name that does not currently exist in the release area.

**Note:** If you enter the name of an existing site defaults file, a warning message appears. You can overwrite the existing file or specify a different file name to create a new site defaults file.

According to your needs, you may create additional site defaults files in this way.

## Installing Rational Rose RealTime from a Release Area

---

When users install Rational Rose RealTime from a release area, in most cases, they accept the defaults as presented on the installation screens. Users who do not want to accept the defaults should speak to you before they make changes.



### **Installing from a release area includes the following:**

- 1 You or an administrator create one or more site defaults (sitedefs) files in a release area using the Rational Setup Wizard or the Site Preparation Wizard. You inform users of the name of the sitedefs file and network location of the release area or send them a shortcut to the file.
- 2 Users can then access the publicly accessible area and execute the site defaults file shortcut from their desktop clients. This shortcut will run an installation on their desktops from the release area.

To install Rational Rose RealTime from a Release Area, you can use the standard configuration or you can customize the standard client configuration for your desktop. The first section, *Using a Standard Configuration* on page 39, describes the procedure for using a standard configuration. The second section, *Customizing Your Own Configuration* on page 40, describes the procedure for customizing your configuration.

## **Using a Standard Configuration**

### **To install a default configuration from the release area:**

- 1 Log on as a user with local administrator privileges.
- 2 To install a Rational product using a specific site defaults file, either run setup.exe on the command line and specify the name of the site defaults file or click the associated site defaults shortcut in the release area. For example, to install ClearQuest using a site defaults file named sitedefs\_cqclient.dat, map a network drive from your computer to the shared release area. Then,
  - On the command line, use the cd command to navigate to the root directory of the release area. Then enter: setup.exe sitedefs\_cqclient.dat, or
  - Double-click the mapped drive and click the sitedefs\_cqclient.dat. shortcut.
- 3 The Rational Setup Wizard runs and guides you through the software installation. In each page, click **Next** to open the next page. Click **Help** for more information.
- 4 The **License Agreement** page displays the Rational Software license agreement. If you accept the license agreement and click **Next**, the installation continues. If you do not accept it, the installation does not let you proceed further. If you click **Cancel** and exit from the installation, no changes are made to your system.
- 5 The **Destination Folder** page displays the default destination folder for the installation. Click **Change** to select a different destination folder for the installation. Click **Next**.

- 6 Click **Use the standard configuration** on the **Site Default Configuration** page. The default features for Rational Rose RealTime and the existing site default values will be used for the client installation. Click **Next**.
- 7 Click **Install** to begin the installation on your client desktop.
- 8 A **Restart Windows** page appears if the Rational Setup Wizard needs to restart your computer. If files required for the installation are in use during the Rational Setup program and if the program needs to install shared components on your system, the Setup Wizard may need to restart your system.

Select **Restart** or **Don't Restart**. If you select **Don't Restart**, the Wizard reminds you that the installation cannot complete until Windows restarts.

After Windows restarts, log on *as the same user*. If you do not, the installation does not complete correctly. The second part of the installation process starts automatically after you log on.

- 9 When the Rational Wizard Completed page appears, we recommend that you review the current information related to new features and known issues in the **readme** file. In addition, you can view the **Rational Developer Network** Web pages. Click **Finish** to complete the installation.

## Customizing Your Own Configuration

### To customize a configuration for a specific computer:

- 1 Log on as a user with local administrator privileges.
- 2 To install a Rational product using a specific site definitions file, either run `setup.exe` on the command line and specify the name of the site defaults file or click the associated site defaults shortcut in the release area. For example, to install ClearQuest using a site defaults file named `sitedefs_cqclient.dat`, map a network drive from your computer to the shared release area. Then,
  - On the command line, use the `cd` command to navigate to the root directory of the release area. Then enter: `setup.exe sitedefs_cqclient.dat`, or
  - Double-click the mapped drive and click the `sitedefs.cqclient.dat` shortcut.
- 3 The Rational Setup Wizard runs and guides you through the software installation. In each page, click **Next** to open the next page. Click **Help** for more information.
- 4 The **License Agreement** page displays the Rational Software license agreement. If you accept the license agreement and click **Next**, the installation continues. If you do not accept it, the installation does not let you proceed further. If you click **Cancel** and exit from the installation, no changes are made to your system.

- 5 The **Destination Folder** page displays the default destination folder for the installation. If you want to select a different destination folder for the installation, click **Change**.
- 6 Click **I will create my own custom client configuration** on the **Site Default Configuration** page.
  - The Custom Setup page displays product features to select.
  - When you click **Next**, you may change any of the existing site default values. (Any changes to the site default values apply only to this single installation.) After modifying the values, click **Done**.
- 7 Click **Install** to begin the installation on your client desktop.
- 8 A **Restart Windows** page appears if the Rational Setup Wizard needs to restart your computer. If files required for the installation are in use during the Rational Setup program and if the program needs to install shared components on your system, the Setup Wizard may need to restart your system.

Select **Restart** or **Don't Restart**. If you select **Don't Restart**, the Wizard reminds you that the installation cannot complete until Windows restarts.

After Windows restarts, log on *as the same user*. If you do not, the installation does not complete correctly. The second part of the installation process starts automatically after you log on.
- 9 When the Rational Wizard Completed page appears, we recommend that you review the current information related to new features and known issues in the **readme** file. In addition, you can view the **Rational Developer Network** Web pages. Click **Finish** to complete the installation.

## Post-Installation Tasks

### Licensing

Rational products require licenses. If you do not see the License Key Administrator (LKAD) launch at the end of the installation, Rational Rose RealTime is licensed.

You or your users may see the License Key Administrator (LKAD) launch at the end of a product installation for one of the following reasons:

- You did not provide a Rational license server name when you created the site definitions file (applies to installations from a release area).
- The product requires a desktop license key.

If you see the LKAD, you or your users must perform the following tasks to license Rational Rose RealTime.

- To configure a floating license key, enter the name of the Rational license server in the LKAD. For more information, see the *Rational Software License Management Guide*.
- To configure a desktop license key, import the node-locked or per user license key file in the LKAD. For more information, see the *Rational Software License Management Guide*.

## Canceling a Product Installation From a Release Area

If you click Cancel any time during the Setup Wizard or before the installation completes, you will not see any visible changes to the system. The program returns your system to the state it was in before you launched the Rational Setup Wizard.

## Reinstalling Rational Rose RealTime From a Release Area (Modify, Repair, Remove)

If you have successfully completed a previous installation of Rational Rose RealTime release 2003.06.00, you can re-run the Rational Setup Wizard to modify (add or remove product components) an installation. If you have inadvertently removed files from the installation directory, you can re-run the site defaults file to repair the installation.

The Rational Setup Wizard guides you through the software installation. In each dialog, click **Next** to open the next one.

### To reinstall from a release area:

- 1 Log in as a user with Administrator rights on the local machine on which you want to install Rational Rose RealTime.
- 2 To modify, repair, or remove a Rational Rose RealTime installation installed from a release area, either run `setup.exe` on the command line and specify the name of the associated site defaults file that was used at installation time, or click the associated site defaults shortcut in the release area. For example, to install ClearQuest using a site defaults file named `sitedefs.developers`, map a network drive from your computer to the shared release area. Then,
  - On the command line, use the `cd` command to navigate to the root directory of the release area. Then enter: `setup.exe sitedefs.developers`, or
  - Double-click the mapped drive and click the `sitedefs.developers` shortcut.

3 The **Rational Setup Wizard** provides general information about the software installation. Click **Next** to open the Program Maintenance dialog. This dialog offers three options:

- **Modify the Existing Installation** – Choosing this option enables you to change which product features are installed.

To modify the existing installation, click **Modify** and then click **Next** to select or clear product features in the Custom Setup dialog. Click **Next** and then click **Install** to begin the installation.

- **Repair the Existing Installation** – Choosing this option enables you to repair any installation files that may be damaged. This option may repair a damaged registry or replace files that you may have inadvertently deleted. The Repair option does not repair incomplete installations or an unsuccessful installation.

To repair an installation, click **Repair** and click **Next** and then click **Repair** to begin the repair. At the end of the repair operation, the status of the repair is displayed.

- **Remove the Existing Installation** – Choosing this option removes all files that you previously installed for this product. Before you select this option, we recommend that you read the *Remove, Repair and Modify* chapter in this manual.

To remove the existing installation, click **Remove** and then click **Next**. Click **UnInstall** to begin the de-installation.

## Using Silent Installation Commands

---

A *silent* installation lets you install a Rational Software product, using the same parameters, repeatedly on a number of systems.

### Silent Installation Overview

Your administrator may set up a site defaults file so that many users can perform unattended installations of a Rational Software product with the same parameters.

The site defaults file directs the Rational Setup Wizard to install program files in a specific directory on your computer. For information about setting up a silent installation, see *Using Silent Installation Commands* on page 43.

When you start the silent installation, you do not see any installation screens. If a restart is required, your computer restarts automatically. After the computer restarts, you must log on manually. The install wizard then re-launches automatically and finishes. When the installation finishes, you do not see an installation complete screen

at the end. If your administrator did not specify the license server in the site defaults file or you are using a node-locked or per user license key, you may have to manually configure licensing after the Setup Wizard finishes.

## Running a Silent Installation on your Desktop

The following instructions describe the commands you need to run a silent installation on your computer.

To run the silent installation on your desktop, your administrator gives you the following:

- Path to the site defaults file in the network release area.
- Installation directory (where the Rational Setup Wizard will install the files on your desktop).
- License key information, if necessary.

### To run the site defaults file in in silent mode:

- 1 Map a local drive to the release area.
- 2 Go to **Start > Run** and enter `cmd.exe` in the Run window.
- 3 At the DOS prompt, enter the following command:

```
<local drive>: /setup.exe /g <sitedefaults.dat>
```

The executable installs the products specified in the site defaults file from the source directory to the installation path. The default installation path is `C:\Program Files\Rational\<Rational products>`.

**Note:** If the Rational Setup Wizard detects insufficient disk space on the desktop or server, the Wizard will cancel the installation and note the error in the `Rational_install.log` in your `%TEMP%` directory (for example, `TEMP=C:\DOCUME~1\<username>\LOCALS~1\Temp` or `c:\temp\install.log`). The location of the temp directory depends on the temp environment variable set on the computer. To find the location, open a command window and type `SET` at the prompt.

## Licensing Your Rational Product

- If the License Key Administrator (LKAD) Wizard launches at the end of the installation, specify the Rational license server or import or enter the license key information into the LKAD Wizard. For more information, see the *Rational Software License Management Guide*.

# Setting Up Silent Installations of Rational Rose RealTime from a Release Area

---

You can configure the Rational Setup Wizard to perform silent installations of Rational Rose RealTime. Silent installations let you perform an installation of Rational Rose RealTime, using the same parameters, repeatedly on a number of computers. Silent installations ensure that the correct configuration is installed on each user's computer.

## To set up and run a silent installation:

- 1 Create a release area and site defaults file. You can customize site defaults files for different sets of users. For more information, see *Creating a Release Area* on page 33.
- 2 Use the `setup.exe /g` command to run the site defaults file. You should not see any screens displayed on your computer.

The file directs the Rational Setup Wizard to install program files in a specific directory on your system. If a restart is required, your computer restarts automatically. After the restart, you must log on manually. The installer then re-launches automatically and finishes. When the installation finishes, you do not see an installation complete screen at the end.

If you did not specify the license server in the `sitedefs` file, you and your users may have to manually configure licensing after the Setup Wizard finishes.

## Running a Silent Installation

After you have recorded, play the site defaults file. For example:

```
setup.exe /g <C:\silent installs\sitedefs.dat>
```

`C:\silent installs` is the release area,  
and `sitedefs.dat` is the name of the site defaults file.

By default, the installation log file (`rational_install.log`) is created in your computer temp directory. To find the temp directory, open a command prompt and type `Set`.

## Canceling a Silent Installation

There is no command to cancel a silent installation.

## Command Line Syntax to Run Silent Install

This section provides the syntax for setup.exe.

### Syntax

Setup.exe Command Parameter	Description
/g	Play the silent installation session.
<sitedefs-file>	Specifies the site defaults file in the release area.

Now, you will want to review the topic, *After You Install* on page 46 for additional post-installation activities.

## After You Install

---

After you install Rational Rose RealTime, you may have to perform additional activities, such as configuring environment variables or updating environment variables in your batch files

### Updating Batch Files

If you use a batch file to start Rational Rose RealTime, after you install, you must modify the environment variables to use the new mapped drive. To successfully launch Rational Rose RealTime, you must specify a fully qualified path, including the drive letter. For example, you want to update the ROSE\_HOME variable, as well as the launch command path:

```
set ROSE_HOME=C:\Program Files\Rational\Rose RealTime
set ROSE_HOST=win32
set ROSE_LICENSE_FILE=%ROSE_HOME%\license\license.dat
set path=%ROSE_HOME%\bin\%ROSE_HOST%;c:\Program Files\Microsoft
Visual Studio\Common\VSS\win32;c:\DevStudio\VSS\win32;%PATH%
"C:\Program Files\Rational\Rose RealTime\bin\win32\RoseRT"
```



## Configuring Your Environment

After installation, you must ensure that your environment is properly configured for your compiler.

### Tornado Environment Variables

If you are using Rational Rose RealTime with Tornado, you may want to set the `ROSERT_TORNADO_OPTIONS` and `ROSERT_TORNADO_TIMEOUT` environment variables.

When the `ROSERT_TORNADO_OPTIONS` is set to `VX_FP_TASK`, the application checks if the processor supports floating point, and if so, it runs the target with the floating point option.

When the `ROSERT_TORNADO_TIMEOUT` is set, this is used for the WTX request timeout. The timeout value is in milliseconds.

**Note:** When downloading a VxWorks module to target, by default the timeout for WTX commands is set to 30 seconds (`ROSERT_TORNADO_TIMEOUT=30000`). This may not be enough when downloading large modules or when using a slow network. You may get an error if the timeout is exceeded. This variable allows you to increase the timeout period.

## ClearCase Workstation Setup

---

The following setup must take place on all workstations that will be accessing a VOB or view. For Windows NT, Windows 2000, and Windows XP, this includes all workstations used for development.

These steps will also need to be run on all machines that act as view servers for the ClearCase views used by Rational Rose RealTime. If you use ClearCase MultiSite, you will need to do this at all the sites where the VOBs containing the Rose RealTime elements are replicated.

You can determine which machines are view servers by typing the following:

```
cleartool lsview
```

in a command window. The second item on each output line indicates the machine name where the view server is running. For example, if you see the following line in the output of the `lsview` command:

```
myview \\mymachine\vws\myview.vws
```

then "mymachine" is the name of the machine where the view server for `myview` exists.

For further details, see your ClearCase administrator.

## Command Line Access to the Source Control Tool

For any user wishing to use Rational Rose RealTime's integration with ClearCase, **cleartool** must be accessible from the command prompt.

## Element type setup: type manager

The following steps are required for making ClearCase clients aware of the new element type.

### Windows NT/2000/XP

In the instructions below, <atria-home> refers to the ClearCase installation directory. For newer releases, this typically is c:\Program Files\Rational\ClearCase. For older releases, this typically was c:\Atria.

- From a command prompt, run

```
rtperl <ROSSERT_HOME>\bin\<ROSSERT_HOST>\cc\mi_typeman.pl  
-atriahome <atria-home>
```

## ClearCase Options

### Windows NT/2000/XP

Rational Rose RealTime is case-sensitive when looking for file names.

#### To set the preserve case option for the ClearCase MVFS on Windows:

- 1 In the ClearCase HomeBase tool, select the **MVFS** tab. (The ClearCase Control Panel tool can be started from either the Windows Control Panel or from the **Administration** tab in the **HomeBase** tool)
- 2 Make sure the "preserve case" check box is checked.
- 3 The MVFS service must be restarted for this change to take effect.

## Configuring the ClearCase Repository

Each VOB must be set up to allow files of the new element type to be created. Follow the steps that apply to your platform below for each VOB that will be storing Rational Rose RealTime files.

## Windows NT/2000/XP

Open a command prompt window and change directory to a path within the VOB in which you wish to register the type. To create the element type, use the following command syntax:

```
cleartool mkeltype -supertype text_file -manager  
    petalrt_file_delta -c "RoseRT files" rosert_unit
```

### Test the Type Manager

To determine if the `rosert_unit` element type has been successfully registered in the VOB, perform the following command from a command prompt after changing to a directory contained in the VOB:

```
cleartool lstype -long eltype:rosert_unit
```

A listing of the type details will verify that it is correctly registered.

## Testing your Environment

---

**Note:** If you only want to construct UML models and not execute them, you do not need to read the remainder of this chapter.

You must have Microsoft Visual C++ 6.0 or 7.0 installed on your system and configured to be run from the DOS prompt to make use of the code compilation and execution capabilities of Rational Rose RealTime.

The following instructions help you to determine whether you have Visual C++ properly installed and configured on your system.

### To perform testing on your environment:

- 1 From the Windows **Start** menu:
  - In Windows NT, choose **Start > Programs > Command Prompt**
  - In Windows 2000 and Windows XP, choose **Start > Programs > Accessories > Command Prompt**
- 2 Type **nmake** and press **ENTER**.

### 3 Type **cl** and press **ENTER**.

If your environment is correct, then you should see the following report errors:

#### Command Prompt

```
Microsoft © Windows NT ™  
© Copyright 1985-1996 Microsoft Corp.
```

```
C:\>nmake
```

```
Microsoft © Program Maintenance Utility Version 6.00.8168.0
```

```
Copyright © Microsoft Corp 1988-1998. All rights reserved.
```

```
NMAKE = fatal error B1864: MAKEFILE not found and no target specified
```

```
Stop.
```

```
C:\>cl
```

```
Microsoft © 32-bit C/C++ Optimizing Compiler Version 12.00.8168 for  
80x86
```

```
Copyright © Microsoft Corp 1984-1998. All rights reserved.
```

```
Usage = cl { option... } filename... { /link linkoption... }
```

**Note:** If your environment is NOT properly configured, then you will see an error similar to this one:

```
Command Prompt
```

```
C:\> nmake
```

```
The name specified is not recognized as an internal or external  
command, operable program or batch file.
```

**Note:** If you receive this error message, your compiler environment setup is not configured properly. There is a **vcvars32.bat** file located in the installation directory for Microsoft Visual Studio (for example, **\\Program Files\Microsoft Visual Studio\VC98\Bin\vcvars32.bat**) that lists the environment variables that you must configure.

# Installing Rational Rose RealTime on UNIX

# 4

## Contents

This chapter is organized as follows:

- *Before You Install* on page 51
- *Upgrade Information* on page 52
- *Installation Instructions* on page 54
- *After You Install* on page 58

## Before You Install

---

Before you install Rational Rose RealTime on UNIX, refer to the items in Table 5 to direct you to information in this manual that can help you perform pre-installation tasks.

**Note:** If you installed the Companion Products for an earlier installation of Rational Rose RealTime, ensure that you also uninstall the Companion Products before installing Rational Rose RealTime 2002.06.00.

**Table 5** UNIX Pre-Installation Tasks

License your Rational software	<i>Specifying the Rational License Server</i> on page 29 and <i>UNIX Licenses</i> on page 87
Ensure that your system meets the minimum or recommended system and software requirements	<i>Requirements for UNIX</i> on page 15
Upgrade from earlier versions of Rational software	<i>Upgrade Information</i> on page 52

## Installing in Secure Environments

Problems may occur when trying to perform a remote installation of Rational Suite DevelopmentStudio RealTime (UNIX) in a secure environment (for example, remote access to other machines is through `ssh`) if the environment does not have access to `rsh` or `remsh`. To install Rational Suite DevelopmentStudio RealTime (UNIX) in this situation, perform a local installation of the software rather than a remote installation. If you experience further problems, contact Rational Customer Support.

## Installing Multiple OS Versions of Rational Suite DevelopmentStudio RealTime (UNIX)

If you wish to install different OS versions of Rational Suite DevelopmentStudio RealTime (UNIX) on the same file server, we recommend that you install them in different rational directories (referred to as `<rational_dir>`). If you install them into the same Rational directory, you will not be able to uninstall a single OS version later, if necessary. The uninstall script removes all OS versions that reside in the same Rational directory.

## Stopping and Restarting an Installation

You can stop an installation by entering `q` to quit the installation. If you choose `q`, most of your input is saved to a user defaults file located in `<rational_dir>/config/defaults`. The file name itself is in the following format:

`rs_install.release_name.user_name`

The user defaults file contains general purpose defaults that relate to the username and the license server that you configure. It also keeps track of the product-specific information for the installation of this specific Suite and version.

**Note:** If you enter `q!`, your entries are not saved to the user defaults file.

You can restart the installation by running `rs_install` again. Many of your entries appear as the default value. Press the **ENTER** key to continue with the installation.

## Upgrade Information

---

Refer to the following topics:

- *Upgrading to New Version Only (Uninstalling Earlier Version)* on page 53
- *Upgrading to 2003.06.00 While Maintaining an Earlier Version* on page 54

## Upgrading to New Version Only (Uninstalling Earlier Version)

You can load models created in earlier versions of Rational Rose RealTime directly into 2003.06.00. To convert your existing ObjecTime Developer models, see *Migrating from ObjecTime Developer 5.2/5.2.1* on page 105.

**Note:** Do not attempt to load workspaces created in earlier versions of Rational Rose RealTime, as they are not compatible with the new release.

**If you are upgrading Rational Rose RealTime on any of the UNIX platforms, you must do one of the following:**

- **Manually delete your ~/.registry directory before you run the new version for the first time**
- or
- **Add the "-recreate\_registry" command line option the first time you run the new version.**

### Checking the Validity of Your License Keys

If you upgrade to Rational Rose RealTime 2003.06.00 from Rational Rose RealTime releases 6.0, 6.0.1, or 6.0.2, your license keys are not valid. For information on obtaining new license keys, see *Requesting License Keys* on page 76.

If you upgrade to Rational Rose RealTime 2003.06.00 from Rational Rose RealTime releases 6.1, 2000.02.10, 2001.03.00, 2001A.04.xx, or 2002.05.00, your license keys are valid.

For more information on license keys, see *Installing License Keys* on page 89.

## Upgrading to 2003.06.00 While Maintaining an Earlier Version

Your Unix environment can continue to have a Rational Rose RealTime 2003.06.00 installation and an earlier release of Rational Rose RealTime that uses Unix environment variables. Refer to the following pseudo code to set up your environment to use both releases of Rational Rose RealTime (.csh or .sh setup):

if your current softlink is set to an old version

set up the following environment variables

```
ROBERT_HOME
ROBERT_HOST
ROBERT_LICENSE_FILE
```

else

```
source <rational_dir>/rosert_setup.csh or
. <rational_dir>/rosert_setup.sh
```

set up the following

```
CONNEXIS_HOME to $ROBERT_HOME/Connexis
```

## Installation Instructions

---

**Note:** Unless specified otherwise, your system administrator will generally carry out these steps.

For environments where there is more than one user of Rational Suite DevelopmentStudio RealTime (UNIX), we strongly recommend that you install the main Rational Rose RealTime files on a centralized file server.

Default values, where provided, are prefixed with the following notation:

-->

To accept the default value, simply press **ENTER**.

