

Installation Guide

RATIONAL ROSE® REALTIME

VERSION: 2002.05.20

PART NUMBER: 800-025111-000

WINDOWS/UNIX

IMPORTANT NOTICE

COPYRIGHT

Copyright ©1993-2001, Rational Software Corporation. All rights reserved.

Part Number: 800-025111-000

Version Number: 2002.05.20

PERMITTED USAGE

THIS DOCUMENT CONTAINS PROPRIETARY INFORMATION WHICH IS THE PROPERTY OF RATIONAL SOFTWARE CORPORATION (“RATIONAL”) AND IS FURNISHED FOR THE SOLE PURPOSE OF THE OPERATION AND THE MAINTENANCE OF PRODUCTS OF RATIONAL. NO PART OF THIS PUBLICATION IS TO BE USED FOR ANY OTHER PURPOSE, AND IS NOT TO BE REPRODUCED, COPIED, ADAPTED, DISCLOSED, DISTRIBUTED, TRANSMITTED, STORED IN A RETRIEVAL SYSTEM OR TRANSLATED INTO ANY HUMAN OR COMPUTER LANGUAGE, IN ANY FORM, BY ANY MEANS, IN WHOLE OR IN PART, WITHOUT THE PRIOR EXPRESS WRITTEN CONSENT OF RATIONAL.

TRADEMARKS

Rational, Rational Software Corporation, Rational the e-development company, ClearCase, ClearCase Attache, ClearCase MultiSite, ClearDDTS, ClearQuest, ClearQuest MultiSite, DDTS, Object Testing, Object-Oriented Recording, ObjecTime & Design, Objectory, PerformanceStudio, ProjectConsole, PureCoverage, PureDDTS, PureLink, Purify, Purify'd, Quantify, Rational, Rational Apex, Rational CRC, Rational Rose, Rational Suite, Rational Summit, Rational Visual Test, Requisite, RequisitePro, RUP, SiteCheck, SoDA, TestFactory, TestFoundation, TestMate, The Rational Watch, AnalystStudio, ClearGuide, ClearTrack, Connexis, e-Development Accelerators, ObjecTime, Rational Dashboard, Rational PerformanceArchitect, Rational Process Workbench, Rational Suite AnalystStudio, Rational Suite ContentStudio, Rational Suite Enterprise, Rational Suite ManagerStudio, Rational Unified Process, SiteLoad, TestStudio, VADS, among others, are either 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.

Microsoft, the Microsoft logo, Active Accessibility, Active Channel, Active Client, Active Desktop, Active Directory, ActiveMovie, Active Platform, ActiveStore, ActiveSync, ActiveX, Ask Maxwell, Authenticode, AutoSum, BackOffice, the BackOffice logo, BizTalk, Bookshelf, Chromeffects, Clearlead, ClearType, CodeView, Computing Central, DataTips, Developer Studio, Direct3D, DirectAnimation, DirectDraw, DirectInput, DirectMusic, DirectPlay, DirectShow, DirectSound, DirectX, DirectXJ, DoubleSpace, DriveSpace, FoxPro, FrontPage, Funstone, IntelliEye, the

IntelliEye logo, IntelliMirror, IntelliSense, J/Direct, JScript, LineShare, Liquid Motion, 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, Natural, NetMeeting, NetShow, the Office logo, One Thumb, OpenType, Outlook, PhotoDraw, PivotChart, PivotTable, PowerPoint, QuickAssembler, QuickShelf, Realmation, RelayOne, Rushmore, SourceSafe, TipWizard, TrueImage, TutorAssist, V-Chat, VideoFlash, Virtual Basic, the Virtual Basic logo, 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 trademarks or registered trademarks of Microsoft Corporation in the United States and other countries.

FLEXIm and GLOBEtrotter are trademarks or registered trademarks of GLOBEtrotter Software, Inc. 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.

Portions Copyright ©1992-2001, Summit Software Company. All rights reserved.

PATENT

U.S. Patent Nos. 5,193,180 and 5,335,344 and 5,535,329 and 5,835,701. Additional patents pending.

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

GOVERNMENT RIGHTS LEGEND

Use, duplication, or disclosure by the U.S. Government is subject to restrictions set forth in the applicable Rational Software Corporation license agreement and as provided in DFARS 277.7202-1(a) and 277.7202