### Installation Overview

The following provides an overview of the installation process and show the installed UNIX directories and files.

**Note:** Directory and file names are for example purposes only.



## To Install Rational Rose RealTime on UNIX:

- 1 Log on to the install client. This may be any UNIX computer that:
  - Gives you access to a CD-ROM drive
  - Mounts the file system into which you will load the Rational Suite DevelopmentStudio RealTime (UNIX) release
  - Runs the operating system specified on the *Rational Suite DevelopmentStudio RealTime (UNIX)* CD (Solaris 2.6, 2.7, 2.8, or 2.9)
- 2 Place the *Rational Suite DevelopmentStudio RealTime (UNIX)* CD in the CD-ROM drive.

If the CD-ROM drive is not mounted, **mount the CD-ROM** drive.

As the root, create a directory (if one does not already exist) to be the mount point for the CD-ROM drive. The following examples for each platform use the directory /cdrom. Ensure that you know the device name of the CD-ROM drive. If you do not know the device name, consult your system administrator. Mounting commands for different operating systems are as follows:

▫ **Sparc/Solaris with Volume Management**

Solaris 2.x with volume management mounts to the /cdrom directory. This happens automatically when you load the CD-ROM drive. You have volume management if the **vold** daemon is running on the system.

▫ **Sparc/Solaris (Solaris 2.x) Without Volume Management**

```
# mkdir /cdrom
# mount -r -F hsfs /dev/dsk/c0t6d0s0 /cdrom
```

- 3 From a shell window, change directory to the root level of the mounted CD-ROM device. For example: `cd /cdrom`, and press **ENTER**.
- 4 To run the setup script, type the following:

### **rs\_install**

The **rs\_install** command is a complete installer that includes licensing setup, license checking, product installation, and product setup. Rational recommends that you follow the menus and prompts and allow **rs\_install** to guide you through the installation.

**Note:** You can invoke **rs\_install** with a number of options. For example, you can use the **-no\_log (-nl)** option to stop **rs\_install** from creating a log file. To see a listing of all available options, run **rs\_install -help**.

The **Using RS Install** script appears.

- 5 Press **ENTER** to continue.

In the **Enter Install Location** script, the installation process searches for Rational directories.

- 6 Press **ENTER** to continue.

An arrow (- - >), opposite a number/directory, indicates the default location used for this installation.

Next, you will specify the directory to install Rational Suite DevelopmentStudio RealTime (UNIX).

- 7 Type **0** to specify a new directory, or type a value associated with a listed directory, then press **ENTER**.

If you specify a new directory, `rs_install` copies the Rational files to this location. The directory name must be specified as an absolute path name, and must be a valid path (this means that the directory must exist). A **roseRT** sub-directory is appended in the directory that you specify. The directory needs to be visible on all computers from which you want to run this product, and must be writable by the installer's user name.

Next, the license agreement appears and you are prompted to accept or reject the license agreement. You must accept the license agreement to proceed.

- 8 *Type **Y** and press **ENTER** if you agree with the terms of the agreement.*

If you do not agree with the terms of the license, the installation should be aborted. All software and documentation should be returned to Rational Software.

- 9 Type **Y** or **N** to indicate if you want to **Show this license agreement next time**.

- 10 In the **Product and License Configuration** menu, type the number associated with **Rational Rose RealTime for UNIX**, then press **ENTER**.

- 11 In the **Rational Rose RealTime - Licensing Options Menu**, select a licensing option.

Option	Description
1	Use an existing Rational license (FLEXlm) file or a server that is already configured.
2	<b>Set up permanent or counted license(s).</b> <ul style="list-style-type: none"><li>▪ Request Node-Locked or floating keys through <b>AccountLink</b>.</li><li>▪ After you request Node-Locked key(s) from <b>AccountLink</b>, you will receive an email from Rational that contains an attachment (a .upd file). You must save this file.</li></ul>
3	<b>Set up a temporary license file.</b>

Depending on the licensing option you select, answer the questions and follow the directions.

- 12 After licensing, on the **Rational Rose RealTime - Product Customization Menu**, verify that Rational Rose RealTime for Unix will be installed, and that you have enough space to install it.
- 13 Press **f** (the default) to continue.
- 14 In the **Install Documentation Menu**, specify whether you want other documentation installed
- 15 In **Rational Rose RealTime - Enter Install Mode**, indicate how you want **rs\_install** to deal with components that are already installed.
- 16 Press **ENTER** to continue.  
**rs\_install** installs Rational Rose RealTime.
- 17 After the installation completes, press **ENTER** to continue.

## After You Install

---

After you install, you want to:

- *Sourcing to the Setup Script* on page 58
- *Unmounting the CD-ROM Drive* on page 58
- *Setting the Connexis Variable* on page 61
- *Verifying the Connexis Installation* on page 61

### Sourcing to the Setup Script

After you install Rational Rose RealTime, you should **source** to your `<rational_dir>` to automatically set your environment variables.

- For Rational Suite DevelopmentStudio, type the following:

```
source <rational_dir>/rs_setup.csh or .<rational_dir>/rs_setup.sh
```

- For the Rational Rose RealTime point product, type the following:

```
source <rational_dir>/rosert_setup.csh or  
. <rational_dir>/rosert_setup.sh
```

### Unmounting the CD-ROM Drive

For CD-ROM installs, unmount the CD-ROM drive with the following commands.

For Solaris with volume management (**bold** is running):

```
% eject cd
```

All others must unmount the CD as *root*.

```
% su  
# umount /cdrom
```

**Note:** You cannot eject the CD if you are at the directory `/cdrom` or `/cdrom/cdrom0`. If you receive a "Device busy" error, change your directory location to a location other than the CD-ROM and repeat the above commands.

### ClearCase Workstation Setup

The following setup must take place on all workstations that will be accessing a VOB or view. For UNIX, this includes all machines that are view servers.

These steps will also need to be run on all machines that act as view servers for the ClearCase views used by Rational Rose RealTime. If you use ClearCase MultiSite, you will need to do this at all the sites where the VOBs containing the Rose RealTime elements are replicated.

You can determine which machines are view servers by typing:

```
cleartool lsview
```

in a command window. The second item on each output line indicates the machine name where the view server is running. For example, if you see the following line in the output of the **lsview** command:

```
myview \\mymachine\vws\myview.vws
```

then "mymachine" is the name of the machine where the view server for **myview** exists.

For further details, see your ClearCase administrator.

## Command Line Access to the Source Control Tool

For any user wishing to use Rational Rose RealTime's integration with ClearCase, **cleartool** must be accessible from the command prompt.

## Element type setup: type manager

The following steps are required for making ClearCase clients aware of the new element type.

### UNIX

Use the `$ROSERT_HOME/bin/$ROSERT_HOST/cc/mi_typeman` script to install the type manager in each ClearCase installation. To set up the extensions and tool mappings, the user executing the script must have write access to the following directories in the ClearCase installation:

```
/lib/mgrs  
/config/ui/icons  
/config/ui/bitmaps  
/config/magic
```

Use the following command line to set up the proper file extensions and tool invocations:

```
<ROSERT_HOME>/bin/<ROSERT_HOST>/cc/mi_typeman.sh install  
-server
```

## ClearCase Options

There are no options that need configuring for UNIX ClearCase.

## ClearCase Repository Setup

Each VOB must be set up to allow files of the new element type to be created. Follow the steps that apply to your platform below for each VOB that will be storing Rational Rose RealTime files.

### UNIX

Use the `$ROBERT_HOME/bin/$ROBERT_HOST/cc/mi_typeman` script to register the `rosert_unit` element type in each VOB using the following syntax:

```
<ROBERT_HOME>/bin/<ROBERT_HOST>/cc/mi_typeman.sh install  
-eltype -vob <vob_path>
```

### Test the Type Manager

To determine if the `rosert_unit` element type has been successfully registered in the VOB, perform the following command from a command prompt after changing to a directory contained in the VOB:

```
cleartool lstype -long eltype:rosert_unit
```

A listing of the type details will verify that it is correctly registered.

## Setting the TORNADO 2.0 Debugger Environment Variable

On Solaris, in order to use the TORNADO 2.0 debugger with Rational Rose RealTime, the TORNADO 2.0 shared libraries must be pointed to by the environment variable `LD_LIBRARY_PATH`. The required libraries are usually located at:

```
<path to TORNADO 2.0 installation  
directory>/tornado-2.0/host/sun4-solaris2/lib
```

## Setting Other TORNADO Environment Variables

If you are using Rational Rose RealTime with the Target RTOS, you may want to set the `ROBERT_TORNADO_OPTIONS` and `ROBERT_TORNADO_TIMEOUT` environment variables.

When the `ROBERT_TORNADO_OPTIONS` is set to `VX_FP_TASK`, the application checks if the processor supports floating point, and if so, it runs the target with the floating point option.

When the ROSERT\_TORNADO\_TIMEOUT is set, this is used for the WTX request timeout. The timeout value is in milliseconds.

**Note:** When downloading a VxWorks module to target, by default the timeout for WTX commands is set to 30 seconds (ROSSERT\_TORNADO\_TIMEOUT=30000). This may not be enough when downloading large modules or when using a slow network. You may get an error if the timeout is exceeded. This variable allows you to increase the timeout period.

## Setting the Connexis Variable

After you install the Rational Rose RealTime, you must set the environment variable for CONNEXIS\_HOME to the appropriate location, such as:

```
setenv CONNEXIS_HOME $ROSSERT_HOME/Connexis
```

**Note:** Set this environment variable after \$ROSSERT\_HOME is created (either by setenv ROSSERT\_HOME or in a rs\_install setup after you source <rational\_dir>/rosert\_setup.csh, or . <rational\_dir>/rosert\_setup.sh

## Verifying the Connexis Installation

To increase efficiency and eliminate improper installation and/or setup misconfiguration, you are strongly encouraged to verify your installation. Additionally, verify your Connexis installation by using the BasicTest model provided with Connexis in \$CONNEXIS\_HOME/Connexis. This model uses the CDM transport.

## Verifying your Installation using BasicTest

You can easily verify your installation by using the BasicTest model provided with Connexis in:

```
$ROSSERT_HOME/Connexis/C++/examples
```

## Host Configuration Installation Verification

The following instructions are for a Windows NT, Windows 2000, and Windows XP Pro setup with Microsoft Visual C++ 6.0. If you use Visual C++ 6.0, use the elements corresponding to VC++6.0.

on page 1-63 and on page 1-63 define the names that are applicable for each of the supported host platforms (Windows and Solaris). The package names, component names, and component instance names differ for each platform.

Use the information in on page 1-63 and on page 1-63, when completing the following steps:

- 1 Start Rational Rose RealTime.
- 2 Load the BasicTest model from \$ROSERT\_HOME/Connexis/C++/examples.
- 3 From the **Component View**, expand the component package corresponding to your host platform.
- 4 Select the client component and from its item menu, choose **Build > Rebuild All** to recompile it.
- 5 Select the server component and from its item menu, choose **Build > Rebuild All** to recompile it
- 6 In the **Deployment View** NT40 package, expand the processor that corresponds to you host platform.
- 7 The client will use port 9100 and the server will use port 9900. If these ports are being used by other another application on your workstation, you will need to change them. Open the server component instance's specification sheet and change the 9900 in the **-CNXep** startup parameter to an available port number. Open the client **Component Instance Specification** dialog box and change 9900 specified in the **-s** argument to the server's port number. Change the 9100 in the **-CNXep startup** parameter to an available port number, and then save your changes.
- 8 Select the server component instance and click **Run**. On the **RTS** panel of the instance, click **Start** to execute the server.
- 9 On the Model View, select the client component instance and choose **Run** from its item menu. On the Runtime View of the instance, click the **Start** button to execute the client.



10 Verify that your output for client and server looks similar to the output shown in sections *BasicTest Server Output* on page 63 and *BasicTest Client Output* on page 64.

**Table 6 Components for Referenced Configurations**

	Component Package	Client Component	Server Component
WindowsMS Visual Studio 6.0	NT40-x86-VCC60	basicTestClient_31	basicTestServer_31
SolarisGnu 2.81	SUN5-sparc-gnu-281	basicTestClient_5	basicTestServer_5
SolarisSun Workshop 5.0	SUN5-sparc-SunCC-50	basicTestClient_4	basicTestServer_4

**Note: Note:** This list may be incomplete. Please check the BasicTest model for the complete list.

**Table 7 Component Instances for referenced configurations**

	Component Package	Client Component	Client Component Instance	Server Component Instance
WindowsMS Visual Studio 6.0	NT40	MyNT40Workstation	basicTestClient_31Instance	basicTestServer_31Instance
SolarisGnu 2.81	Solaris	MySparcstation	basicTestClient_5Instance	basicTestServer_5Instance
SolarisSun Workshop 5.0	Solaris	MySparcstation	basicTestClient_4Instance	basicTestServer_4Instance

**Note:** This list may be incomplete. Please check the BasicTest model for the complete list.

## BasicTest Server Output

```
Rational Rose RealTime C++ Target Run Time System
Release 6.50.B.00 (+c)
Copyright (c) 1993-2003 Rational Software
rosert: observability listening at tcp port 30399
```

\*\*\*\*\*

\* Please note: STDIN is turned off. \*

```
* To use the command line, telnet to the above mentioned port. *
* The _output_ of any command will be displayed in _this_ window. *
*****
```

```
Rational Software Corp. Connexis(tm) - Distributed Connection
Service (dcs)
Release 6.50.B.82
Copyright (c) 1999-2003 Rational Software Corporation
```

```
dcs: CRM Transport : enabled
dcs: CDM Transport : enabled
dcs: CRM listening at [crm://192.139.251.167:2005]
dcs: CDM listening at [cdm://192.139.251.167:9900]
dcs: target agent enabled
dcs: locator service not available
dcs: metric service enabled
```

```
BasicTest-Server-started:
```

```
Server : Received simple greeting message... sending it back
```

```
Server : test cycle completed, received rtunbound !
```

```
Server : Received simple greeting message... sending it back
```

```
Server : test cycle completed, received rtunbound !
```

**Note:** The above represents a partial listing of the BasicTest Server Output.

## BasicTest Client Output

```
Rational Rose RealTime C++ Target Run Time System
Release 6.50.B.00 (+c)
```

Copyright (c) 1993-2003 Rational Software

rosert: observability listening at tcp port 30380

```
*****  
*           Please note: STDIN is turned off.           *  
*   To use the command line, telnet to the above mentioned port. *  
* The _output_ of any command will be displayed in _this_ window. *  
*****
```

Rational Software Corp. Connexis(tm) - Distributed Connection  
Service (dcs)

Release 6.50.B.82

Copyright (c) 1999-2003 Rational Software Corporation

BasicTest-Client-started:

dcs: CRM Transport : enabled

dcs: CDM Transport : enabled

dcs: CRM listening at [crm://192.139.251.167:2010]

dcs: CDM listening at [cdm://192.139.251.167:9100]

dcs: target agent enabled

dcs: locator service not available

dcs: metric service enabled

Client : sending a greeting message...

->Client: received message:

RTString"Hello, Welcome to the Connexis world!"

Client : unbound received

```
Client : reregistering SAP

Client : sending a greeting message...

->Client: received message:
RTString"Hello, Welcome to the Connexis world!"

Client : unbound received
```

**Note:** The above represents a partial listing of the BasicTest Client Output.

## Starting Rational Rose RealTime (UNIX)

To start Rational Rose RealTime, run the command displayed at the end of the rs\_install process.

**Note:** The installation process creates a rosert\_setup.csh or a rosert\_setup.sh.

# Converting Connexis Models

# 5

## Contents

This chapter is organized as follows:

- *Converting Connexis version 2000.02.10 Models to Connexis Version 2003.06.00 Models on page 67*
- *Verifying Component Compatibility on page 70*

This chapter describes how to convert a model from Rational Connexis version 2000.02.10 to Rational Connexis version 2003.06.00.

## Converting Connexis version 2000.02.10 Models to Connexis Version 2003.06.00 Models

---

If you are using version 2000.02.10 of Connexis, the Connexis Model Conversion Tool searches your model, identifying any incompatibilities, and provides a detailed description, explaining the changes. Table 8 explains the changes made to your model during the conversion process.

**Note:** If you encounter any problems with migrating your model to Connexis version 2003.06.00, contact Rational Customer Support.

**Table 8 Model Conversion for Connexis version 2000.02.10 to Connexis version 2003.06.00**

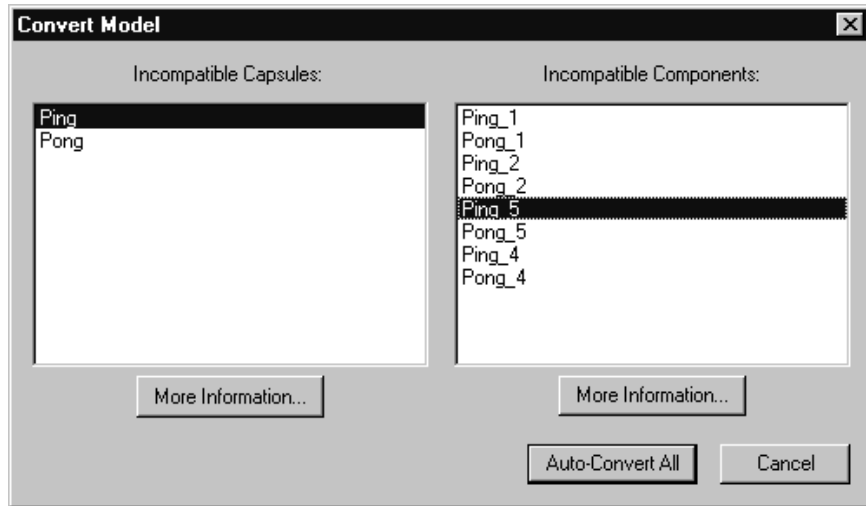
Condition	Change
RTDXBase, RTDXBase_Agent, RTDXBase_Locator, RTDXBase_Agent_Locator fixed capsule roles are in the model	Replaces the capsule roles with the corresponding RTDBase configuration. Integrates the CDM transport with the capsules containing the new RTDBase or RTDBase_Locator capsule roles. Integrates the CRM transport into the containing capsule.
RTDBase, RTDBase_Locator fixed capsule roles are in the model but do not have the CDM transport as an attribute.	Integrates the CDM transport using a composite aggregation relationship into the capsules containing RTDBase or RTDBase_Locator capsule roles.

**Table 8 Model Conversion for Connexis version 2000.02.10 to Connexis version 2003.06.00**

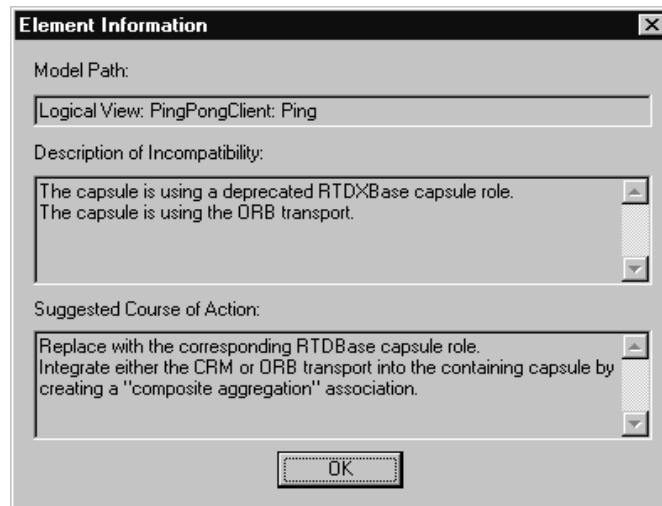
Condition	Change
RTDXBase optional capsule role is in the model	Converts to the RTDBase and integrates the CRM and CDM transports.
RTDXBase_Agent optional capsule role is in the model	Converts to RTDBase_Agent and integrates the CRM transport.
RTDXBase_Locator optional capsule role is in the model	Converts to RTDBase_Locator and integrates the CDM and CRM transports
RTDXBase_Locator_Agent optional capsule role is in the model	Converts to RTDBase_Locator_Agent and integrates the CRM transport.
RTDBase or RTDBase_Locator optional capsule role is in the model	Users are notified that the CDM transport is integrated.
RTDBase_Agent or RTDBase_Locator_Agent optional capsule role is in the model	Searches the model identifying any of the components that have a dependency on the ORB. If the dependency exists, the CRM transport is integrated.
A component depends on a XDCS library component	Changes the component dependency to use the DCS library component.
The TargetConfiguration property of a component references a -CNX-M or a -CNX-target configuration	Removes the -CNX- or -CNX-M from the TargetConfiguration name.

**To convert your model:**

- 1 Load a model that uses version 2000.02.10 of Connexis in Rational Rose RealTime.
- 2 Select **Tools > Connexis > Convert Model**.



- 3 View more information about incompatible capsules and components by selecting the capsule or the component from the dialog and clicking **More Information**.



The **Element Information** dialog provides the following information:

**Table 9 Element Information Dialog Chart**

Information Heading	Description
Model Path	Shows the path of the selected capsule or component.
Description of Incompatibility	Explains the reason for the incompatibility between version 2000.02.10 and version 2003.06.00.
Suggested Course of Action	Explains how the Conversion tool will make the capsule or component compatible with Connexis version 2003.06.00.

- 4 Click **OK** after you read the information, and repeat step 3 for additional capsules and components that appear in the **Convert Model** dialog.
- 5 Click **Auto-Convert All** from the **Convert Model** dialog.

The Conversion Tool converts the incompatible capsules and components in your model. As the conversion takes place, the Conversion Tool may prompt you to confirm some conversion changes.

**Note:** Only run the conversion tool once. If you run the tool a second time, the information displayed in the **Convert Model** dialog may not be accurate.

## Verifying Component Compatibility

---

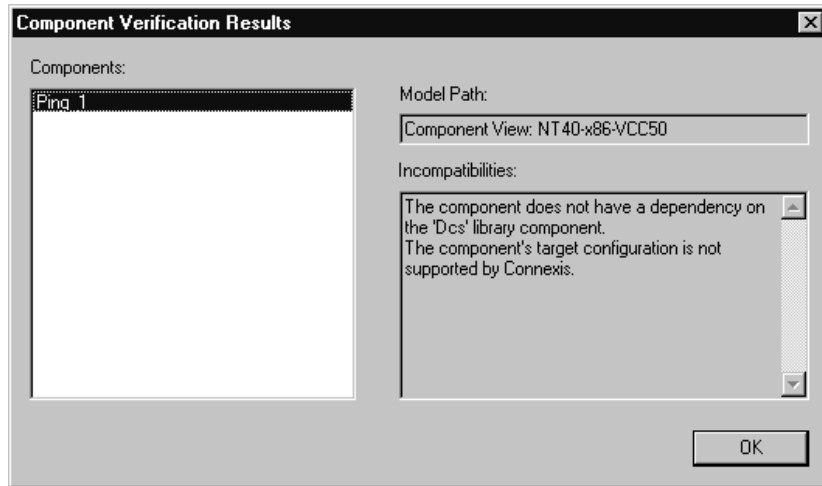
The Component Verification Tool verifies that a component is compatible with Rational Connexis version 2003.06.00.

**To verify that a component is compatible with version 2003.06.00:**

- 1 Right-click a component.
- 2 Select **Connexis > Verify**.



The **Component Verification Results** dialog appears.



- 3 Select the component from the **Components** area.

The model path and the incompatibilities for the selected component appear.

- 4 Open the component from the browser and fix the incompatibilities.

**Note:** You do not have to close the Component Verification Results dialog while fixing the incompatibilities.

- 5 Click **OK**.



## Contents

This chapter is organized as follows:

- *How Licenses Work* on page 73
- *Types of Licenses* on page 74
- *Requesting License Keys* on page 76
- *Converting a Temporary License to a Permanent License* on page 78
- *The License Manager - UNIX* on page 79
- *License Manager Commands* on page 80
- *Licensing on UNIX* on page 84
- *The License File* on page 85
- *UNIX Licenses* on page 87
- *Frequently Asked Questions* on page 88

When you buy Rational Rose RealTime, you purchase a number of node-locked and/or floating licenses. A node-locked license allows you to use Rational Rose RealTime on a specific workstation. Floating licenses allow anyone on your network to use Rational Rose RealTime as long as a floating license is available. Thus, the number of licenses that you purchase determines the maximum number of users who can use Rational Rose RealTime simultaneously.

For example, if you purchased five licenses and three users are currently using Rational Rose RealTime, then two more users can use Rational Rose RealTime.

## How Licenses Work

---

Licenses are managed by a *license manager* (FLEXlm™ software delivered as part of Rational Suite DevelopmentStudio for UNIX) that runs on a *license server*. The license manager monitors license access, simultaneous usage, idle time, and so on.

When you start Rational Rose RealTime from the Rational Suite DevelopmentStudio, you are initially unlicensed. If a license is available, the license manager gives you a license for the Suite, which allows you to run any of the products included in the

Suite. You retain the license as long as you keep using any of the products in the Suite. When you exit the last program in the Suite, your license is returned to the license manager and is made available for another user.

If no license is available, you are unable to use Rational Rose RealTime until a license is returned by another user. An "Unable to obtain a license" message is displayed.

**Note:** The inability to obtain a license may also be caused by a corrupted license file, a change to the host id (network card, IP address) or a hard disk drive replacement when a node-locked license is used on NT. Please ensure you are able to communicate with the license server through a simple ping command. For example:

```
ping <IP address of license server>
```

## Types of Licenses

---

The types of licenses are:

- *Node-Locked Licenses* on page 74
- *Floating Licenses* on page 74
- *Permanent Licenses and Temporary License Keys* on page 75
- *Emergency and Evaluation Keys* on page 75

### Node-Locked Licenses

Node-locked licenses are created only for a specific system. A node-locked license can be a permanent license, a temporary license, or it can be an evaluation license.

**Note:** Because node-locked licenses are uncounted licenses, there is no need to have a license server process running to manage their use.

### Floating Licenses

Floating licenses are licenses that can be shared by multiple users on multiple systems. A Rational license server controls use of the floating licenses.

Floating licenses allow anyone on your network to use Rational Suite DevelopmentStudio as long as a license is available. Thus, the number of licenses that you purchase determines the maximum number of users who can use Rational Suite DevelopmentStudio concurrently.

## Permanent Licenses and Temporary License Keys

When you register Rational products to specific systems (license server or client) in AccountLink, Rational generates license keys and sends you an e-mail message with these permanent license keys in a license file. The permanent keys let you use the Rational products have no expiration date. However, Rational assigns an expiration date to the license keys if your company has negotiated a Term License Agreement (TLA). TLA keys are not permanent, but the process of ordering and installing TLA licenses is the same as a permanent license.

To use Rational products for an evaluation period or if you expect a delay in receiving your permanent keys, you can install the temporary license key provided in your Rational License Key Certificate. Because Rational has not generated the temporary key for a specific system, you can use it on any system until the specified expiration date.

Permanent and temporary license keys can be floating or node-locked. The difference is that a temporary key is not generated for a specific system and a permanent key is generated for a specific system.

## Emergency and Evaluation Keys

Emergency and evaluation license keys are temporary license keys. They can be floating or node-locked. They are short-term licenses that are not generated for a specific system.

## Suite Licenses and Point Product Licenses

A Rational license key indicates whether it is a Rational Suite license, such as Rational Suite DevelopmentStudio, or a point-product license, such as Rational Purify. A Rational license file can contain multiple floating or node-locked Suite and point-product license keys.

## Returning License Keys

You may need to replace an old system or decide another system should act as the new Rational license server. Because permanent license keys are tied to a system's host ID, Rational products will not work on another system until you import new license keys that are tied to the new system's host ID.

To get your new license key, you need to "return" the existing license key back to your Rational account and then "get" or order a license key for the other system. You could also call this task moving the license key from one system to another or removing the license key from the old system.

When you return a license key, you do not physically give the license key back to Rational. Instead, the return transaction updates Rational's records to indicate that you are no longer using the software on that system. This adjusts the count of registered products in your account and allows you to get the license key for the other system.

In accordance with the Legal Agreement provided on AccountLink, you have 30 days to shut down the license server that corresponds to the server identified in the returned license file. If you have a license file that contains more than one license and you are returning only one of those licenses, remove the entry for the license that you are returning. When you have finished editing the file, use the **Imreread** command to reread the license file and restart the vendor daemon. For more information about licensing commands, see *License Manager Commands* on page 80.

## Upgrading Licenses

If you are upgrading from an earlier version of a Rational Suite or point-product, you can reuse your current Rational Suite and point-product license keys.

## Requesting License Keys

---

AccountLink (<http://www.rational.com/accountlink>) is a Web tool that you can use to manage your permanent (or Term License Agreement) license keys. To use AccountLink, you need the License Key Certificate to order and install your license keys. AccountLink's interface offers three license transactions:

- Get License Key(s)
- Return License Key(s)
- Request a Copy of a License File

With these three transactions, you can order and return permanent license keys for Windows and UNIX products from single or multiple Rational accounts.

**Note:** AccountLink does not support temporary license key transactions.

AccountLink requires you to register your Rational software to specific systems using the system's host ID or ethernet address. You can register:

- Rational Windows or UNIX products that will be served from a Rational license server.
- Single or redundant Rational license servers on Windows or UNIX systems.

- Remote Windows or UNIX systems; you do not need to sit at the system for which you are requesting license keys.

If you are not at the computer for which you are requesting license keys, you must have the following information available: *Hostname* and Host ID. You can download a tool from AccountLink that provides this information automatically for you.

Alternatively, you can run `rs_hostinfo` directly from the CD to get the host information. This applies to UNIX host information only. To obtain information about a Windows host, you need to use the download tool.

The license key types for Rational Rose RealTime that are supported in Rational Suite DevelopmentStudio are:

Component type	License type
Rational Rose RealTime for UNIX	Node-locked and floating
Rational Rose RealTime for Windows	Node-locked and floating

## Receiving and Importing License Keys

After you register your Rational products to a specific system with AccountLink, Rational generates a license key file that contains the license key. The file is sent in an e-mail message to the contact e-mail address that you designate in AccountLink's License Contact page.

You need to save the file to a known directory location as you will need to provide this information when you install the Rational software.

**Note:** If AccountLink is unavailable, see *Requesting License Keys by Fax* on page 77 or call Rational Licensing Support. See *Contacting Rational Customer Service by Email or Telephone* on page 166 for Support phone numbers.

## Requesting License Keys by Fax

This section summarizes the steps for getting a node-locked or floating permanent license key when you do not have an internet connection or when Rational AccountLink is unavailable.

Although this section gives customers instructions for obtaining license keys by fax, Rational recommends that you use Rational AccountLink ([www.rational.com/accountlink](http://www.rational.com/accountlink)) to request permanent license keys.

### To request license keys by fax:

- 1 Find your License Key Certificate in your Rational product shipment.
- 2 Print the license request form.

The documentation browser can be used directly from the CD-ROM and from the installed product area. To view the form directly from the CD-ROM, run the command **rs\_help** from the CD-ROM root directory. The form is located in the HTML Tool Documentation/Rational Suite DevelopmentStudio/FAX License Request Form.

- 3 Use the License Key Certificate to fill out the form. Make sure that the contact, Rational account number, product, licensing, and host information are correct. Any errors will cause delays in receiving your license keys.

**Note:** If you are requesting a node-locked license, be sure to select **NodeLocked** and not **NodeLocked UNIX**.

- 4 Fax the request to Rational. See *Contacting Rational Customer Service by Email or Telephone* on page 166 for fax and phone numbers.

Call Rational Licensing Support if you cannot use Rational AccountLink or the fax form to order your permanent license keys. See *License Support Contact Information* on page 167 for phone numbers.

## Receiving Permanent License Keys

If you request a new license using AccountLink, Rational will send you a license key file through email. If you request a permanent license key by fax and you have specified an email address in your contact information, you will receive a license key file through email. You can copy the permanent license file from the email enabled system and install it on the system that is not e-mail enabled.

If you cannot provide an email address, contact Rational Licensing Support. See *License Support Contact Information* on page 167 for the phone numbers.

## Converting a Temporary License to a Permanent License

---

If you initially used a temporary license (evaluation or startup) to install Rational Suite DevelopmentStudio, you can convert your license to a permanent license by using the **license\_setup** command. The **license\_setup** command allows you to run a subset of the install script, **rs\_install**. The **license\_setup** command allows you to set up license options and run the license check sequence.



You may also do this by running `rs_install`; however, using `license_setup` will save you time as there is no need to run through a full product installation or any of the post product installation setup.

**Note:** You need to have a permanent or TLA license before you start. See *Requesting License Keys* on page 76.

## Licenses for Windows

---

You can request and install license keys before or after installing Rational products; however, you must have a license key installed and configured to run Rational Rose RealTime. To configure a license key, click Configure Licenses to launch the Rational License Key Administrator and License Key Administrator Wizard. If you do not install the license keys before installing, the License Key Administrator will appear and the end of the installation process.

The Rational Suite License Management Guide describes the licensing terms and the Rational License Key Administrator.

## The License Manager - UNIX

---

Rational Suite DevelopmentStudio for UNIX uses the Flexible License Manager, FLEXlm™, from Globetrotter Software, Inc. The DevelopmentStudio requires FLEXlm 7.0f. The license manager includes the following components:

- A *vendor daemon* named **rational** that dispenses DevelopmentStudio licenses.

The **rational** daemon is used for all of Rational's licensed products. If you have other products from other vendors that also use FLEXlm, they will include their own vendor daemons.

- A *license daemon* named **lmgrd**.

The same license daemon is used by all licensed products from all vendors that use FLEXlm. The **lmgrd** daemon does not process requests on its own, but forwards requests to the appropriate vendor daemon.

- A *license file* that you maintain.

It specifies your license servers, vendor daemons, and product licenses.

**Note:** Rational recommends that you use a single combined license file for all of our products.

After the license file is in place and the license daemons are running, the server system needs to be set up to automatically restart the license server when it reboots. You will be instructed by `rs_install` or `license_setup` how to do this. These commands cannot do this because this step requires root permissions. The commands to do this are as follows:

On Solaris:

```
$ su
# cp <rational_dir>/config/start_lmgrd_on_server-name \
    /etc/rc2.d/S98Rational
```

## License Manager Commands

---

To verify that your license manager is operational, you can enter these commands on your license server to see if its daemons are running:

```
% ps axw | grep -v grep | egrep "lmgrd|rational"
```

or

```
% ps -e | grep -v grep | egrep "lmgrd|rational"
```

Their output should include lines similar to the following (your path names may vary):

```
538 ?? S 0:03.50 /rational/base/cots/flexlm.7.0f/platform/lmgrd
    -c /rational/config/servername.dat
    -l /rational/config/servername.log
539 ?? I 0:00.90 rational -T brazil 6.0 3 -c ...
```

The license manager supports several system-administration commands.

Command	Description
lmdiag	Allows you to diagnose problems when you cannot checkout a license.
lmdown	Shuts down license and vendor daemons
lmhostid	Reports license manager host ID of workstation
lmreread	Rereads license file, starts new vendor daemons
lmstat	Reports status on daemons and feature usage
exinstal	Reports on licenses in license file you specify on the command line.

For more information on these commands, you can view the FLEXlm online documentation in the `rational_dir/docs/html/FLEXlm_End-User_Manual` directory. This documentation is in HTML format.

## Additional Licensing Commands

**license\_check** - This command allows you to run a subset of **rs\_install**. In addition to using the commands above, you can also use the **license\_check** command to run the FLEXlm **lmstat** command for counted licenses and the **exinstal** command for any license file (not `port@host`). The **lmstat** command queries the license server for a list of licenses that are in the license pool. The **exinstal** command checks the license file format and license codes to see if everything is consistent.

## License Manager Daemon (lmgrd)

The *license manager daemon* (**lmgrd**) handles the initial contact with the client application programs, passing the connection on to the appropriate vendor daemon. It also starts, stops, and restarts the vendor daemons.

## Vendor Daemon

In FLEXlm, licenses are granted by running processes. There is one process for each vendor who has a FLEXlm-licensed product on the network. This process is called the *vendor daemon*. The vendor daemon keeps track of how many licenses are checked out, and who has them. If the vendor daemon terminates for any reason, all users lose their licenses. (This does not mean that the applications suddenly stop running. Users

can save their work and exit safely.) Users normally regain their license automatically when `lmgrd` restarts the vendor daemon, although the applications may exit if the vendor daemon remains unavailable.

Client programs communicate with the vendor daemon usually through TCP/IP network communications. The client application and the daemon processes (the license server) can run on separate nodes on your network across any size wide-area network. Also, the format of the traffic between the client and the vendor daemon is machine independent allowing for heterogeneous networks. This means that the license server and the computer running an application can be on different hardware platforms or even different operating systems (for example, Windows NT as a server system and UNIX as a client or UNIX as a server and Windows NT as a client).

## License Key File

Licensing data is stored in a text file called the *license key file*. The license key file is created by the software vendor and is edited and installed by the License Key Administrator. It contains information about the server nodes and vendor daemons, and at least one line of data (called FEATURE or INCREMENT lines) for each licensed product. Each FEATURE line contains a license key based on the data in that line, the *hostids* specified in the SERVER lines, and other vendor specific data.

In some environments, you can combine the licensing information for several vendors into a single license key file. The FLEXlm default location is:

```
/usr/local/flexlm/licenses/rational.dat (UNIX)
```

**Note:** We strongly recommend that you keep a copy of the license key file in a safe location.

## Application Program

The application program using FLEXlm is linked with the program module (called the FLEXlm client library) that provides communication with the license server. On Windows, this module is called `LMGRxxx.DLL`, where *xxx* indicates the FLEXlm version. During execution, the application program communicates with the vendor daemon to request a license.

## Configuring a UNIX Workstation to Point to a FLEXlm Server

To configure a UNIX workstation to point to a FLEXlm server, point to a copy of the license file on the UNIX client computer. You can make a copy of the license file if you cannot see it from the client computer.

Use the following command to help debug problems on the UNIX client computer:

```
$ROSERT_HOME/bin/sun5/lmstat -c $ROSERT_LICENSE_FILE
```

## License Activation Process

When you run a "counted" FLEXlm-licensed application, such as a Rational Suite product that uses a floating license, the following occurs:

- 1 The license module in the client application finds the license key file, which includes the host name of the license server node and port number of the license manager daemon, **lmgrd**.
- 2 The client establishes a connection with the license manager daemon (**lmgrd**) and specifies the appropriate vendor daemon.
- 3 **lmgrd** determines which machine and port correspond to the master vendor daemon and returns that information to the client.
- 4 The client establishes a connection with the specified vendor daemon and sends its license request.
- 5 The vendor daemon checks in its memory to see if any licenses are available and sends a grant or denial back to the client.
- 6 The license module in the application grants or denies use of the feature, as appropriate.

"Uncounted" features, where the number of licenses is '0' (zero), do not require a server and the FLEXlm client library routines in the application grant or deny usage based solely upon the license contents. Node-locked licenses, for example, set the license number to 0 (zero).

# Licensing on UNIX

---

## Running the lmgrd from a Command Prompt

From a command prompt execute:

```
lmgrd -c <licenseFileList> -l <logfile>
```

**Note:** lmgrd can be found in \$ROSE\_HOME/bin/<arch>, where <arch> is the host that Rational Rose RealTime is installed on (sun5).

- **licenseFileList** is the path to the license file or a list of license files. If the FLEXlm daemon is only being used to provide Rational Rose RealTime licenses, use -c \$ROSE\_LICENSE\_FILE. Otherwise, include the \$ROSE\_LICENSE\_FILE environment variable in a semicolon (“;”) separated list.
- **logfile** is the path to a log file. \$ROSE\_HOME/license/log is recommended if lmgrd is only providing Rational Rose RealTime licenses.

For convenience, you will probably want to augment a system initialization script on your license server to automatically start the license daemon each time the license server boots.

The names, locations, organization, and contents of system initialization scripts varies from UNIX system to UNIX system. You might begin by looking at the following files: for Solaris:

```
/etc/rc2.d/SlmRational.sh
```

To verify that your license manager is operational, you can enter these commands on your license server to see if its daemons are running:

```
% ps axw | grep -v grep | egrep "lmgrd|rational"
```

or

```
% ps -e | grep -v grep | egrep "lmgrd|rational"
```

## Example

```
lmgrd -c $ROSE_LICENSE_FILE -l /apps/logs/logRRT
```

or

```
lmgrd -c $ROSE_LICENSE_FILE;$SLM_LICENSE_FILE -l  
/apps/logs/current_log
```

## Administration Commands

The license manager supports several system-administration commands.

---

Command	Description
lmdiag	Allows you to diagnose problems when you cannot checkout a license.
lmdown	Shuts down license and vendor daemons.
lmhostid	Reports license manager host ID of workstation
lmremove	Returns specific licenses to license pool (for example, after a workstation crashes).
lmreread	Rereads license file, starts new vendor daemons.
lmstat	Reports status on daemons and feature usage.
exinstal	Reports on licenses in license file you specify on the command line.

---

**Note:** These commands can be found in `$ROSERT_HOME/bin/<arch>`, where `<arch>` is the host that Rational Rose RealTime is installed on (sun5).

## The License File

---

The default Rational license file is either:

```
<rational_dir>/config/rational.dat
```

or

```
<rational_dir>/config/temporary.dat
```

The **temporary.dat** file is used for both startup and evaluation licenses while the **rational.dat** file is used for permanent and TLA licenses.

FLEXlm uses this variable to locate the license file.

### Format

The license file is a text file that you can edit with any text editor. Your license file will contain lines similar to:

```
SERVER garcon 1874350 1706
DAEMON rational
FBE669014E142A4CF37 " "
```

In general, one or three server lines are followed by one or more vendor daemon lines, which are followed by one or more feature lines. Rational Rose RealTime requires only one of each, but your license file may include data for other products.

Each server line contains:

- Keyword SERVER
- Host name of the license server, from *hostname*
- License manager host ID of the license server, from *lmhostid*
- TCP port number to use

Each vendor daemon line contains:

- Keyword DAEMON
- Name of the vendor daemon (always rational for Rational Rose RealTime)
- Pathname to the directory that contains the executable code for this daemon
- Pathname to your options files for this daemon (optional)

Each feature line contains:

- Keyword FEATURE
- Name of the feature
- Name of the vendor daemon, previously defined on a DAEMON line, that serves this feature (always rational for Rational products)
- Latest (that is, highest number) version of this feature that is supported (5.000) for the current release of Rational Rose RealTime
- Expiration date. This is specified as 'dd-mmm-yy' or as 'dd-mmm-yyyy', where 'yy' is the last 2 digits of the year and 'yyyy' is the unabbreviated year. You must specify 4 digits for the year 2000 and beyond. You must specify '00' to indicate a license which does not expire.
- Number of licenses
- Encryption code (obtained from Rational for Rational Rose RealTime)
- Vendor string, enclosed in double quotes, contains node-locked information when licensing Rational Rose RealTime as node-locked
- License manager host ID, supplied only when this feature is bound to a specific host (that is, node-locked)

**Note:** You cannot combine floating and node-locked licenses for the same product in a single license file.



The tokens on each line can be separated by any amount of white space (spaces or tabs). You can edit only four kinds of tokens in the license file:

- Host names on SERVER lines
- TCP port numbers on SERVER lines
- *Pathnames* to vendor daemons on DAEMON lines
- *Pathnames* to options files on DAEMON lines

All other tokens are included as input to the encryption algorithm that produces the encryption codes on the FEATURE lines.

**Note:** A DEMO FEATURE Line (includes "DEMO" at the end of the FEATURE Line) is a special temporary license which does not require running `lmgrd` or `start_lm`. Licensing is activated when the DEMO FEATURE Line is placed in the license file.

## UNIX Licenses

---

The type of licenses are:

- Start-up or Emergency keys
- Node-Locked keys
- Floating keys
- TLA (Temporary License Agreement)

### Start-up or Emergency keys

#### Notes:

- Use `-startuplicense` to enter keys.
- When the UNIX LKAD displays, fill the fields with the information from Welcome letter.

### Node-Locked keys

#### Notes:

- Request Node-Locked keys through **AccountLink**.
- After you request Node-Locked key(s) from **AccountLink**, you will receive an email from Rational that contains an attachment (a .upd file). You must save this file.
- FLEXlm is not required for Node-locked licenses.

## Floating keys

### Notes:

- Request Floating keys through **AccountLink**.
- After you request Node-Locked key(s) from **AccountLink**, you will receive an email from Rational that contains an attachment (a .upd file). You must save this file to a desired location on the server.
- We strongly recommend that you keep a copy of this license file in a safe location.

**Note:** We strongly recommend that you keep all of your Rational Floating licenses in a single license file. Do not mix Floating and Node-Locked keys in the same file. Use `lmgrd -c <key_file1; key_file2; keyfile3>` to point to several different license files.

## TLA (Temporary License Agreement)

### Notes:

- Temporary license keys that are valid for a specified period of time.

## Frequently Asked Questions

---

- 1 Can I use the FLEXlm licensing software I already have installed?

Yes. Install our license code in the default location (in **rational\_dir/base/cots**) and use it to serve the Rational licenses.

- 2 I already have FLEXlm installed and managing non-Rational licenses, and now I want to install Rational Suite DevelopmentStudio for UNIX. Can I do this?

Yes. You can have more than one **lmgrd** on a system, but they must use different ports. You can only have one rational daemon on the system.

- a What do I do if my existing FLEXlm installation uses port 27000?

27000 is the default port, so you need to specify a different port number for DevelopmentStudio. Do this by editing the license import file (.upd file) and modifying the SERVER line. Change the port number to something other than 27000 (for example, 2001). Note that the port number follows the host ID.

- b What do I do if my existing FLEXlm installation uses a port other than 27000?

You don't have to do anything since `rs_install` will default to port 27000. If you are using the same server for other Rational products, you must specify the port number you are using.

## Contents

This chapter is organized as follows:

- *Before You Begin* on page 89
- *Installing a Startup or Permanent License on Windows* on page 89
- *Installing a Startup or Permanent License on UNIX* on page 92
- *Integration With Rational Suites Licensing* on page 95
- *Troubleshooting* on page 96

## Before You Begin

---

For specific information on license keys please refer to the Installation Instructions and License Certificate that accompany the product shipment. If either of these two documents is missing, please contact Rational License Support for replacement information.

Before you begin, ensure that you know the name of your license server. You will be prompted for the server name during the installation.

You can install Rational license keys before or after you install a Rational product. If you want to install a license key before you install a Rational product, open the Rational License Key Administrator by selecting **Programs > Rational Software > Rational License Key Administrator** from the Windows **Start** menu. Use the Rational License Key Administrator Help or see the *Administering Licenses for Rational Software* manual for information about requesting and installing license keys.

## Installing a Startup or Permanent License on Windows

---

The License Key Administrator (LKAD) lets you install startup or permanent license keys, as required. The startup license keys are time-limited and allow you to start using Rational Rose RealTime immediately.

### To obtain a license key:

- 1 Do one of the following:
  - To install a temporary license key, select the **Enter a Temporary or Evaluation License Key option**.
  - To obtain a permanent license key, select one of the other options.
- 2 Follow the prompts in the wizard after you have chosen your option.

If you choose **Request a license using Rational AccountLink on the World Wide Web**, your web browser opens and takes you to the AccountLink web site:

<http://www.rational.com/accountlink>

We recommend that you bookmark this site. You will need to access AccountLink when you are ready to obtain a permanent license.

## Installing a Permanent License on Windows

### To install a permanent license key:

- 1 Open the **Rational Rose RealTime AccountLink** web site:  
[www.rational.com/accountlink](http://www.rational.com/accountlink)
- 2 Click **Get License Key(s)**.  
AccountLink prompts you to enter your account information.
- 3 View your company's License Key Certificate and enter your Rational account number found on this certificate.  
**Note:** If you are unable to find your Rational account number, contact Rational License Support.
- 4 Click **Next**.  
AccountLink prompts you to specify the license type.
- 5 To select a license type, do one of the following:
  - Click **NodeLocked** to obtain a license for a client install.
  - Click **Floating** to obtain a license for a server install.
- 6 Select the product line **Rose RealTime**.
- 7 Select the product name **Rational Rose RealTime for Windows**.
- 8 Enter the required quantity of licenses.  
**Note:** For node-locked licenses, the quantity can only be "1".

**9 Click Next.**

AccountLink prompts you to enter your Host Name and Host ID.

**10 Enter your Host Name and Host ID.**

**11 If you do not know the host name and host id, you can download an application from AccountLink:**

- Select **Windows operating system** from the scroll down list.
- Click **Download**.

The **File Download** dialog box appears, prompting you to open the file from its current location or to save the file. We recommend that you open the file, to import it to disk automatically.

- Click **OK**.
- A dialog box appears containing the Host Name and Host ID.
- Copy the Host Name and Host ID from the dialog box.
- Paste the contents into the Host Name and Host ID fields.

**12 Select the platform on which the toolset will be running.**

**13 Click Next.**

**14 Enter the contact information.**

**15 Click Next.**

**16 Verify the information:**

- If the information is correct, click **Submit**.
- If the information is NOT correct, click **Modify email**. Correct the information as required, then click **Submit**.

**Note:** An email message will be sent to the inbox for the email address which you submitted.

## Installing the License Key

### To install the license key:

- 1 Double-click the attached .upd file.  
A dialog box appears prompting you to save the file to disk or open the file.
- 2 Click **Open** and then click **OK**.  
The **LKAD Confirm Import** dialog box appears.
- 3 Click **Import**, then click **OK**.

## Installing a Floating License Key on a UNIX server

### To install a floating license key on a UNIX server:

- 1 Obtain the license key as outlined in *Installing a Permanent License on Windows* on page 90.
- 2 Set the HOST NAME and HOST ID to be the UNIX LICENSE SERVER.
- 3 FLEXlm v7.0f or greater and the rational daemon are both required on the UNIX machine. If either of these is not available, they can be downloaded from our **ftp** site at:  
  
`ftp://ftp.rational.com/public/tools/flexlm`
- 4 Activate the new licenses with the FLEXlm software. For information about the FLEXlm license manager, see *The License Manager - UNIX* on page 79, or refer to the FLEXlm documentation.
- 5 Using the License Key Administrator, set your license server using the **Settings - Service Configuration** menu.

## Installing a Startup or Permanent License on UNIX

---

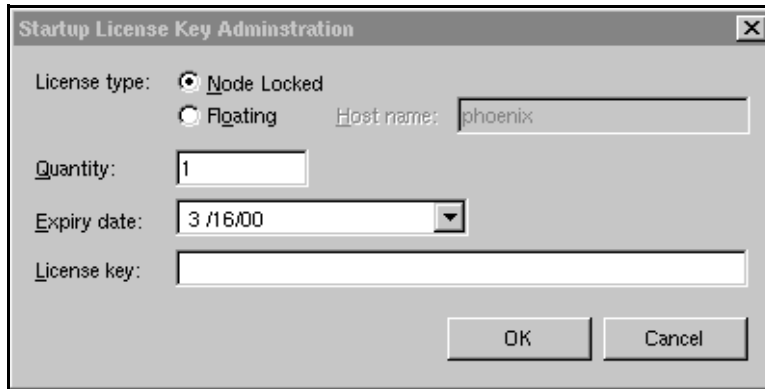
The startup license keys are time-limited and allow you to start using Rational Rose RealTime immediately.

### Installing a Startup License on UNIX

#### To install a startup license on UNIX:

- 1 Go to the \$ROBERT\_HOME/bin directory.
- 2 Type `RoseRT -startuplicense`.

The **Startup License Key Administration** form appears.



The screenshot shows a dialog box titled "Startup License Key Administration" with a close button (X) in the top right corner. The dialog contains the following fields and controls:

- License type:** Two radio buttons are present. The first is labeled "Node Locked" and is selected (indicated by a filled circle). The second is labeled "Floating".
- Host name:** A text input field containing the value "phoenix".
- Quantity:** A text input field containing the value "1".
- Expiry date:** A date selection field showing "3 /16/00" with a downward arrow on the right side.
- License key:** A large empty text input field.
- Buttons:** Two buttons are located at the bottom right: "OK" and "Cancel".

Locate the **Startup License Key** certificate that accompanied your product shipment.

- 3 Based on the license type and product name indicated on this certificate, copy the appropriate information into the **Startup License Key Administration** form, and click **OK**.

**Note:** A floating license requires you to start the license server. See *Understanding Rational Rose RealTime Licenses* on page 73.

Your startup license is created. Remember that your Startup license will expire on the date listed on the certificate. You will have to request and install permanent license keys before this expiry date.

Now you are ready to start Rational Rose RealTime.

## Installing a Permanent License on UNIX

Licenses are obtained from the Rational website, using AccountLink. After obtaining the license(s), they need to be installed on Rational Rose RealTime.

### To install a permanent license on UNIX:

- 1 Visit the Rational Rose RealTime AccountLink web site:  
[www.rational.com/support/accountlink](http://www.rational.com/support/accountlink)
- 2 Click **Get License Key(s)**.  
AccountLink prompts you to enter your account information.

- 3 View your company's License Key Certificate and enter your Rational account number found on this certificate.

**Note:** If you are unable to find your Rational account number, contact Rational License Support.

- 4 Click **Next**.

AccountLink prompts you to specify the license type.

- 5 To select a license type, do one of the following:
  - Click **NodeLocked** to obtain a license for a client install
  - Click **Floating** to obtain a license for a server install

- 6 Select the product line **Rose RealTime**.

- 7 Select the product name **Rational Rose RealTime for UNIX**.

- 8 Enter the required quantity of licenses.

**Note:** For node-locked licenses, the quantity can only be "1".

- 9 Click **Next**.

AccountLink prompts you to enter your Host Name and Host ID.

- 10 Enter your Host Name and Host ID.

- 11 If you do not know the host name and host id, you can download an application from AccountLink:

- Select **UNIX operating system** from the scroll down list.
- Click **Download**.

The **File Download** dialog box appears, prompting you to open the file from its current location or to save the file. We recommend that you open the file, to import it to disk automatically.

- Click **OK**.
- A dialog box appears containing the Host Name and Host ID.
- Copy the Host Name and Host ID from the dialog box.
- Paste the contents into the Host Name and Host ID fields.

- 12 Select the platform on which the toolset will be running.

- 13 Click **Next**.

- 14 Enter the contact information.

- 15 Click **Next**.



**16** Verify the information:

- Click **Submit** if the information is correct.
- Click **Modify email** if the information is NOT correct. Correct the information as required and then click **Submit**.

**Note:** An email message will be sent to the inbox for the email address which you submitted.

## Installing the License Key

**To install the License Key:**

- 1** Save the attached .upd file as: `$ROSERT_HOME/license/license.dat`
- 2** Do one of the following:
  - To integrate Rational Rose RealTime with other Rational products, see *Integration With Rational Suites Licensing* on page 95.
  - To not integrate Rational Rose RealTime with any other Rational products, see *The License Manager - UNIX* on page 79, to initially set up FLEXlm and activate your new keys.

## Integration With Rational Suites Licensing

---

If you are using other Rational products with Rational Rose RealTime, the license.upd file that you receive from Rational in response to a license request will contain the keys for all the Rational products. If you are using floating licenses, you will already be using the FlexLM **lmgrd** daemon and the rational vendor daemon.

Rational Rose RealTime assumes that the `ROSERT_LICENSE_FILE` variable points to a valid FlexLM license file that contains a valid Rational Rose RealTime license. If you follow the instructions provided, the existence of the additional license keys will not cause any problems.

**Note:** Only one instance of the rational daemon can be executed at any given time for floating licenses. Your project's license administrator should ensure that only one instance of the rational command exists and/or all paths are set correctly so that only one instance of the rational command is used.

For additional information on integration with Rational Suites Licensing, see the *Installing Rational Suite Guide*.

# Troubleshooting

---

You may encounter some difficulties with the following configurations:

- Windows
- UNIX server
- UNIX

## Windows

### Problem 1

If a FLEXlm License Manager dialog appears indicating that "Your application was unable to obtain a license because...", do the following:

- 1 Click **Cancel**.

You will get a Rational Rose RealTime message stating "Unable to obtain a license".

- 2 Click **OK**.
- 3 Run the **LMTools** application, located in:  
C:/Program Files/Rational/CommonLM
- 4 Verify that **FlexLM** is pointing to the correct license file.

### Problem 2

If you receive an "Unable to obtain a license message" message after the splash screen is displayed, check the expiration date of your license.

### Problem 3

If you have received a floating license file and are unable to obtain a license, verify that the license daemon is running. See *Installing a Floating License Key on a UNIX server* on page 92.

## UNIX server

**Note:** This section applies only if you are installing a floating license on a UNIX server.

### Problem 1

If a FLEXlm License Manager dialog appears indicating that "Your application was unable to obtain a license because...":

- 1 Ensure that your Windows client environment variable for ROBERT\_LICENSE\_FILE is set to the appropriate location.
- 2 Ensure that your UNIX server is set up correctly. For information on setting up your UNIX server, see *Understanding Rational Rose RealTime Licenses* on page 73, or refer to the FLEXlm documentation.

### Problem 2

If you receive an "Unable to obtain a license message" message after the splash screen is displayed, check the expiration date of your license.

### Problem 3

If you have received a floating license file and are unable to obtain a license, verify that the license daemon is running. *Installing a Floating License Key on a UNIX server* on page 92.

## UNIX

### Problem 1

If a FlexLM License Manager dialog appears indicating that "Your application was unable to obtain a license because...", do the following:

- 1 Click **Cancel**.

You will get a Rational Rose RealTime message stating "Unable to obtain a license".

- 2 Click **OK**.

- 3 Verify the location and naming of the license file:
  - Verify that the variable set matches the actual location and file name, by typing the following in a command prompt:  

```
echo $ROSERT_LICENSE_FILE
```
  - If you are incorporating this file into an existing FLEXlm license file, see *Understanding Rational Rose RealTime Licenses* on page 73, or refer to the FLEXlm documentation, to ensure that the setup and key activation was done correctly.
- 4 If both the name and location are correct, verify that the install process set the **ROSERT\_LICENSE\_FILE** environment variable to the location of the file.  
  
If the environment variable is not set or set incorrectly, add or modify as appropriate.

### **Problem 2**

If you receive an "Unable to obtain a license" message after the splash screen is displayed, check the expiration date of your license.

### **Problem 3**

If you have received a floating license file and are unable to obtain a license, verify that the license daemon is running. See the *Installing a Floating License Key on a UNIX server* on page 92.

## Contents

This chapter is organized as follows:

- *Migrating from Rational Rose* on page 99
- *Migrating from ObjecTime Developer 5.2/5.2.1* on page 105
- *Migrating from Rational Rose RealTime 6.0/6.0.1/6.0.2/6.1* on page 108
- *C Language Migration* on page 112
- *C++ Language Migration* on page 115

This chapter provides help for users migrating models from Rational Rose, ObjecTime Developer, or previous releases of Rational Rose RealTime.

## Migrating from Rational Rose

---

The Rational Rose RealTime interface is similar to Rational Rose; however, there are some subtle differences that Rose users should understand before using Rational Rose RealTime.

### User Interface Differences

If you are familiar with Rose, you should not have too much trouble understanding the Rational Rose RealTime user interface. Rational Rose RealTime has maintained the same architecture as Rational Rose and has preserved the main toolset features: a model browser, diagrams, model properties, add-ins, and an extensibility interface (RRTEI).

**Note:** Some of the icons have been modified but they have remained intuitive.

However, to support modeling real-time systems, to allow full code generation, and to provide an executable interface, you will notice the following main changes to the Rational Rose RealTime interface. For a complete description of the Rational Rose RealTime user interface please refer the *Rational Rose RealTime Toolset Guide*.

## Multiple Model Browsers

The browser in Rational Rose RealTime have three views (tabs): the **Model View**, the **Containment View**, and the **Inheritance View**. Each view displays the elements in your model from different perspectives.

In addition, you can create multiple model browser windows by selecting **View > Browsers > Create New Browser**.

## Output Windows

In Rational Rose, the log is in an undockable window that cannot be dragged onto another section or window. In Rational Rose RealTime, the output window is dockable, and contains a set of windows that show different kinds of output from the toolset.

## Code Editors

In Rational Rose, code is added to operations outside the toolset; in Rational Rose RealTime, code is added in the tool. Code is added to model elements through their specification dialogs. For example, the **Details** tab of an Operation specification contains a Code window in which you can write the body source code of the operation.

Code can also be added to capsule state diagrams.

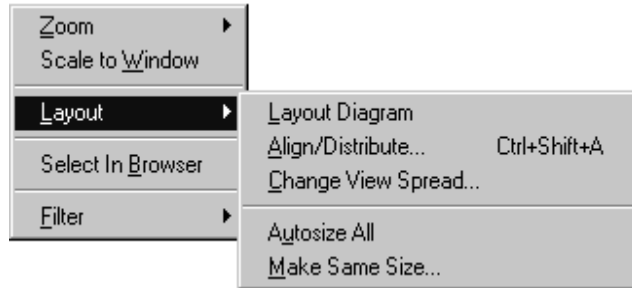
## Code Browser

During the development of a model, you spend considerable time writing source code. In Rational Rose RealTime, you can edit the code for the currently selected element in the code window, rather than having to open the element's specification dialog.

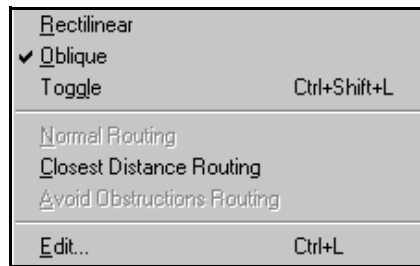
## Layout Tools and Line Styles

Rational Rose RealTime allows you to perform advanced layout operations on diagrams. For example, you can align, change the view spread, and make elements the same size. You can also configure the way lines are drawn:

**Figure 1 Layout Menu - Right-click on any Diagram**



**Figure 2 Line Attributes Menu - Edit > Line Attributes**



## New Modeling Language Elements

Rational Rose RealTime introduces new modeling elements - capsules, protocols, and ports - and a new diagram - the structure diagram. The *Rational Rose RealTime Modeling Language Guide* contains information about the new modeling elements, as well as a summary of the real-time specializations to the UML.

You can also review the Concept Tutorials.

## Code Generation, Building, and Running

An important difference between Rational Rose and Rational Rose RealTime is the support for building and executing models from within the toolset. Note the following:

- Rational Rose RealTime is not meant to be used in a round trip process. The model contains all the information required to generate, build, and run elements in the model.
- Rational Rose RealTime does not ship with a compiler for your target environment. You must install and configure a compiler for your target. Rational Rose RealTime will use that compiler to build the model.

For more information, see the *Rational Rose RealTime Toolset Guide*, available through the online Help.

## Opening Models from Rational Rose

Rational Rose RealTime can open files saved with Rational Rose 98, 98i, or 2002 (.mdl files).

### Fixing Unresolved References

When importing a model from Rational Rose 98 or Rational Rose 98i into Rational Rose RealTime, you should fix any model errors in Rational Rose (**Tools > Check Model**) before trying to import the model. In particular, it is important to resolve any unresolved references. Rational Rose is not concerned with unresolved references; however, they are very important in Rational Rose RealTime as they can result in incomplete code generation and compilation errors.

For more information, see “Model Validation” in the *Guide to Team Development*.

### Opening a Rational Rose Model in Rational Rose RealTime

To open a Rational Rose model in Rational Rose RealTime:

- 1 Select **File > Open** and choose **Rose Model (.mdl)** from the **Files of Type** drop-down menu.
- 2 Select a file and click **Open**.

Opening a new model discards any existing model that you have. The tool prompts you to save changes first.

## List of Importation Log Messages

The following messages may appear in the Log after a Rose98 model has been imported.

**Message:** Warning: Renamed elementClass “oldElementName” to “newElementName”.

**Description:** A loaded model element has been renamed to conform with Rational Rose RealTime's naming requirements. Double-clicking on the warning in the log, this may display the renamed element.

**Message:** Error: Unresolved reference from ... to ... by ...

**Description:** The toolset was unable to resolve a reference between two model elements. This is usually the result of loading an incomplete model, for example, when the user has updated only part of a model from CM. The rest of the model needs



to be loaded in order for the reference to be resolved. However, in some cases, the unresolved model element is removed from the model and the deletion is recorded in the log window.

**Message:** Error: Error reading file fileName at line lineNumber or Error message detail.

**Description:** The error message detail may contain validation errors originating from the internal meta-model. Possible error message details that originate from the petal reader are listed below.

**Message:** Invalid syntax.

**Description:** The file contents cannot be read by the toolset. The user should send the file to customer support with a description of what they were doing when the file was created. For example, if you import a Rose98 model and make some changes to the Component View, the file will not reload in Rational Rose RealTime.

## Limitations and Restrictions

When a Rose model is opened in Rational Rose RealTime, the following elements are not converted:

- Importing Rational Rose models containing controllable units is not supported  
Load the model with controllable units in Rational Rose. Export the model into a single .ptl petal file. Import the .ptl file into Rose. Save the model as a .mdl file in Rational Rose. Open the .mdl file in Rational Rose RealTime.
- Three-tier class diagrams are not supported in Rational Rose RealTime.  
Rational Rose RealTime skips over three-tier class diagram making it unnecessary to remove them before importing.
- Rational Rose elements that are not supported are written to the Documentation field in Rational Rose RealTime.

**Note:** The conversion of models is supported in one direction only: once models are brought into Rational Rose RealTime, if they are converted back to Rational Rose, the additional Rational Rose RealTime functionality will not appear in Rose. Working in a mixed Rational Rose RealTime/Rose environment is not supported. Generated code is not compatible between the two tools.

## Importing Rational Rose Generated Code

Source code that has been generated from a Rational Rose model and has been edited within the preserved regions may be imported.

### To Import Rational Rose Generated Code:

- 1 Verify that the Rational Rose .mdl file is not newer than the generated code. If so, regenerate the code.
- 2 Open the Rational Rose model.

For details, see *Opening Models from Rational Rose* on page 102.

- 3 Choose **Tools > Import Code...**

If code was generated from this model using Rational Rose and the model was saved after the code generation was performed, a "Rose Code Import" window appears. Otherwise, a "There are no .cpp or .h files available for import" message is displayed.

The Rational Rose Code Import Window lists all the .cpp and .h files that were generated from the model and lets you select all or a subset of the files. It also displays the classes that will be affected by each file that is selected. After a file is imported, it will not be listed if code importation is repeated.

- 4 After you have complete importation and are satisfied with the results, save the model.

### Limitations and Restrictions

- No action will be taken on empty preserved regions. As a result, constructors, destructors and operators that are generated by Rational Rose and have empty preserved regions, will not be added to the model.
- Use of the **Code Name** properties for classes and operations can cause inconsistent naming in the generated code. The inconsistencies can cause compile time errors, which can be resolved manually.

## Migrating from ObjecTime Developer 5.2/5.2.1

---

Users migrating from ObjecTime Developer can open their models in Rational Rose RealTime. First, see the *Conversion Guide - ObjecTime Developer to Rational Rose RealTime* to get your ObjecTime Developer model loaded and built in Rational Rose RealTime.

**Note:** Contact Rational Customer Support to obtain the latest Objectime model conversion patches.

### Terminology

The modeling language and toolset terminology in Rational Rose RealTime is different than that used in ObjecTime. This section provides an overview of the changes.

#### Actor/binding/protocol Class

Rational Rose RealTime supports the UML modeling language. Therefore, certain modeling elements are referred to by UML standards differently than they are in ROOM (Real-Time Object-Oriented Modeling). For detailed information regarding the UML modeling elements supported in Rational Rose RealTime, see the *Modeling Language Guide*.

**Table 10 Terminology Mappings from ROOM to UML**

ROOM	UML
actor	capsule
actor reference	capsule role
protocol	protocol
port	port
SAP/SPP	unwired ports
binding	connector

#### Context/update

In ObjecTime Developer, contexts and updates contain a group of related actors, protocols, and data classes. In Rational Rose RealTime, models are stored in controlled units that can vary in granularity. For example, the whole model can be stored as a single controlled unit (default) or each element can be stored individually. If a model

is stored as one controlled unit, then the model file (.rtmdl) contains all information about a model. If the model file is read-only, then when the model is opened in Rational Rose RealTime it is also read-only.

### **Activation/passivation**

These terms have been replaced by more commonly used open and save. You open a model into Rational Rose RealTime, and save it to disk.

For more information, see the *Toolset Guide*.

### **Workspace Browser**

In ObjecTime Developer, workspace browsers showed all activated contexts and updates. Since Rational Rose RealTime only supports one model loaded at a time, there is no equivalent concept.

The workspace in Rational Rose RealTime is associated with a specific model and is saved as such. The workspace can be stored under Configuration Management, if desired.

### **Model Browser**

Rational Rose RealTime still has a model browser. You can, however, have more than one browser for a model, and each browser shows the model from three different views (tabs): the **Model View**, the **Containment View**, and the **Inheritance View**.

For more information, see the *Toolset Guide*.

### **Project Files**

Project files do not exist in Rational Rose RealTime. An equivalent concept is the model file (.rtmdl) that contains references to a set of packages, but does not contain version information. Rational Rose RealTime does not manage versions of files. Instead the model file loads the packages it finds on disk. It is up to the developer, through their configuration management process, to ensure that the files on disk are the correct version.

### **Library Browser**

Library browsers do not exist in Rational Rose RealTime. Because of the changed underlying model representation, the configuration management integration has changed significantly in Rational Rose RealTime.

It is highly recommended that you read the *Guide to Team Development* for a detailed introduction to using source control with Rational Rose RealTime.

## User Interface Differences

For a complete description of the Rational Rose RealTime user interface, please refer to the *Toolset Guide*. Rational Rose RealTime looks very different than ObjecTime Developer. Although you can accomplish almost everything you can in ObjecTime Developer, the steps and mechanics are very different. For this reason, it is recommended that you review the tutorials to become familiar with the interface. You can also take the Rational University course called DRTSwRRRT.

**Note:** When using Rational Rose RealTime, everything is right-click-centric, meaning that you can right-click on every element in the toolset to show a context-menu that contains actions that you can perform.

### Property Editors

Property editors have been replaced by specification dialogs. Every modeling element has a specification dialog that contains a non-graphical view of its properties. To access an element's specification, right-click on the element (in either the browser or on a diagram) and select **Open Specification**.

### List Headers

In ObjecTime Developer, every window has a list header in which you can access menu items specific to that window. In Rational Rose RealTime, these have been replaced by right-click menus and the main application menu.

### State and Structure Diagrams

To open a state or structure diagram, right-click a capsule, and click **Open Structure Diagram** or **Open State Diagram**. The state and structure diagram editors appear in the same window. You can switch between one and the other using the tabs at the bottom of the window. If you want to see the structure and state diagrams simultaneously, click and drag one of the tabs away from the window. This undocks the diagram and creates a new window containing only the selected diagram. You can redock the diagrams by dragging one of the tabs into the other.

For more information, see the *Conversion Guide, ObjecTime Developer to Rational Rose RealTime*

## Compilation

In ObjecTime Developer 5.2/5.2.1, data classes were compiled one package at a time. In Rational Rose RealTime, data classes are compiled one class at a time.

## Migrating from Rational Rose RealTime 6.0/6.0.1/6.0.2/6.1

---

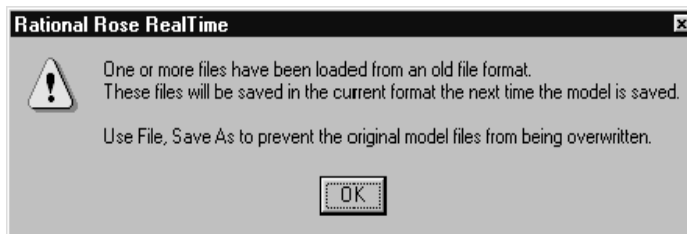
Models from these previous versions of Rational Rose RealTime are compatible with this version. However, there are some changes in team development and language add-ins that require you to plan some changes to your model.

**Note:** Beta customers must uninstall before installing the new release.

### File Format Changes

When opening a Rational Rose RealTime 6.x model, a dialog may warn you that the next time the model is saved, the files will be saved in the new file format. To prevent the original model from being overwritten, on the **File** menu, click **Save As**.

**Figure 3** Warning Dialog



For this reason, when working with a model under source control, you must check out all controlled units so that they can be saved in the new format.

### Source Control Migration

If your model is in source control, you need to load it into the new release of Rational Rose RealTime.

#### To Save a File in the New File Format:

- 1 In the 6.0 toolset, all files should be checked in, and the model should build and test successfully.

The source control administrator/model converter **checks out all files** from the 6.0 toolset.

- 2 Install and start the new release of Rational Rose RealTime.
- 3 Open the .rtmdl file in Rational Rose RealTime.

**Note:** Do not open the workspace (.rtwks).

- 4 Save the model.

- 5 Configure the source control settings.
- 6 Save the Workspace.
- 7 Submit all changes.

**Note:** Migration from 6.0 is one-way. After you have migrated a model , you cannot successfully reload a controlled unit in 6.0 format. Although the toolset lets you attempt to reload a controlled unit, several errors will be reported. A mixed model is not supported.

### **ClearCase Integration**

Rational Rose RealTime models currently stored in a ClearCase VOB should be converted to use the type manager in order to take advantage of the new integration features. A script, `cc_chtype.pl`, has been included to help in the conversion process. The script, located in `$ROSE_HOME/bin/$ROSE_HOST/cc`, produces a log of commands that will convert the existing model files from the default "text\_file" type to the supplied "rosert\_unit" type.

After following the setup directions detailed in the "Source Control Tools" chapter in the *Guide to Team Development*, use the following invocation from the root of your VOB to produce a batch file, which when executed will convert any Rational Rose RealTime files to the `rosert_unit` type:

```
rtperl cc_chtype.pl -cmdfile chcmds.bat -recurse *
```

After examining the `chcmds.bat` file and verifying that the commands contained within it are the commands you want to perform, execute the batch file.

If you do not want to be queried to convert each file, add "**-chargs -f**" to the `cc_chtype.pl` command line before the `-recurse` argument.

```
rtperl cc_chtype.pl -cmdfile chcmds.bat -chargs -f -recurse *
```

This will generate commands that force the type change without querying.

For ClearCase users who want to use `clearmake`, there is a problem with filenames with spaces in them. For help with this, contact Rational Customer Support at:

<http://www.rational.com/support>

### **Migrating Customized CM Scripts**

For complete information on library scripts and what scripts may require modification to meet your specialized CM needs, see the *Guide to Team Development*.

## Language Add-in Changes

The C and C++ Language Add-Ins have changed, it is very important to read *C Language Migration* on page 112 and *C++ Language Migration* on page 115 for instructions on migrating existing models to either of these Language Add-Ins.

**Note:** Rational Rose RealTime version 2001A.04.00 (and later) supports the Java language.

## Running Two Different Releases of Rational Rose RealTime

### Windows NT

When you install version 2003.06.00 of Rational Rose RealTime, all older versions of Rational products are uninstalled.

### UNIX

You can set up your environments to run both releases of Rational Rose RealTime, but do not run them from the same machine at the same time. This is a MainWin limitation. There are workarounds that are available as solutions on the Rational Customer Services knowledge base.

For additional information, see *Upgrading to 2003.06.00 While Maintaining an Earlier Version* on page 54.

## Workspace Files

Version 6.0.x workspace files are not supported. You must open the model without the workspace. The unsupported workspace is backed up to a file.

## RRTEI Changes

If you have previously used any of the following classes or functions in your scripts, they have to be removed in order for your scripts to be compatible with this new release:

- ComponentAggregationCollection class
- ComponentAggregation class
- Component::GetComponentAggregation()
- Component::AddComponentAggregation()
- Component::DeleteComponentAggregation()
- ComponentPackage::GetObject()
- RSSchedule enumeration
- Schedule rich type



If you have previously used any of the following classes or functions in your scripts, they have to be replaced in order for your scripts to be compatible with this new release. Use the model element's tool's properties. For example, The old **Component::OutputPath** property can now be retrieved by the "C++ Generation" **OutputDirectory** property from the component.

- Component::OutputPath
- Component::TopCapsule
- Component::RTSType
- Component::TargetLibrary
- Component::RTSDescription
- Component::CompilerName
- Component::CompilerLibrary
- Component::CompilerFlags
- Component::CompilerDescription
- Component::Inclusions
- Component::UserObjectFiles
- Component::InclusionPaths
- Component::LinkerName
- Component::LinkerFlags
- Component::LinkerDescription
- Component::ExecutableFileName
- Component::Platform
- Component::MultiThreaded
- Component::DefaultArgs
- Component::TargetDescription
- Component::CodeGenMakeName
- Component::CodeGenMakeFlags
- Component::CodeGenMakeOverridesFile
- Component::CodeGenMakeDescription
- Component::CompilationMakeName
- Component::CompilationMakeType
- Component::CompilationMakeFlags
- Component::CompilationMakeOverridesFile
- Component::CompilationMakeDescription
- Component::UserLibraries
- Component::UserSourceFiles
- Component::UserLibraryPaths
- Component::CodeGenMakeType
- Component::AddInclusion()
- Component::DeleteInclusion()
- Component::AddUserLibrary()

- Component::RemoveUserLibrary()
- Component::AddUserObjectFile()
- Component::DeleteUserObjectFile()
- Component::AddInclusionPath()
- Component::DeleteInclusionPath()
- Component::GetInclusionPathFlag()
- Component::AddUserLibraryPath()
- Component::DeleteUserLibraryPath()

## C Language Migration

---

The following section provides details on migration issues specific to the C Language Add-in.

For more information on the C Language Add-in, refer to the *Rational Rose RealTime C Reference* .

### Converting a C++ Model to C

You can convert a C++ model to C, however, the process is not as simple as changing the language of each model element. First, the C Services Library's API is different than that of the the C++ Services Library, meaning that all the Services Library references in the detail code must be changed. Secondly, the C Services Library does not support dynamic structure (import/deport), which may require you to re-design you model. In addition, all issues regarding conversion from regular C++ to C still apply to the conversion (for example, polymorphism is not supported in C, encapsulation is not enforced, all fields in a struct are public, and so on...).

You should decide early in the development cycle whether your project will be developed in C or C++ because changing languages in the middle of development requires a lot of work.

#### To Convert an Existing Rational Rose RealTime Model Based on the C++ Language:

- 1 Make a backup copy of the C++ model that you are trying to convert.
- 2 Change the language of each model element. The language setting is on the **General** tab of each element's specification dialog.

**Note:** When model elements change languages, all the C++ language properties are replaced by C language properties. Therefore, any properties that have been modified are lost when the language is changed.

- 3 Review the *Rational Rose RealTime C Reference* for descriptions of the new C properties and how these are to be used in your model.
- 4 All attribute and operations should be made public. The model will continue to build with them as private or protected, but the code generator will output many warnings in this regard.
- 5 If your C++ model depends on dynamic structure and importation, you can mimic this behavior in a C model by combining the static linkage of ports between capsules and the dynamic linkage of unwired ports. With some re-design, you can replace importation from your C++ model to use unwired ports and the **RTPort\_registerAs()** and **RTPort\_deregister()** functions to bind and unbind ports dynamically.
- 6 Convert all timing ports to C Timing, and then add a timing capsule to your model.
- 7 Remove all Log ports and all Exception ports.
- 8 When your design can be supported by C Services Library features, you can convert the syntax in your detail code.  
**Note:** We recommend that you start converting a small set of capsules that can be built and tested separately before trying to convert the whole model. Iteratively modify detail code, build, and test.
- 9 Update your components to C components.
- 10 Configure any of the build properties that are required.

## ObjecTime Developer for C Migration

ObjecTime Developer for C models can be imported into Rational Rose RealTime, compiled, and run with only minor modifications to the model. Functional updates (like a proper recall mechanism and data integration) was not provided via the ObjecTime Developer for C interface and thus will only be available via the new C UML Services Library API.

### Importing Models

Prior to importing a model, you should read the *Conversion Guide, ObjecTime Developer to Rational Rose RealTime* to understand important issues involved with migrating ObjecTime Developer models to Rational Rose RealTime.

## To Import an ObjecTime Developer for C Model into Rational Rose RealTime:

- 1 Set the default language to C.
- 2 Set the default environment to C TargetRTS through **Tools > Options > Language/Environment Tab**. This will ensure that protocol classes import as C Protocols.
- 3 Export and import your OTD for C model. For details, see the *Conversion Guide - ObjecTime Developer to Rational Rose RealTime*.
- 4 When the model has been imported, replace all ports of type Timing with type CTiming in your model.

**Note:** Your triggers (on timeout) will remain valid.

- 5 Update your timing service. If you have a simple timing service, to get you started, replace whatever timing capsule you had with the one available in **Logical View::RTCClasses::TimerPackage::Timer**. You can override this later with a custom timer after you get your model working.
- 6 Build your target.  
**Note:** If you receive a **signal** is undefined build error, replace **signal** with **ROOM\_Signal( port, signal )** for the given port.

## Converting Global Signals to Local Signals

A common update that may be required to some imported models involves the way the signals are now represented. In order to provide local signals, and thus the ability to build libraries without global system knowledge, more macro operations are necessary.

The only supported way of creating signals with the backwards compatible interface is with these primitives:

- **ROOM\_Signal( port, signal )**, where port is the name of the port (unqualified with respect to the this pointer) and signal is the name of the signal.
- **ROOM\_InSignal( port, signal )**, where the parameters are specified identically to the previous case.

In ObjecTime Developer, these macros returned signal. You may have tried to optimize out the use of these macros, and used the signal name when sending messages through these services. However, this will no longer work because these

macros now create a local signal (relative to the protocol class of the port). As a result, you will find compile errors when you go to build your model indicating that the signal is undeclared. Do the following:

Every call of

```
ROOM_PortSend( port, signal )
```

needs to be replaced with

```
ROOM_PortSend( port, ROOM_Signal( port, signal ) )
```

This change applies to all signals used in ROOM\_ macros.

## Timing Service

The global signal **timeout** no longer exists. You need to use **Timing\_rt\_timeout** or use the ObjecTime Developer **RSL\_Timeout()** macro that has been mapped to **Timing\_rt\_timeout**.

Also, remember that these macro operations de-references the pointer for you, so all you have to do is provide the names.

## C++ Language Migration

---

The following section provides details on migration issues specific to the C++ Language Add-in.

For more information on the C Language Add-in, refer to the *C++ Reference*.

If you are upgrading from a previous release of either ObjecTime Developer or Rational Rose RealTime, to build and run your model in *Backwards Compatibility Mode* on page 115. Then, you can convert to the new syntax described in *Changes* on page 119.

See the *Conversion Guide, ObjecTime Developer to Rational Rose RealTime*, that is available as part of the online Help system.

## Backwards Compatibility Mode

An essential requirement of the C++ Language Add-in is that it allows models from previous releases to be loaded, compiled, and run with only small syntax changes to the model. Because of the scope of the changes required to the Language Add-in, most

models will contain constructs that still will not compile even in backwards compatibility mode because of the increased send type checking and removal of global signals.

**Note:** Global signals have been replaced by a signal operation local to each protocol class defining the signal. Signals with the same name in different protocols do not share the same operation.

## Migrating in Two Steps

You can plan your conversion in two steps:

- 1 Build your model in backwards compatibility.
- 2 Convert to the new syntax.

Since you retain the benefits of type safety even in backwards compatibility mode, one option would be to keep active projects in backwards compatibility and only use the new syntax on new projects.

### Advantages of Backwards Compatibility Versus Changing All Syntax

- Only small changes to user code are required.
- There are no run-time penalties.
- You can optionally benefit from the new message send type safety.

### Disadvantages

- There are stubs generated for each protocol to allow backwards compatibility. More code is therefore generated in backwards compatibility mode.
- Compilation times are longer because there is more code to compile.

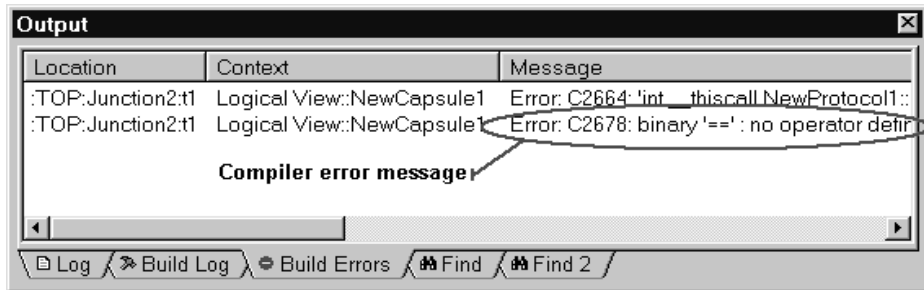
## What Does Backwards Compatibility Do?

Protocols can be marked as backwards compatible (see the **C++ Target RTS** tab of the Protocol Specification). This will tell the code generator to create stub code in the protocol classes to allow use of the old Communication Services syntax.

## Compiler Will Find All Errors

Many errors in existing models will be discovered by the compiler. After a build, the **Build Errors** pane of the output window will have a list of all compile errors. Double-click on the error and the code section containing the error appears.

**Figure 4 Sample Output Window Showing Build Errors**



## Building a Model in Backwards Compatibility Mode

Follow these steps to build and run a model loaded into Rational Rose RealTime to be built and run in backwards compatibility mode.

### Step 1: Optional Type Checking

A flag has been added to the C++ TargetRTS tab for protocols called **TypeSafeSignals**. By default this property is turned on. Turning off the flag causes the code generator to ignore the types for all signals in the protocol class. This is the same as setting them all to blank (for example, any). This sets the type of the data to be sent to void \* and allows **SEND\_SCALAR** to work without change. This is considered a true backwards compatibility mode with the added advantage that it affects the new send syntax as well (i.e. you can turn off backwards compatibility and turn off type safe signals).

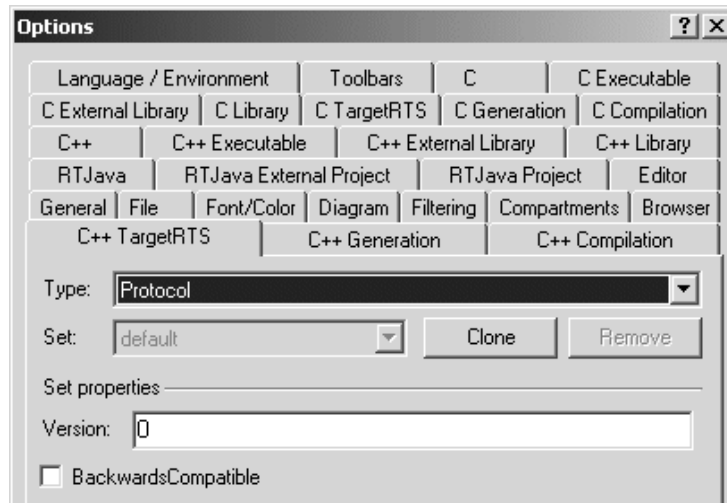
If you want to continue to use the **SEND\_SCALAR** macro you should turn off the **TypeSafeSignals** property on these protocols.

### Step 2: Enable BackwardsCompatible Protocol Property

- Press **F12** or select **Tools > Options** from the main menu, and in the **Options** tab and select the **C++ Target RTS** tab. Then set the **Type** to **Protocol** and ensure that the **BackwardsCompatible** checkbox is checked.

This will ensure that all protocols default to backwards compatibility mode.

**Note:** On loading of ObjecTime Developer models all protocols will automatically be set to backwards compatibility mode.



### Step 3: Clean up Unsafe Sends

Most models contain unsafe sends and sends that are not used as defined in the associated protocol. You should fix these constructs so that you do not need to debug bugs caused by these kinds of errors.

The compiler will find these errors. However if you know where you have signal-type incompatibilities, you can manually fix them.

Previous versions of the C++ UML Services Libraries allowed sending a signal, defined in the protocol to have a data class, to be sent without data. Because of the new tightened type safety of sends, this is no longer allowed and will result in compilation errors. To compile in backwards compatibility mode you will have to modify all errors of this type.

This is an example of a typical compile error for a signal-data class mismatch:

```
int __thiscall NewProtocol1::base::send(const struct RTSignal_start
&,const class AClass1 &,int)' : cannot convert parameter 2 from
'int' to 'const class AClass1 &
```

### Step 4: Remove Unspecified '\*' Replication Values

You can search your model for unspecified replication values by using the find tool and searching Cardinality/Multiplicity fields for the value '\\*'.



## Step 5: Investigate Remaining Syntax Changes

- The first step is to identify if you use message forwarding or if you access signal names in user code. You will have to convert these constructs as described in *Forwarding* on page 126 and *Discriminating in Code the Signal of a Received Message* on page 126.

Example compile error message when using old forwarding syntax:

```
int __thiscall NewProtocol1::base::send(const struct RTSignal_start
&,const class AClass1 *,const struct RTOBJECT_class *,int)' : cannot
convert parameter 1 from 'int' to 'const struct RTSignal_start &'
```

Example compile error message when using signal name in user code:

```
binary '==' : no operator defined which takes a left-hand operand of
type 'int' (or there is no acceptable conversion)
```

**Note:** If you still have compilation problems, review *Changes* on page 119 to ensure that you are not using classes that have been removed from the Services Library.

## Full Migration

When your model is compiling and running in backwards compatibility mode, the next step for full migration is a communication service syntax change. You will have to find and replace occurrences of old syntax with the new syntax and individually turn off the BackwardsCompatibility flag on a per protocol basis. For a complete listing of the change communication service primitives, see *Changes* on page 119 section.

## Changes

This section explores all the changes affecting users of the C++ Language Add-in who will be migrating their existing models to this new version.

### C++ UML Services Library

Adding support for libraries and type safety required changing the Communication Service API. Review these sections to understand the new C++ Services Library changes.

- *Type Safety Explained* on page 120
- *New Classes for Protocols, Signals, and Ports* on page 120
- *API Changes Summary* on page 121
- *Macros* on page 130
- *External Layer Service (ELS)* on page 131

No attempt will be made to describe changes made to the private or undocumented features of the C++ Services Library. We recommend that you always use only the documented interfaces.

**Note:** For minor problems migrating customizations or configurations of the C++ UML Services Library contact Rational Customer Support. For all other problems migrating your custom changes contact your sales representative to arrange for consulting services to assist in the migration.

## Code Generation and Compilation

Components have been expanded to allow building libraries and model external libraries.

## New Classes for Protocols, Signals, and Ports

In previous versions of the Services Library **RTEndPort** and **RTEndPortRef** classes were used to represent port instances and port references. These classes have been replaced by **RTProtocol**, **RTOutSignal**, **RTInSignal**, and **RTSymmetricalSignal** classes.

For each protocol in a model a structure is generated. Contained in the structure are a Base and Conjugate class which are subclasses of **RTProtocol**. For each signal defined in the protocol an operation is generated in the Base and Conjugate classes. The introduction of the new classes has changed the syntax of communication service operations.

## Type Safety Explained

In a protocol specification, a signal may be defined with an associated data class. Previously, it was optionally up to the software designer whether or not to actually send data along with such signals. In addition you were able to send signals that were not defined on the port on which they were sent.

In summary, there has never been any support for compile-time validation that user code conformed to a protocol specification. Consequently all errors of this type could only be caught at run-time, resulting in developers having to track down “unexpected message warnings” and run-time exceptions.

## How Has This Changed?

In the new UML Services Library, you must send data if the signal has an associated data type. The data must be of the type, or a subclass of the type, specified for that signal. Alternatively, the data may be of type `void` or left empty. A data class type left empty (that is, no type specified) implies that you can send anything with the signal. In addition you can only send signals that have been defined on the protocol role associated with the port.

**Note:** Backwards compatibility mode allows previous release syntax to be used in models compiled with the current release of the C++ Services Library.

The **TypeSafeSignals** flag on protocols can be used to force the code generator to ignore the data class value of all signals defined in a protocol. The code generator treats the signal's data class as being empty, thus allowing any type of data class to be sent with the signal.

## API Changes Summary

The changes affecting the communication service interface can be grouped into the following usage scenarios:

- *Asynchronous Sends* on page 122 (to one or all port instances)
- *Synchronous Sends* on page 123 (to one or all port instances)
- *Message Reply* on page 123
- *Defer, Recall, and Purge* on page 124 (one or all signals to one or all port instances)
- *Port Indexes* on page 125
- *Discriminating in Code the Signal of a Received Message* on page 126
- *Forwarding* on page 126 (potentially from one protocol to another and to one or all port instances)
- *RTPortRef Operations* on page 128

In addition to the changes in the communication service review, the following issues that may impact your conversion:

- **RTTimespec** parameters

All examples in this section assume that a replicated port called **aPort** of type **aProtocol** is defined on a capsule.

The protocol is symmetric (in and out signals are the same) and is defined as:

Signal	Data Class
start	AClass1
stop	int
reset	RTInteger

**Note:** The examples show sending RTInteger (a type of **RTDataObject** with which ObjecTime Developer 5.2 users will be familiar), and regular classes created using Rational Rose RealTime 6.0, AClass1.

## Asynchronous Sends

5.2/6.0

```
port.send(signal, rtdataobject, priority);
port.send(signal, data, type, priority);
port[index]->send(signal, rtdataobject, priority);
port[index]->send(signal, data, type, priority);
```

New syntax

```
port.signal(rtdataobject).send(priority);
port.signal(data).send(priority);
port.signal(rtdataobject).sendAt(index, priority);
port.signal(data).sendAt(index, priority);
```

New syntax example

```
RTInteger level(15); // RTDataObject
AClass1 mdata(49, 1.23);

aPort.reset(level).send(); // broadcast
aPort.start(mdata).send(); // broadcast
aPort.reset(level).sendAt(1); // single port
aPort.start(mdata).sendAt(1); // single port
```

## Synchronous Sends

5.2/6.0

```
port.invoke(repbufs, signal, rtdataobject);
port[index]->invoke(repbuf, signal, rtdataobject);
port.invoke(repbufs, signal, data, type);
port[index]->invoke(repbuf, signal, data, type);
```

New syntax

```
port.signal(rtdataobject).invoke(repbufs);
port.signal(data).invoke(repbufs);
port.signal(rtdataobject).invokeAt(index, repbuf);
port.signal(data).invokeAt(index, repbuf);
```

New syntax example

```
RTInteger level(5); // RTDataObject
AClass1 mdata(49, 1.23);
RTMessage replyBuffers[5];
RTMessage replyBuffer;

aPort.reset(level).invoke(&replyBuffers); // broadcast
aPort.start(level).invokeAt(1, &replyBuffer); // single port
aPort.reset(mdata).invoke(&replyBuffers); // broadcast
aPort.start(mdata).invokeAt(1, &replyBuffer); // single port
```

## Message Reply

5.2/6.0

```
msg->sap()->send(signal, rtdataobject);
msg->sap()->send(signal, data, type);
msg->reply(signal, rtdataobject);
msg->reply(signal, data, type);
```

New syntax

```
rtport->signal(rtdataobject).reply();
rtport->signal(data).reply();
```

New syntax example

```
RTInteger level(5); // RTDataObject
AClass1 mdata(49, 1.23);
```

```
rtport->reset(level).reply();
rtport->start(mdata).reply();
```

**Note:** **rtport** is an argument passed to each transition code segment. It is a pointer to the port on which the triggering signal was received. For more information see *Parameters Available in Transition Code* on page 132.

If a transition is triggered by signals arriving from different ports with different protocols, then the **rtport** argument cannot be used to reply. In these cases you will have to either explicitly cast the port or create a separate transition to reply to signals arriving on a specific port.

```
((AProtocol::Base *)msg->sap())->Ack().Send();
```

The difference between **rtport** and **msg->sap()** is that **rtport** is coerced to the correct protocol type by the code generator whereas **msg->sap()** is a pointer to a generic **RTProtocol** object.

## Defer, Recall, and Purge

5.2/6.0

```
port.purge(signal);
port[ index ]->purge(signal);
port.recall(signal, front);
port[ index ]->recall(signal, front);
port.recallAll(signal, front);
port[ index ]->recallAll(signal, front);
port.recallAll();
port.recallAll(0, front);
```

### New syntax

```
port.signal().purge();
port.signal().purgeAt(index);
port.signal().recall(front);
port.signal().recallAt(index, front);
port.signal().recallAll(front);
```

```

port.signal().recallAllAt(index, front);
port.recall();
port.recallFront();
port.recallAt(index);
port.recallAll();
port.recallAllFront();
port.recallAllAt(index);
port.purge();
port.purgeAt(index);

```

### New syntax example

```

// a signal must have already been deferred
// using a call to msg->defer().

// purge all deferred messages on all port instances
aPort.purge();

// recall all deferred bye signals
aPort.bye().recall();

```

## Port Indexes

5.2/6.0

```

msg->sap()->getIndex(); // 0-based
msg->sap()->index(); // 1-based
msg->sap()->at(index) // 1-based

```

### New syntax

```

msg->sapIndex0(); // 0-based
msg->sapIndex(); // 1-based

```

**Note:** The `at()`, and `getIndex()` operations are no longer supported.

### New syntax example

```

AClass1 mdata(1, 4.56);
int      index = msg->sapIndex0();

// send back to same port instance on

```

```
// which we just received a message.
rtport->start(mdata).sendAt(index);
```

## Discriminating in Code the Signal of a Received Message

You may have code that used a signal outside the scope of a message send. For example:

```
AClass1 mdata(1,4.56);
int      index = msg->sap()->getIndex();
if( msg->getSignal() == hello )
{
    aPort.start(mdata).sendAt(index);
}
```

Since these signal values are not global you have to use the enumeration values for the signals defined in their respective protocol role. For example, you would have to change the above code fragment to:

```
AClass1 mdata(1,4.56);
int      index = msg->sapIndex0();

if( msg->getSignal() == NewProtocol1::Base::rti_hello )
{
    aPort.start(mdata).sendAt(index);
}
```

**Note:** The signal value in the protocol will always be called `rti_<signalname>`. You can easily reference it by using the following syntax:

`Protocol::<ProtocolRole>::rti_<signalname>`, as shown above. `ProtocolRole` will be either **Base** or **Conjugate**.

## Forwarding

In previous versions of the C++ UML Services Library, you were permitted to blindly forward signals out other port instances. Because signals are no longer global (that is, a signal with the same name and data class in two protocols won't have the same signal operation) this will no longer work.



5.2/6.0 forwarding syntax:

```
port.send(msg->signal, msg->data);
port.send(msg->signal, msg->data, msg->type);
```

## Static Forwarding Pattern

In most cases, you can implement simple forwarding behavior by discriminating the received signal then explicitly sending a signal out another port. The outgoing signal doesn't necessarily have to be the same name as the incoming signal. Static forwarding requires signal discrimination in a transition (for example, using a switch statement) or adding transitions for each signal being forwarded.

Examples

```
// using one transition to route all
// incoming messages to other ports.

switch( msg->getSignal() )
{
    case NewProtocoll::Base::rti_start:
        outport.start(*rtdata).send();
        break;

    case NewProtocoll::Base::rti_stop:
        outport.stop(*rtdata).send();
        break;

    default:
        log.log("Unexpected message");
}

// or one transition per
// signal. In this case each transition
// would forward one signal.

outport.start(*rtdata).send();
```

## Dynamic Forwarding

Some routing capsules are designed so that they won't know the exact protocols for the forwarding ports at design time (that is, they could be overridden at run-time). In these cases, the switch statement described in the static forwarding pattern does not provide a good solution.

Dynamic forwarding provides run-time mapping from one protocol to another. It works by creating a signal map table to map signal numbers from one protocol to another based on the signal name and the data class. This provides constant signal lookup. In addition, signals that don't have compatible data classes are not added to the signal map.

Dynamic forwarding support has not been added to the UML Services Library. Instead a set of classes has been created that can be used in any model that requires this level of forwarding. To use dynamic forwarding please refer to the Dynamic Forwarding model example in the Examples. The example model contains the forwarding classes, or adaptors, and sample usage of these classes. In general capsules requiring dynamic forwarding will have to do the following:

- 1 For each port pair where forwarding will be used, an adaptor object is created to initialize and encapsulate the signal map. If you have forwarding from port A to B and A to C you will need 2 adaptor objects.
- 2 Each adaptor is initialized at run-time with the in and out protocols. This will create the signal map.
- 3 When forwarding is required in a transition, pass the message to be forwarded to the adaptor.

The example model that contains the forwarding classes (adaptors and signal maps) can be found in:

```
$ROBERT_HOME/Examples/Models/C++/DynamicForwarding
```

## **RTPortRef Operations**

The **RTPortRef** class is no longer part of the UML C++ Services Library. Operations that used to be available on this class have been moved to the **RTProtocol** class. This is a summary of the operations that have changed going from the **RTPortRef** to the **RTProtocol** class:

## **RTEndPoint \*\* RTPortRef::incarnations()**

This was last present in ObjecTime Developer 5.2. You will have to use a port (RTProtocol) paired with an index wherever a pointer to **RTEndPoint** appeared previously. For example, before you would have:

```
RTEndPoint ** ports = portref.incarnations();
for( int i = 0; i < portref.size(); i++ )
    (*ports)[i]->send(ack);
```

This has to be converted to:

```
for( int i = 0; i < portref.size(); i++ )
    portref.ack().sendAt(i);
```

The valid indices are from 0 to (port.size()-1), inclusive.

## **RTEndPoint \*\* RTPortRef::incarnationsTo()**

There is no direct replacement for this. Users will have to base their loop on the port index rather than an index into the returned array of pointers. Within that loop you will want to use

```
int RTProtocol::isIndexTo( int, RTActor * ) const
```

to discover the replication indices which correspond to incarnations that would previously have been included in the array. This new interface is more efficient because it avoids the need to allocate and release a block of memory.

## **RTEndPoint \* RTPortRef::incarnationTo():**

This operation is replaced by **RTProtocol::indexTo()**. For example, here is a common use of incarnationTo and how it can be converted to use **indexTo**:

```
RTActorId aid = frame.incarnate( role1 );
RTEndPoint * port = (RTEndPoint *)0;
if( aid.isValid() ) {
    port = replicatedportref.incarnationTo( aid );
    if( port != (RTEndPoint *)0 )
        port->send( Signal );
}
```

Is replaced with **RTProtocol::indexTo()**,

```
RTActorId aid = frame.incarnate(role1);
int port_index;
if( aid.isValid() ) {
    port_index = replicatedportref.indexTo( aid );
```

```
if( port_index != -1 )
    port.Signal().sendAt(port_index);
}
```

## RTTimespec Parameters

ObjecTime Developer (OTD) models which used the **RTTimespec** constructor with only one parameter, as in the following code:

```
timer.informIn(RTTimespec(2));
```

will result in a compile error after conversion of the model to Rational Rose RealTime. The compile error will appear something like:

```
..\rtg\Driver.cpp(67) : error C2440: 'type cast' : cannot convert from 'const int' to 'struct RTTimespec'
```

No constructor could take the source type, or constructor overload resolution was ambiguous.

The reason is that in OTD, the **RTTimespec** constructor included default arguments, that is, **RTTimespec** (long=0, long=0). The default constructor values are not supported on **RTTimespec** in Rational Rose RealTime. Any code that made use of the default arguments needs to be changed to supply both constructor arguments. For example:

```
RTTimespec(2);
```

must be changed to:

```
RTTimespec(2, 0);
```

## RTSignalNames

Some users have accessed this private structure to find signal names. Support for accessing this structure was never supported and has been removed from the UML Services Library. If you have referenced this structure look at replacing this functionality with the **RTMessage::getSignalName()** operation which returns the name of the signal received in the current message.

## Macros

The following pre-defined macros will continue to be backwards compatible.

```
SEND_PTR( ptr )
```

```
RECEIVE_PTR( type )
SEND_SCALAR( value )
RECEIVE_SCALAR( type )
SEND_EXT( value )
RECEIVE_EXT( type )
```

## External Layer Service (ELS)

In version 6.0 of the C++ Services Library the ELS was included in the pre-compiled C++ UML Services Libraries. However source code was not shipped. In the current release of Rational Rose RealTime the ELS is not provided for use, nor supported with the release. Please refer to the IPC Application Note and Example for information on how the ELS can be replaced. The External Layer has been replaced by Rational Connexis. Further information on Add-ins, including Connexis, can be found in the online Help and on the Rational Rose RealTime product web site:

<http://www.rational.com/products>

## Code Generation

To support scalable build environments the C++ Language Add-in now supports the ability to break systems into a number of independently buildable components. You can now use components to build libraries, executables, and model external libraries. See *Components* on page 131. To support different component types and provide an extensible interface for components several Model Properties have been added to components.

## Components

Components are collections of references to model elements that are used to build something. In Rational Rose RealTime, there are three kinds of components:

- **C++ Executable:** produces an executable.
- **C++ Library:** produces a library file containing the object files for the classes referenced by the component.
- **C++ External Library:** does not actually produce a build output, but represents a pre-built and packaged component within a model.

The build options for each component type are stored in a set of model properties. In Rational Rose RealTime 6.0, a component's build options were hard-coded attributes of the component. See the *Rational Rose RealTime C++ Reference* for more information about how to use the new component types.

## Directory Structure

The code generation directory structure has changed, it is now:

```
<component name>
  <build>
  capsule1.exe
  capsule1.obj
  ...
  <src>
    Makefile
    capsule1.dep
    capsule2.dep
    capsule1.cpp
    capsule1.h
  ...
```

## Parameters Available in Transition Code

Within each transition code segment there are two new parameters that are available.

**Note:** The `msg` variable is still available in transition code and capsule operations.

**rtdata:** This is the equivalent of the `RTDATA` macro. It is the data sent with the message cast to the data type specified in the protocol for the incoming signal. The `rtdata` parameter is cast to the lowest common superclass of the possible data classes for the given code segment.

```
int level = *rtdata;
```

**Note:** Models which used `RTDATA` do not have to change. `RTDATA` and `rtdata` are equivalent.

If a transition is triggered by multiple signals with different data classes, you will have to cast `msg->data` yourself.

```
int level = *(const int *)msg->data;
```

**rtport:** This is a pointer to the port cast to the appropriate protocol type, on which the message that triggered the transition was received. You can use this parameter to reply to messages. See *Message Reply* on page 123.

## Port Cardinality Cannot be Unspecified

Because there is no way to resolve unspecified cardinalities between libraries, capsule role replication cardinalities cannot be left unspecified as *'\**'. You should use constants to specify replication values.

## Makefile Override Changes

Previously the makefile override property was set to a file name which contained a makefile fragment which was to be included into the main makefiles with an **include** statement. Now the makefile overrides property is added, as is, to the makefile. That means that you don't have to create a separate file outside of the toolset to contain any additional makefile commands.

Previous models which contain makefile overrides are converted by adding the include statement to the property.

## Model Properties

Component build settings are now stored in model properties. This allows easy extensibility and sharing of build options. Although the actual build properties have not changed much, they have been re-arranged. Build options now exist for each component type and for generic generation and compilation settings.

Component type properties: C++ Executable, C++ Library, C++ External Library.

Generic build settings: C++ Generation, C++ Compilation

See the *Rational Rose RealTime C++ Reference* for descriptions of the component model properties.

## Advanced property Editors

A number of properties introduced in this release require more than simply a true or false value. Instead some properties represent a set of parameters. To assist configuring properties that have several parameters that can be set, graphical editors

have been added to property sheets to allow editing of these complex properties. If a property has an advanced property editor you will notice an **Edit...** or **Select...** button beside the property. Press the button to access the extended property editor window.



## Contents

This chapter is organized as follows:

- *Overview* on page 135
- *Configuration Management (CM) Tools Integration* on page 135
- *Requirements Management Tools Integration* on page 137
- *Unit Testing Tools Integration* on page 137
- *Microsoft Development Environment* on page 138
- *Integration with Rational Robot* on page 138
- *Naming Directories* on page 139

## Overview

---

Rational Rose RealTime can coexist on the same workstation with any Rational or ObjecTime product. In addition Rational Rose RealTime is shipped with "out-of-the-box" integrations with several popular development tools. It will simplify tool-chain complexity by providing teams with seamless integration to leading real-time operating systems, compilers, symbolic debuggers, and other market-leading Rational Software products. For a list of supported platform "line-ups", see *Referenced Host Configurations* on page 16.

## Configuration Management (CM) Tools Integration

---

The following CM tools are supported with integration for Rational Rose RealTime. For more information on integrating these tools, see the *Guide to Team Development*.

Tools	Version
Rational ClearCase (Base and UCM)	3.2.1 (requires patch 10), 4.0, 4.1, 5.0, 6.0
Microsoft Visual SourceSafe (NT, 2000 and XP Pro only)	5.0 and 6.0

Tools	Version
RCS (UNIX only)	5.7
SCCS (UNIX only)	5.6 on Solaris
PVCS Integration	6.5

## ClearCase on a UNIX Server and Clients on both NT and UNIX

You can access a ClearCase server on UNIX with Rational Rose RealTime clients running on both NT and UNIX workstations. For more information on integrating these tools, see the *Guide to Team Development*.

## Migrating from Rational Rose and ObjecTime Developer

In order to migrate models into Rational Rose RealTime from either Rational Rose or ObjecTime Developer where models were previously stored in a configuration management system, the model must be brought into the Rational Rose or the ObjecTime Developer tool and written to a single file. Please refer to *Migration* on page 99.

When importing a model from Rational Rose into Rational Rose RealTime, you are encouraged to resolve any model errors in Rose (**Tools > Check Model**) before trying to import the model. It is important to fix unresolved references. In general, Rational Rose is not concerned with unresolved references; however, they are very important in Rational Rose RealTime as they can result in incomplete code generation and compilation errors.

In order to export the ObjecTime model in a format that is readable by Rational Rose RealTime, a patch must be applied to the 5.2 or 5.2.1 toolset to format the file in a single linear form file with all the required information. The patch is available from Rational Customer Support for both the 5.2 and 5.2.1 product release only. Please contact the Rational Customer Support group for further information.

After the model is imported into Rational Rose RealTime, it can then be stored in the configuration management system.

**Note:** RRT\_Export patches are available on the Rational web site.

## Requirements Management Tools Integration

---

The following tools are supported for integration with Rational Rose RealTime.

---

Tools	Version
Rational SoDA for Word (NT only)	2000.02.10 and later
Rational RequisitePro	2000.02.10 and later

---

### Rational SoDA for Word

SoDA and Rational Rose RealTime will work together out of the box if installed from the Suite. Rational Rose RealTime offers the same level of SoDA integration as Rose. For information on how SoDA and Rational Rose RealTime integrate, see the Rational Rose integration section in the SoDA documentation.

Please refer to the product support page at

<http://www.rational.com/support>

for the latest updates on SoDA integration.

**Note:** To generate a report using SoDA, the Rational Rose RealTime model must have been saved at least once. If the Rational Rose RealTime model has never been saved, it will be untitled. An untitled model will cause SoDA to generate errors.

### Rational RequisitePro

RequisitePro and Rational Rose RealTime will work together out of the box if installed from the Suite. Rational Rose RealTime offers the same level of RequisitePro integration as Rose. For information on how RequisitePro and Rational Rose RealTime integrate, see the Rose integration section in the RequisitePro documentation.

## Unit Testing Tools Integration

---

The following tools are supported for integration with Rational Rose RealTime.

---

Tools	Version
Purify for UNIX and Windows NT	2001.03.00 and later

---

## Rational Purify

After a component is built and a component instance has been created, the instance can then be run and observed. Purify detects errors in model code as well as the model. For information, see the Running and Debugging section in the *Rational Rose RealTime Toolset Guide*.

### Adding Options to Purify on UNIX

Occasionally, you may need to add options during a Purify'd build on UNIX.

Options can be added by changing **PURIFY\_OPTIONS** in the **CompilationMakeInsert** field of the executable component.

The default value of **PURIFY\_OPTIONS** (generated in the Makefile by the code generator) is:

```
PURIFY_OPTIONS = -log-file=$(BUILD_TARGET).txt -windows=no
```

To accommodate using **g++**, you can add the following:

```
PURIFY_OPTIONS = -log-file=$(BUILD_TARGET).txt -windows=no  
-collector=/usr/lib/gcc-ld -g++=yes
```

Where the path of to the collector, **gcc-ld** in most cases, should be the path that is specific to your environment.

For proper integration of Purify when running the Purify'd executable from the toolset, you should preserve the default options.

For an explanation of Purify options, see *Running a component instance with Purify* in the *Toolset Guide*.

## Microsoft Development Environment

---

We recommend that you install the latest service packs available from Microsoft for Visual Studio or Visual C++.

## Integration with Rational Robot

---

Installing the 2003.06.00 release of Rational Rose RealTime will interfere with the operation of the 6.1 release of Rational Robot.

We recommend that you upgrade to the 2003.06.00 release of Rational Robot.

## Naming Directories

---

Avoid using spaces in directory names if you plan to integrate with Tornado, OSE or VRTX embedded operating systems. For additional information, see the Technical Notes in the Customer Support section on our web site at:

<http://www.rational.com/support>



## Contents

This chapter is organized as follows:

- *Starting Rational Rose RealTime on Windows* on page 141
- *Starting Rational Rose RealTime on UNIX* on page 141
- *Rational Rose RealTime for UNIX and the X Window System* on page 142
- *Automating Rational Rose RealTime* on page 144
- *Command Line Options* on page 145

## Starting Rational Rose RealTime on Windows

---

To start Rational Rose RealTime on Windows, on the **Start** menu, choose **Programs > Rational Software > Rational Rose RealTime**.

**Note:** You must first install license keys before running Rational Rose RealTime.

Temporary license keys can be found in the product package. Instructions on how to request permanent license keys, see *Installing License Keys* on page 89.

For additional information on configuring your environment variables, see *Configuring Your Environment* on page 47.

## Starting Rational Rose RealTime on UNIX

---

You can start Rational Rose RealTime from a UNIX command shell prompt by typing:

```
RoseRT
```

You can also use the following to start Rational Rose RealTime on Unix:

```
RoseRT -recreate_registry
```

Setting the **-recreate\_registry** option creates a default registry.

## Start-up Options for UNIX

### **-regedit**

Edits the internal registry that maintains mappings of directory names and other information required by the Rational Rose RealTime tool. This registry mimicks the function of the WindowsNT registry, except that on UNIX the registry is maintained directly by Rational Rose RealTime.

### **-startuplicense**

Creates a startup license.

### **-recreate\_registry**

Creates a default registry, throwing away any changes made through the -regedit option.

### **-q | -quiet**

Limits the output of the tool on startup.

### **-v | -verbose**

Provides verbose output on startup.

### **-cleanup**

Kills all running applications using MainWin and then cleans up the x-server resources.

You should be very careful with this command as it will kill all MainWin applications running under your Id.

**Note:** If your **last** Rational Rose RealTime session ended unexpectedly (by crashing), always use this option.

## Rational Rose RealTime for UNIX and the X Window System

---

When running on UNIX platforms, Rational Rose RealTime relies on the X Window System to provide basic user interface services. Rational Rose RealTime supports the most common versions of the X Window System: Version 11 Release 5 and Version 11 Release 6.



The following topics provide background information on how Rational Rose RealTime interacts with the X Window System and highlights any specific requirements.

## **X Clients**

The X Window System employs a network-enabled client-server architecture. Rational Rose RealTime is a client application within this architecture. X clients interact with the user via an X server which may or may not be running on the same system as the client application. If the server and client are not running on the same system, the X client is said to be using a remote display.

## **X Servers**

The X server is a program that controls interaction between the user and an X client application via the keyboard, mouse and graphical display screen. The X server runs locally on the system where the display is attached.

On UNIX workstations the X server is normally provided by the system vendor. If you want to run Rational Rose RealTime on a UNIX workstation and remotely display it on a Windows workstation, a third-party X server (such as, Hummingbird Exceed) is required. Rational Rose RealTime has been qualified to be used with Hummingbird Exceed 6.1.

## **X Window Managers**

The X window manager is a special X application that facilitates running multiple X clients within separate windows on a single X server. The window manager provides mechanisms for resizing and moving windows and designating which X client has input focus at a given time.

Most X environments include a window manager. Rational Rose RealTime supports most commonly used window managers including:

- Common Desktop Environment (CDE)
- Motif (MWM)
- Exceed native window manager

**When available, the CDE window manager is recommended.**

## Input Focus (Active Window) Policy

The X window manager often allows the user to specify a policy for delegating input focus. This window is also referred to as the active window. There are two common settings:

- Click to focus. In this mode, the user must click on a window with the mouse to give it input focus. This is most consistent with the Windows focus policy and is the recommended configuration.
- Point to focus. In this mode, the user points to a window with the mouse to give it input focus.

## Window Order Policy

When using the CDE window manager, to ensure that the proper Secondary and Transient window policy is in effect, set the following environment variable in `.Xdefaults`:

**`dtwm*secondariesOnTop: True`**

Setting this XResource to True in each user's Xdefaults file ensures dialogs are always stacked above their associated primary windows. Not using this XResource can result in the secondary window, such as an external editor, getting caught behind the primary window and resulting in the user's inability to regain focus of the secondary window.

When using CDE as your XWindow manager, please note that the **Allow Primary Windows on Top** and **Raise Windows When Made Active** options are enabled by default. These options should be disabled when setting the `dtwm*secondariesOnTop` option to True.

## Automating Rational Rose RealTime

---

Rational Rose RealTime can be programmed to automatically perform a wide variety of tasks through the Rational Rose RealTime Extensibility Interface (RRTEI). The RRTEI is accessible through Basic scripts and from COM automation clients. This interface can be used to create add-ins and scripts. Rational Rose RealTime also supports the Rose Extensibility Interface (REI) for compatibility with Rose. The complete documentation for the RRTEI is included in the Rational Rose RealTime Online Help System.

Running Rational Rose RealTime as an automation server consumes a license when the application is made visible.

## Command Line Options

---

The following are command line options for Rational Rose RealTime:

### <filename>

A user option to load a model on startup.

**Note:** To ensure that a file name containing spaces is processed properly, on the command line, the file name must contain a combination of single and double quotation marks. The file name will require two levels of quotation marks so that the spaces in the file name are not interpreted as space characters, and to ensure that the file name and path are passed as a single argument to the Rational Rose RealTime executable. The <filename> parameter is quoted as follows:

```
<open_single_quote><open_double_quote>path/filename<close_double_quote><close_single_quote>
```

For example, if you have the following path and file name:

```
My Models/My working model.rtmml
```

you would use the following to invoke the toolset from the command line:

```
RoseRT "My Models/My working model.rtmml"
```

### -nologo

A user option to suppress the logo screen on startup.

### -emulateREI

A user option to enable the Rose Extensibility Interface (REI). Overrides the settings in tools/options.

**Note:** The Rational Rose RealTime Extensibility Interface (RRTEI) is still available.

### -noEmulateREI

A user option to disable the Rose Extensibility Interface (REI). Overrides the settings in tools/options.

**Note:** The Rational Rose RealTime Extensibility Interface (RRTEI) is still available.

### -register or -regserver

Enters the applications registry settings into the registry.

**-unregister or -unregserver**

Removes the applications registry settings from the registry.

**-runScriptAndQuit**

Use in conjunction with a compiled script passed as parameter. When the toolset is launched with this command line option, the toolset starts hidden, runs the script and quits. All of this is done without consuming a license. This is particularly useful to allow batch mode builds.

## Contents

This chapter is organized as follows:

- *Web Publisher* on page 147
- *Model Integrator* on page 149
- *Rose C++ Analyzer* on page 150

## Web Publisher

---

Web Publisher enables you to create a web-based (HTML) representation of a Rational Rose RealTime model, which others can view using a standard browser such as Netscape Navigator or Microsoft's Internet Explorer.

Unlike sequential formats, such as paper or text files, Web Publisher lets you non-sequentially browse, search, and navigate your design. You can publish successive iterations of an evolving model for review or for sharing information. Another potential use is to publish documentation for a frozen API or framework.

Web Publisher recreates model elements, including diagrams, classes, packages, relationships, attributes, and operations. Once published, hypertext links enable you to traverse the model much as you would in Rational Rose RealTime.

You can control what Web Publisher includes by setting a variety of options. For example, you can select which packages of a model are published, the amount of detail to include, the notation to use, and the graphics format for diagrams. The View feature lets you launch your default browser and view the published model directly from Web Publisher.

## Suggested Workflow

Follow these steps to generate the files needed to create a web-based version of a Rational Rose RealTime model:

- 1 Open the model you want to publish.
- 2 Select **Tools > Web Publisher**.

- 3 From the Web Publisher dialog, select the publishing options you need.  
**Note:** The dialog displays the options that were selected the last time a model was published.
- 4 Click **Publish** when you are ready to publish the model.
- 5 Use **View** to open your default web browser and view the published model.  
Remember that in the future you can open the published model in the browser by opening the *root file name* you specified on the Web Publisher dialog.
- 6 Click **Close** to close the dialog.

## Limitations

The following browsers are supported:

- Microsoft Internet Explorer 5.5 or better. ([www.microsoft.com](http://www.microsoft.com))
- Netscape Navigator 4.72 or better. (<http://www.netscape.com/download>) If you want to publish the images in PNG format you need to add PNG support to Netscape Communicator. PNG Live (<http://codelab.siegelgale.com/solutions/pnglive2.html>) is a plug-in that provides PNG support for Netscape Communicator. Netscape Communicator 4.5 or better has built-in support for PNG and therefore does not require any special plug-in to view web pages created by Web Publisher. ([www.netscape.com/download](http://www.netscape.com/download))
- Only eight colors are directly supported in published diagrams. Other colors are obtained by dithering. If you want to avoid dithering, set up Rational Rose RealTime to use line and fill colors that are among the eight available.

The following table includes the eight available colors and their RGB values.

Red	255	0	0
Green	0	255	0
Blue	0	0	255
White	255	255	255
Black	0	0	0
Yellow	255	255	0
Magenta	255	0	255
Light Blue	0	255	255

- In published diagrams, you can normally click on a model element to go to that model element's specification information. This does not work for some model elements. These include aggregation relationships on the class diagram, transitions on the state diagram, association roles on the collaboration diagram, and connections on the deployment diagram.

For more information consult the Web Publisher online help.

## Model Integrator

---

The Rational Rose RealTime Model Integrator add-in allows you to compare up to seven units/models - called contributors - to a common root model/units - called the base contributor.

The add-in exists as a separate executable that can be launched stand-alone or from the toolset using **Tools > Model Integrator**. It is launched by the toolset when using the **Source Control > Show Differences**.

It is capable of acting as a ClearCase Type Manager, meaning that ClearCase uses Model Integrator for showing differences and merging Rational Rose RealTime units/models.

### Suggested Workflow

#### Merging two branches of a model

Assuming a base model B and two models C1 and C2, having B as their common historical ancestor.

From Rational Rose RealTime, select **Tools > Model Integrator** to launch Model Integrator

#### From Model Integrator:

- 1 Select **File > Contributors** to open the **Contributors** dialog.
- 2 First enter the base contributor B, then the two other contributors C1 and C2.
- 3 Click **Merge**.

For each contributor, Model Integrator loads the first level of subunits and brings up the subunits dialog.

- 4 Press **OK** to load all subunits.

Model Integrator now shows the merged model potential conflicts.

- 5 Resolve each conflict by selecting the contributor to use for that conflict. To see model differences, select **Options > Compare Model**.
- 6 When all conflicts are resolved, select **File > Save As** and choose a file name.
- 7 In the subunits dialog that follows, click **OK**.

### **Comparing local unit with the one in source control database**

From Rational Rose RealTime, select the unit to compare in the browser. Open the context menu and select **Source Control > Show Differences**.

For more information consult the Model Integrator online help.

## **Rose C++ Analyzer**

---

The Rose C++ Analyzer is an executable bundled with Rational Rose 2000's Rose C++ add-in. Used in conjunction with the **Tools > Import** menu command, it provides a way to import legacy C++ systems into Rational Rose RealTime.

Rational Rose RealTime only supports the initial reverse engineering since the code is embedded within its model. Full target observability from the toolset is supported, thus eliminating the need to update code outside the toolset environment.

**Note:** The online help for the Rose C++ Analyzer contains Rose 2000 specific information that may not be applicable to Rational Rose RealTime. We suggest you limit your use of the add-in to the Suggested Workflow described below.

### **Suggested Workflow**

From Rational Rose RealTime, select **Tools > C++ Analyzer** to launch Analyzer.

#### **From Rose C++ Analyzer:Create Project:**

- 1 Set compiler settings.
- 2 Add Files.
- 3 Analyze.
- 4 Code Cycle.
- 5 Export to Rose.

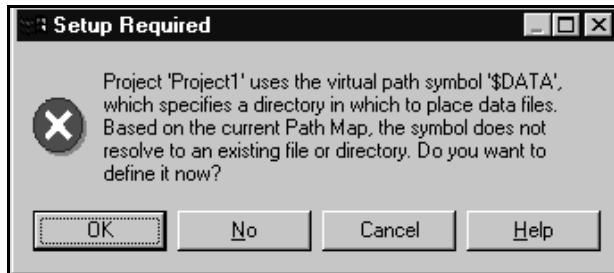
#### **From Rational Rose RealTime:**

- 1 Select **File > Open** to load the Rose Model.
- 2 Select **Tools > Import Code** to import code from source files.

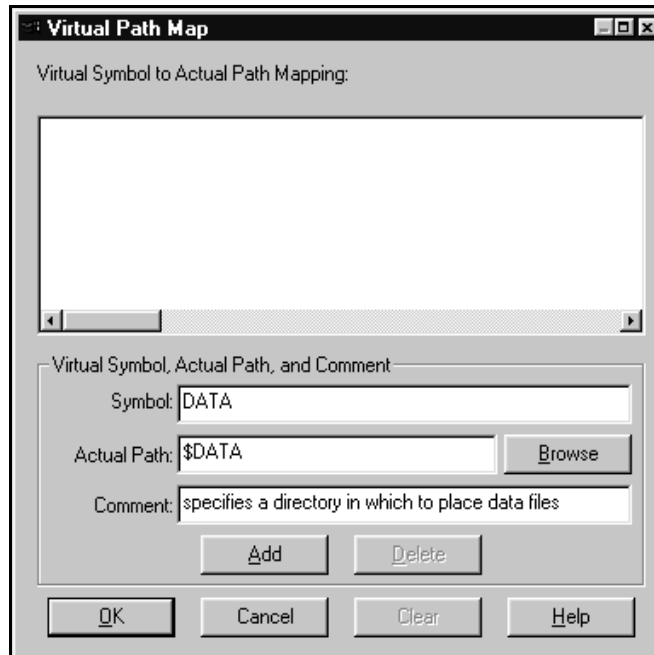


## Notes

- When you create a Rose C++ Analyzer project for the first time, the following message prompts you to define the \$DATA/Rose pathmap symbol:



Click **OK** to bring up the following dialog:



In the **Actual Path** field, enter an existing path where the Rose C++ Analyzer will store information about analyzed source files. Click **Add** and then **OK**.

- **Windows** users: You may not get this dialog if Rational Rose 2000 is already installed on your machine. In this case, the Import Code window appears.
- **UNIX** users: The default pathmap symbol \$DATA/ must be replaced with \$DATA.

## Limitations

- C++ capabilities are limited by Rational Rose RealTime's code generator's own limitation, for example, C++ templates, namespaces
- Round-trip engineering is not supported (and not needed).
- Pathmap functionality is not supported (and not needed).

For more information consult the Rose C++ Analyzer online help.

## Contents

This chapter is organized as follows:

- *Windows* on page 153
- *UNIX* on page 153

## Windows

---

**To uninstall Rational Rose RealTime from a Windows machine:**

- 1 Click **Start > Settings > Control Panel**.
- 2 Double-click **Add/Remove Programs**.
- 3 Select Rational Rose RealTime and click **Change/Remove**.

Follow the instructions on your screen to remove Rose RealTime.

**Note:** We recommend that you also remove the Rose RealTime directories and registry settings from your system after uninstalling Rational Rose RealTime. These directories are:

```
HKEY_CURRENT_USER\Software\Rational Software\Rose RealTime  
HKEY_LOCAL_MACHINE\Software\Rational Software\Rose RealTime
```

## UNIX

---

**To uninstall Rational Rose RealTime from a UNIX machine:**

- 1 Save any user data files in another location before removing the installation directory.
- 2 Remove the installation directory and all of its contents.



## Contents

This chapter is organized as follows:

- *Overview* on page 155
- *Rational Connexis* on page 155
- *Rational Quality Architect - RealTime Edition* on page 157

## Overview

---

The following is a list of the problems and limitations in Rational Rose RealTime products at the time of release. Some of these problems may have been addressed by the time you read this. For the most recent list of known problems and any fixes that may have been published, please visit the Rational Rose RealTime web site at:

<http://www.rational.com/support/>

Select **Upgrades, Patches, and Service Releases**, then select **Rational Rose RealTime**.

## Rational Connexis

---

The following topics describe the troubleshooting, known issues, and updates for Rational Connexis version 2003.06.00:

- Troubleshooting
  - Transport Integration Framework
  - Turning Off Auditing for a Single Transport is Not Recommended
  - Signals No Longer Supported

## Troubleshooting

### Transport Integration Framework

When implementing your Transport Integration using the Transport Integration Framework (TIF), you must create a component to build the Transport Integration. The Transport Integration component requires an additional inclusion path for the TargetRTS.

#### Add an inclusion for:

`$(ROBERT_HOME)/C++/TargetRTS/src/target/<your target>`, where `<your target>` maps to the target you are building.

#### Example:

If building for "NT40," the inclusion path would be:

`$(ROBERT_HOME)/C++/TargetRTS/src/target/NT40`

### Turning Off Auditing for a Single Transport is Not Recommended

You may not be able to turn off auditing for a single transport. This is because setting `-CNXtap=0` will result in no auditing taking place for the CRM and CDM transports. The internal timer used by the transporter will be set to 1 second. This may affect the `-CNXtrre` and `-CNXtbrd` parameters which are rounded up to a multiple of `-CNXtap`. Any transports integrated with an audit type other than "No Audit," will find the `-CNXtcapi` and `-CNXtcapo` periods to be a multiple of 1 second. Setting `-CNXtap` to be `> 1` second will affect the frequency at which metrics can be collected by the viewer and other DCS enabled applications.

### Signals No Longer Supported

The `RTDDCSRunning` and the `RTDDCSrunningReply` signals are no longer supported in this version of Connexis. The recommended fashion for determining if DCS is running, is to wait for the `rtBound` signal, indicating that a successful connection to the `RTDInitStatus` publisher has been made.

# Rational Quality Architect - RealTime Edition

---

The following topics describe any troubleshooting, known issues, and update information for Rational Quality Architect - RealTime Edition version 2003.06.00:

- Target Observability Behavior When the Model is Modified
- Running Verify Behavior with Eighty or More Sequence Diagrams (UNIX)
- Driver Methods for Sending Messages to the Log and Custom Comparison
- Lost Information in To Port for a Message
- Do Not Use -runScriptAndQuit When Running RQART From a Script
- Creation of Container Capsules
- Converting MSCs in Rational Rose RealTime Using the RQA-RT
- Creating Messages and Sequence Diagrams
- Sending Message Specification Data Field Format for Java
- Customizing a Sequence Diagram Created From a Trace
- RQA-RT Limitations

## Target Observability Behavior When the Model is Modified

Target observability was modified to change the way it reacts when it detects a model change. Previously, a message box with the text "Target observability session terminated due to model change." would appear and the component instance would either be detached or shut down. The new behavior is to continue execution. A modeless dialog box will appear and contain the text "Target observability has detected a model change. Execution is continuing. Do you wish to detach/shutdown this component instance?". If you do nothing or click **Continue**, execution will continue and additional model changes will be ignored. If you click **Detach/Shutdown**, the component instance will be detached. If the component instance is the instance used to start the run, and the check box **Attach to target on startup** on the **Detail** tab of the Component Instance specification dialog is selected, an attempt will be made to shut down the target. If that attempt is successful, any other component instances attached to the same target are also detached.

## Running Verify Behavior with Eighty or More Sequence Diagrams (UNIX)

If you want to select Verify Behavior for a model that contains eighty or more Sequence diagrams, ensure that they have at least 32MB of stack space available ( use **limit stacksize** in csh, and **ulimit -s** in sh).

## Driver Methods for Sending Messages to the Log and Custom Comparison

RQA-RT includes two helper functions in `RQARTAbstractTestWrapper`, the capsule that is the superclass for all generated drivers. These functions are:

`SendACompareFailure(<some_string>);LogAMessage(<some_string>);` Both of these functions send a message to the RoseRT log that is hyperlinked to the appropriate message in the trace Sequence diagram. `SendACompareFailure` also causes the sequence diagram differencing algorithm to fail on that message. These functions can be called in any user-specified code in a Sequence diagram. This includes:

Local action code blocks from the "Quality Architect - RT" tab of the Local Action Specification.

The "Sender Driver Test Code" and "Receiver Driver Test Code" code blocks available from the "Quality Architect - RT" tab of the Send Message Specification".

This code will be inserted on the appropriate transition in the generated driver capsule. If you want to see exactly where it is inserted, use the Find In feature to search for some identifiable piece of the code (possibly a unique string in a comment) in the generated driver capsule.

## Lost Information in *To Port* for a Message

If you load an LF-file into Rational Rose RealTime and perform the conversion, there are a number of instances where the

information on the *To Port* of a message is lost. In Rational Rose RealTime 2003.06.00, the conversion was enhanced to determine the receiver port on any message sent between two instances that have a direct logical connection from the sender port. This calculation increases the time required to perform the conversion. Diagrams which have messages between instances that skip relay ports will continue to require the receiver port to be entered manually.

## Do Not Use `-runScriptAndQuit` When Running RQART From a Script

When running RQART from a script using the `RunVerifyBehavior` method do not use the `-runScriptAndQuit` command line option to cause RoseRT to exit once the script is complete. Since Verify Behavior runs asynchronously after `RunVerifyBehavior` is called this will cause RoseRT to exit before Verify Behavior is complete. Use the `szScriptOnCompletion` parameter of `RunVerifyBehavior` to specify a script that contains the actions that you want to occur after Verify Behavior is complete. This script can exit RoseRT using the `Exit` method.



## Creation of Container Capsules

RQA-RT does not automatically create container capsules for a nested capsule when the container is not included in the Sequence Diagram.

## Converting MSCs in Rational Rose RealTime Using the RQA-RT

### Problem

Many MSCs have a variable of the same name (prepareSetupReqD), but they can have a different type. When RQA-RT synthesizes these attributes from the multiple driver instances and attempts to generate one test driver, there is a name conflict and it selects the last Sequence Diagram attribute's type as the type for the attribute of the test driver class.

### Background

In ObjecTime Developer, attributes are a characteristic of MSCs; each MSC can have its own attributes. These attributes can be considered variables for the environment which acts as a driver in TestScope.

When the MSCs are converted in Rational Rose RealTime using the RQA-RT conversion tool, each MSC is converted to a Sequence Diagram. In this Sequence Diagram, what was the environment in ObjecTime Developer is now a driver capsule that is automatically created. The interaction instance (of the driver) in the sequence diagram has attributes which were converted from the MSC in ObjecTime Developer.

This newly created interaction instance is set as a driver during the Verify Behavior operation. As a result, a new RQADriver is created.

### Example:

Seq1:: driver has an attribute xAttrib of type Xtype

Seq2:: driver has an attribute xAttrib of type Ytype

Test harness generates a driver class with an attribute xAttrib of type Xtype or Ytype. The compilation occurs and the result will be many errors in the test harness. The errors occur because some of the action code interprets xAttrib as type Xtype and some as type Ytype.

### Workaround

Run the Sequence Diagrams from separate test harnesses. This means that you will have completely separate components and will have to compile again.

## Creating Messages and Sequence Diagrams

The Item Properties in the Create Message Specification dialog, with the exception of Thread, only apply if you are creating a Capsule Under Test (CUT). Thread applies in every case. The Item Properties entries are as follows:

- Capsule class - a capsule name. Enter a value here only if you want to override the default capsule class associated with the role that is associated with the interaction instance being created.
- Initial data - Enter a value here only if your Capsule Under Test (CUT) requires data on startup. To provide data you must specify an attribute in the driver sending the create message with the appropriate initial values. You cannot provide initial data in create messages sent from the environment.
- Data Descriptor - if you provided initial data, you must specify the type descriptor of the type of data that you specified. This is in the format `RTType_<type of initial data>`
- Thread name

## Sending Message Specification Data Field Format for Java

For the data to be passed with the messages, the data type should be specified as `<data_type> <constructor_arguments>`. For example, to pass Integer object referring to the number 5, place the following:

Integer 5 or `java.lang.Integer 5`

If `<data_type>` is omitted, Integer is assumed.

The `<constructor_arguments>` should contain the exact line to be passed as an argument to constructor. For example, for the `MyObject(Integer, String)` the following line can be used: `MyObject 5, "Acme"`

**Note:** No additional brackets or quotes need to be placed around `<constructor_arguments>`.

## Customizing a Sequence Diagram Created From a Trace

To customize a Sequence diagram created from a Trace do not delete columns from the Trace window. This may cause Rational Rose RealTime to crash. Instead, create the Sequence diagram from the full Trace window, and delete unwanted instances from the Sequence diagram.

## RQA-RT Limitations

- RQA-RT does not support C models.
- Only leaf node instances in the interaction/sequence diagram can be specified as a driver/stub.
- Within a specification sequence diagram, each interaction instance without a specified role must have a unique name. Only one interaction instance per test set can be left unnamed.
- When running multiple specifications in a test, a port on a capsule under test (CUT) can only be connected to one interaction instance with an unspecified role for the set of specifications.
- An interaction instance with an unspecified role cannot have a cardinality index. Care should be exercised using unspecified roles to test capsules and/or ports with cardinality greater than 1.
- If the generated trace sequence diagram is manually compared to the specification sequence diagram, it is important to remember that such comparison is asymmetric - the specification sequence diagram should always be selected before the trace diagram.
- When running multiple specifications in a test, remember that the capsules under test continue to execute between tests. Ensure that each sequence diagram becomes quiescent at the end of each test scenario.
- If you generate a new harness into a controlled package which isn't checked out warnings will be generated.
- In Java, replicated subcapsules can only be specified as drivers if the replication index is 1.
- Interaction instances with unspecified roles cannot be used to simulate interactions with sub-capsules. An error will occur if an unsupported use of "unnamed" interaction instance is detected.

**Workaround:** For example: top capsule A has capsule role B in it's structure. Capsule B has capsule role C in it's structure. If you run a test on the Sequence Diagram located under A's Structure Diagram, "unnamed" interaction instances can communicate with the B, but not C. To use "unnamed" interaction instance with the C, a Sequence Diagram should be located under B's Structure Diagram.

**Note:** "unnamed" interaction instances are not a mandatory part of the verification process.

- If a capsule role has a cardinality greater than 1, the cardinality index should be specified on interaction instances.
- RQA-RT support of unspecified interaction instances allows you to easily drive an unconnected port without having to add a driver capsule to the collaboration. More complex tests will require the use of a driver capsule. Regardless, you cannot drive a test using a port that is part of the system currently being tested.
- Verify Behavior in RQART always performs differencing on any test completed, even if the entire test run did not finish.
- The Environment in a sequence diagram is an InteractionInstance, just like all the other InteractionInstances in the diagram. The only RQART property currently used for the Environment is the **Minimum Run Time**, which allows the you to specify a minimum runtime for the entire test.

## Contents

This chapter is organized as follows:

- *Submitting Problem Reports* on page 163
- *Submitting Feature Requests* on page 164
- *Submitting Support Requests* on page 165
- *Contacting Rational Customer Service by Email or Telephone* on page 166

This chapter describes how to submit problem reports, feature requests and support requests to Rational Customer Service.

## Submitting Problem Reports

---

With Rational Rose RealTime, you can email problem reports to the Rational Customer Service department that services your location. When you email a problem report directly from the Rational Rose RealTime application, a wizard guides you through the process, ensuring that you provide the correct information to the Rational Customer Service team. This information includes contact and location information, and a detailed description of the problem that you are reporting.

### To submit a problem report:

- 1 From the **Help** menu, click **Email Technical Support**.

A submenu appears, providing you with three options.

- 2 Click **Problem Report**.

The **General Information** dialog appears.

- 3 Type your contact and location information in the text areas provided and click **Next**.

The **Problem Report - Additional Information** dialog appears.

- 4 In the **Defect Title** text area, type a detailed name for the problem that you are reporting.

- 5 Select the type of problem that you are reporting from the appropriate list boxes.

- 6 Describe the problem, using the categories provided in the **Details** area.
- 7 Click **Next**.  
The **Email Summary** dialog appears.
- 8 Ensure that the information that appears in the Email Summary dialog is accurate.  
**Note:** The email information displayed in the **Technical Support Email Address** is chosen based on the location information that you provided in the General Information dialog. It is not recommended that you edit the email address.
- 9 If you want to save or print a copy of the email, click the appropriate button.
- 10 Click **Send Email** to send your email.

## Submitting Feature Requests

---

With Rational Rose RealTime, you can email feature requests to the Rational Software Customer Service department that services your location. When you email a feature request directly from the Rational Rose RealTime application, a wizard guides you through the process, ensuring that you provide the correct information to the Rational Software Customer Service department. This information includes contact and location information, and a detailed description of the feature that you are requesting.

### To submit a feature request:

- 1 From the **Help** menu, click **Email Technical Support**.  
A submenu appears, providing you with three options.
- 2 Click **Feature Request**.  
The **General Information** dialog appears.
- 3 Type your contact and location information in the text areas provided and click **Next**.  
The **Feature Request - Additional Information** dialog appears.
- 4 In the **Request Title** text area, type a detailed name for the Feature that you are requesting.
- 5 Select the level of urgency for the feature that you are requesting.
- 6 Describe the feature, using the categories provided in the **Details** area.

- 7 Click **Next**.

The **Email Summary** dialog appears.

- 8 Ensure that the information that appears in the **Email Summary** dialog is accurate.

**Note:** The email information displayed in the **Technical Support Email Address** is chosen based on the location information that you provided in the General Information dialog. It is not recommended that you edit the e-mail address.

- 9 If you want to save or print a copy of the email, click the appropriate button.
- 10 Click **Send Email** to send your email.

## Submitting Support Requests

---

With Rational Rose RealTime, you can email Support requests to the Rational Software Customer Service department that services your location. When you email a Support request directly from the Rational Rose RealTime application, a wizard guides you through the process, ensuring that you provide the correct information to Rational Customer Service department. This information includes contact and location information, and a detailed description of the support request that you are submitting.

### To submit a support request:

- 1 From the **Help** menu, click **Email Technical Support**.

A submenu appears, providing you with three options.

- 2 Click **Support Request**.

The **General Information** dialog appears.

- 3 Type your contact and location information in the text areas provided and click **Next**.

The Support Request - Additional Information dialog appears.

- 4 In the **Request Title** text area, type a detailed name for the request that you require.
- 5 Select the level of urgency for the question with which you need help.
- 6 Type your question in the **Question** text area.
- 7 Click **Next**.

The **Email Summary** dialog appears.

- 8 Ensure that the information that appears in the **Email Summary** dialog is accurate.  
**Note:** The email information displayed in the **Technical Support Email Address** is chosen based on the location information that you provided in the General Information dialog. It is not recommended that you edit the e-mail address.
- 9 If you want to save or print a copy of the email, click the appropriate button.
- 10 Click **Send Email** to send your email.

## **Contacting Rational Customer Service by Email or Telephone**

---

When contacting Rational Customer Service by email or by telephone, please be prepared to supply the following information:

- Name, telephone number, and company name
- Product name and version number
- Operating system and version number (for example, Windows NT 4.0, Windows 2000, Windows XP, Solaris 2.6, 2., 2.8, and 2.9)
- Computer make and model
- Your service request id (if you are calling about a previously reported problem)
- A summary description of the problem, related errors, and how it was made to occur

If your organization has a designated, on-site support person, please try to contact that person before contacting Rational Customer Service.

You can obtain technical assistance by sending electronic mail to the appropriate email address. Electronic mail is acknowledged immediately and is usually answered within one working day of its arrival at Rational. When sending an email place "Rational Rose RealTime" in the subject line, and in the body of your message include a description of your problem.

When sending email concerning a previously-reported problem, please include in the subject field: "[ SR#XXXXXXXX ]", where XXXXXXXXX is your Service Request number. For example:

```
[SR#111222333] Rational Rose RealTime installation issues
```

Sometimes Rational Customer Service engineers will ask you to fax information to help them diagnose problems. You can also report a technical problem by fax if you prefer. Please mark faxes "**Attention: Customer Service**" and add your fax number to the information requested above.



Telephone, fax, and email information for Rational Customer Service are Table 11. If you have problems or questions regarding licensing, please see *License Support Contact Information* on page 167.

**Table 11 Telephone and Fax and Email Information**

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

## License Support Contact Information

If you have a problem or questions regarding the licensing of your Rational Software products, please contact the Licensing Support office nearest you.

Telephone numbers for license support are listed in the following table. Ask for, or select, **Licensing Support**.

**Table 12 License Support Telephone and Fax**

Region	Telephone Number	Fax Number
North, Central, and South America	+1 (800) 433-5444 (toll free) +1 (408) 863-4000 Cupertino, CA	+1 (781) 676-2460
Europe, Middle East, and Africa	+31 20 4546 200	+31 20 4546 202
Asia Pacific	+61 2 9419 0111	+61 2 9419 0123

Email addresses for license support are listed in the following table.

**Table 13 License Support Email**

<b>Region</b>	<b>Email Address</b>
North, Central, and South America	license@rational.com
Europe, Middle East, and Africa	license@europe.rational.com
Asia Pacific	license@apac.rational.com
Asia Pacific (Mainland China, Hong Kong, and Taiwan)	license@china.rational.com
Asia Pacific (Korea)	license@apac.rational.com
Asia Pacific (Japan)	license@japan.rational.com

# Index

## A

- accessing
  - online help 12
- AccountLink 76
- activation process
  - licenses 83
- active window policy (X window system) 144
- adding
  - printer on UNIX 20
- add-ins
  - Model Integrator 149
  - Rose C++ Analyzer 150
- administration commands
  - licensing on Unix 85
- Advanced property editors 133
- Advantages of backwards compatibility 116
- agreement, licenses 30, 36
- Allow sub-capsule instances to be drivers 8
- API Changes 121
- API Changes Summary 121
- asynchronous sends 122
- automating
  - Rational Rose RealTime 144
  - Rose RealTime 144
- Automating Rose RealTime 144

## B

- backwards compatibility
  - advantages 116
  - disadvantages 116
- Backwards Compatibility Mode 115
- BasicTest server output 63
- Batch Files
  - updating 46
- bin (directory) 11

- building
  - Model in Backwards Compatibility Mode 117
- Building a model in backwards compatibility mode 117

## C

- C Language Migration 112
- C++ Analyzer 150
  - Limitations 152
  - Suggested Workflow 150
- C++ Language Migration 115
- C++ UML Services Library 119
- Capsule interface generation 9
- CD-ROM
  - mounting instructions 55
  - unmount 58
- classes 120
- cleanup 142
- ClearCase
  - command line access 48, 59
  - element type 48, 59
  - repository setup 48, 60
  - workstation setup 47, 58
- ClearCase integration 109
- ClearCase on a UNIX Server 136
- ClearCase options
  - Unix 59
  - Windows 48
- Client Installation 29
- Client Installation Tasks 28
- Code browser 100
- Code editors 100
- Code Generation 101, 120, 131
- command line access to ClearCase 48, 59

- Command Line Options 145
- command line options 145
- commands
  - License Manager 80
- compilation 120
- Configuration Management (CM) Tools
  - Integration 135
- configuration requirements
  - UNIX 15
  - Windows 2000 14
  - Windows NT 13
  - Windows XP Pro 14
- configurations
  - host 16
- configuring
  - environment 47
- Configuring a UNIX Workstation to Point to a FLEXlm Server 82
- connectivity 7
- Connexis 7
  - converting models 67
  - converting models from 67
  - Run-Time connectivity viewing 7
  - troubleshooting 155
  - verifying the installation 61
- contacting Rational Customer Service xviii
- contacting Rational Technical Support 9
- Container Capsules 159
- convert an existing Rose RealTime model 112
- converting
  - C++ Model to C 112
  - Connexis model 69
  - connexis models 67
  - global signals to local signals 114
  - Models from Connexis 67
- converting a C++ model to C 112
- Converting MSCs 159
- converting temporary licenses to permanent 78
- creating
  - container capsules 159
  - Executables for Hosts without Toolset
    - Support 18
  - messages 160
  - Sequence diagrams 160

- creating executables 18
- customizing
  - Sequence diagram from a trace 160

## D

- data qualifier 9
- Defer 124
- DEMO FEATURE 87
- determining connectivity 7
- Devices 21
- drivers 8, 9
- Dynamic Forwarding 127

## E

- element type setup
  - Unix 59
  - Windows 48
- ELS 131
- Emergency Keys 75, 87
- Emergency License 75
- emulateREI 145
- Enable BackwardsCompatible protocol
  - property 117
- environment configuration 47
- environment variables
  - configuring for Windows 47
- Evaluation License 75
- exinstal 81, 85
- External Layer Service (ELS 131

## F

- file
  - license 79
- file format changes 108
- fixing unresolved references 102
- FLEXlm
  - application program 82
  - configuring for UNIX 82
- FLEXlm Server 82
- Floating License 74

floating license key for Unix 92  
Forwarding 126  
Forwarding Pattern 127  
Frequently Asked Questions 88

## G

generating  
    executable without a common file system 19  
generating executables 19  
getting help 9

## H

host configurations 16  
Host platform installation 61  
hosts  
    creating executables without Toolset  
        support 18  
how to get help 9

## I

Imgrd 81  
import an ObjecTime Developer for C model 114  
Importation Log Messages 102  
importing  
    Generated Code 104  
    Limitations and Restrictions 104  
    log messages when migrating 102  
    models 113  
    Rational Rose generated code 104  
Importing license keys 77  
input focus (active window) policy 144  
install  
    types 28  
install program  
    run 55  
Install Rational Rose RealTime on UNIX 55

install types  
    administrative install 28  
    client install 28  
    client install from Network 28  
installation  
    preparing for (Windows) 28  
    procedure 25  
    restarting (UNIX) 52  
    stopping (UNIX) 52  
Installation Guide Updates 2  
installation instructions  
    Unix 54  
installing  
    compiler environment setup 50  
    floating license key for Unix 92  
    instructions 27  
    license key 95  
    license keys 89, 92  
    mixed versions 24  
    Multiple OS Versions 52  
    permanent license keys 89  
    permanent license on Unix 93  
    permanent license on Windows 89  
    Rational Rose RealTime on Windows 28  
    startup license on Unix 92  
    startup license on Windows 89  
    testing your environment 49  
    upgrade information 28  
Installing license keys  
    Before You Begin 89  
installing on windows  
    types 28  
installing startup license keys 89  
integration xvii  
    Microsoft Development Environment 138  
    naming directories 139  
    Rational Purify 138  
    Rational RequisitePro 137  
    Rational Robot 138  
    Rational SoDA for Word 137  
integration notes 135  
Integration With Rational Suites Licensing 95

## J

- Java
  - sending message specification data field format for 160
- Java language support 9

## K

- key file for licenses 82
- keys
  - emergency 75

## L

- Language Add-in Changes 110
- Layout tools 100
- Library browser 106
- license
  - floating key for Unix 92
  - Node-Locked 74
- license activation process 83
- license agreements 30, 36
- license daemon 79
  - start 80
- license file 79
  - format 85
- license file format 85
- license files 85
- license key file 78, 82
- license keys 53
  - importing 77
  - installing 89, 92
  - receiving 77
  - requesting 76
  - returning 75
  - validity 28
- License Manager 79
  - commands 80
- license manager
  - verify 80
- license manager daemon 81
- license\_check 81
- license\_setup 78

## licenses

- Emergency 75
- Evaluation 75
- Floating 74
- key file 82
- License Manager 80
- node-locked 74
- on Unix 84
- Permanent 75
- Temporary 75
- types 74
- upgrading 76
- Windows 79

## licensing

- integration with Rational Suites 95
- troubleshooting 96

## licensing on Unix 84

- administration commands 85

## Licensing Options (UNIX) 57

## limit stacksize 157

## limitations

- RQA-RT 161

## line styles 100

## List headers 107

## lmdiag 81, 85

## lmdown 81, 85

## lmgrd 79, 81, 84

- running from command prompt 84

## lmhostid 81, 85

## lmread 81

## lmremove 85

## lmreread 85

## lmstat 81, 85

## Log messages 158

## log messages 102

## M

## Macros 130

## Mainsoft 20

## MainWin 20

## makefile override changes 133

## Makefile overrides changes 133

## manual mode 8

- message 158
- Message Reply 123
- messages
  - creating 160
- migrating
  - building 101
  - C language migration 112
  - C++ language migration 115
  - code generation 101
  - Compilation 107
  - converting a C++ model to C 112
  - customized CM scripts 109
  - file format changes 108
  - from ObjecTime Developer 136
  - from ObjecTime Developer 5.2 and 5.2.1 105
  - from Rational Rose 99, 136
  - from Rational Rose and ObjecTime Developer 136
  - from Rose RealTime 6.0, 6.0.1, 6.0.2, 6.1 108
  - importation log messages 102
  - importing Rational Rose generated code 104
  - limitations and restrictions 103
  - new modelling language elements 101
  - ObjecTime Developer for C migration 113
  - opening models from Rational Rose 102
  - RRTEI changes 110
  - running 101
  - source control migration 108
  - terminology changes 105
  - user interface differences 99
- Migrating customized CM scripts 109
- Migrating from Rational Rose 99
- migration 119
  - language add-in changes 110
  - RRTEI Changes 110
  - running two different releases of Rose
    - RealTime 110
  - workspace files 110
- Minimum Run Time 162
- mixed versions 24
- Model browser 106
- model browsers 100
- Model Integrator 149
  - Suggested Workflow 149
- Model Integrator add-in 149

- Model Properties 133
- model timing services 9
- modelling language elements 101
- models
  - converting Connexis models 67
  - opening from Rational Rose 102
- mounting the CD-ROM 55
- MSCs
  - converting 159
- multiple model browsers 100

## N

- Naming Directories 139
- node-locked license 74
- Node-Locked Licenses 74
- noEmulateREI 145
- nologo 145
- ninstall 92

## O

- ObjecTime Developer for C migration 113
- opening
  - Rational Rose model 102
- opening models from Rational Rose 102
- order policy for windows 144
- Ordering Information 10
- Output windows 100

## P

- Parameters available in transition code 132
- passivation 106
- Permanent License 75
- permanent license key
  - receiving 78
- permanent licenses 75
- platforms (see referenced configurations) 13, 16
- Point to a FLEXlm Server 82
- Port cardinality 133
- Port Indexes 125
- ports 20, 120

- PrinterPorts 21
- printing
  - adding a printer on UNIX 20
- Project files 106
- property editors 133
- protocols 120
- PSCRIPT driver 20
- Purge 124
- Purify 138
- Purify on UNIX 138

## Q

- q 142
- quiet 142

## R

- rational
  - vendor daemon 79
- Rational Connexis
  - overview 7
- Rational Customer Service
  - contacting xviii
- Rational Quality Architect
  - description 8
  - overview 8
- Rational Setup Wizard
  - procedure 25
- Rational SoDA for Word 137
- Rational Web site 10
- rational\_dir 52
- read
  - license file 81
- Recall 124
- Receiving License Keys 77
- recreate\_registry 141, 142
- referenced configuration requirements
  - Windows 2000 14
  - Windows NT 13
  - Windows UNIX 15
  - Windows XP Pro 14
- referenced configurations 13, 14
- referenced configurations and targets 17

- referenced host configurations 16
- regedit 142
- register 145
- regserver 145
- replication values 118
- repository setup for ClearCase 48, 60
- Request a Copy of a License File 76
- Requesting License Keys 76
- requirements 15
  - referenced configurations 13, 14, 15
  - Toolchain 15
- Requirements Management Tools
  - Integration 137
- requirements management tools integration 137
- RequisitePro 137
- Restarting an Installation 52
- Returning License Keys 75
- RGB values 148
- Robot 138
- ROOM 105
- ROOM\_InSignal 114
- ROOM\_PortSend 115
- ROOM\_Signal 114
- Rose C++ Analyzer add-in 150
- Rose RealTime for Unix 142
- RoseRT -recreate\_registry 141
- ROSERT\_HOME 11
- RQA-RT
  - allow data qualifier in data field of send message 9
  - allow drivers to model timing services 9
  - allow sub-capsule instances to be drivers 8
  - Capsule interface generation 9
  - Java language support 9
  - Verification mode changes to allow more control in manual mode 8
- RQA-RT limitations 161
- RRTEI changes 110
- rs\_hostinfo 77
- rs\_install 55
  - license\_check 81
  - license\_setup 78
- RTDDCSRunning 156
- RTDDCSrunningReply 156
- RTPortRef operations 128



- RTSignalNames 130
- RTTimespec Parameters 130
- run
  - install program 55
- runScriptAndQuit 146, 158
- Run-Time connectivity viewing 7
- RunVerifyBehavior 158

## S

- send
  - asynchronous 122
  - synchronous 123
- SEND\_SCALAR 117
- sending
  - message specification data field format for
    - Java 160
  - messages to the Log 158
- Sequence Diagram
  - customizing from a Trace 160
- Sequence diagrams
  - creating 160
- Setup Script 58
- signals 120
- silent installation
  - overview of procedure 43
  - performing 44
- SoDA for Word 137
- softlink 54
- source control
  - command line access to ClearCase 48, 59
- Source Control Migration 108
- start
  - license daemon 80
  - new vendor daemon 81
- start script
  - single server 80
- starting
  - command line options 145
  - Rational Rose RealTime (UNIX) 66
- starting Rational Rose RealTime on UNIX 141
- starting Rational Rose RealTime on
  - Windows 141

- starting Rose RealTime
  - Unix 141
  - Unix startup options 142
  - Windows 141
- Start-up keys 87
- Start-up options for UNIX 142
- startuplicense 142
- Static Forwarding Pattern 127
- status
  - feature usage 81
  - license daemons 81
- stopping an Installation 52
- sub-capsule instances 8
- synchronous sends 123

## T

- Target Deployment Package 155
- Target Observability Behavior 157
- targets 17
- Technical Support
  - contacting 9
- Temporary License 75
  - converting to permanent 78
- Terminology mappings (from ROOM to UML) 105
- test your environment 49
- testing 8
- Timing service 115
- timing services 9
- TLA 88
- To Port 158
- Toolchain requirements 15
  - compiler 16
  - Help Viewer 15
  - real-time Operating System 16
- Toolchan requirements 15
- trace
  - customizing a Sequence diagram from 160
- transition code parameters 132
- troubleshooting
  - licensing 96
  - Rational Connexis 156
  - Rational Quality Architect 157

- Rational Rose RealTime Professional Edition 155
- Signals no longer supported 156
- Transport Integration Framework 156
- Turning off auditing for a single transport is not recommended 156
- type safety 120
- Type safety explained 120

## U

- ulimit -s 157
- UML 105
- unattended installations. *See* silent installations
- Uninstalling 153
  - UNIX 153
  - Windows 153
- uninstalling
  - Rational Rose RealTime on Unix 153
  - Rational Rose RealTime on Windows 153
- unit testing tools integration 137
- UNIX
  - adding printer on 20
  - after you install 58
  - before you Install 51
  - configuration requirements 15
  - installation instructions 54
  - installation Overview 54
  - set Connexis Variable 61
  - Setup Script 58
  - softlink 54
  - starting Rational Rose RealTime 66
  - unmount CD-ROM 58
  - upgrade Information 52
- UNIX and the X Window System 142
- UNIX server 97
- unmount
  - CD-ROM 58
- unmount CD-ROM 58
- unregister 146
- unregserver 146
- unresolved references 102
- unsafe sends 118
- Updating Batch Files 46

- upgrade information (Windows) 28
- Upgrading Licenses 76
- User Interface Differences 99, 107

## V

- v 142
- vcvars32.bat 50
- vendor daemon 79, 81
  - licenses 81
- verbose 142
- verify
  - license manager operation 80
- Verify Behavior 157
- verifying
  - Component Compatibility with Connexis Version 70
  - Connexis Installation 61
  - host platform installation 61
  - installation 61

## W

- Web Publisher 147
  - Limitations 148
  - Suggested Workflow 147
- web site
  - Rational 10
- window order policy 144
- window order policy (X window system) 144
- Windows
  - after you install 46
  - before you install 27
  - licenses 79
  - Toolchain requirements 15
- windows 20
- Windows 2000 14
- Windows NT
  - configuration requirements 13
- Windows XP Pro
  - configuration requirements 14
- with other Rational products xvii
- Workspace browser 106
- workspace files 110

## **X**

X clients 143

X servers 143

X window managers 143

X Window system 142, 143, 144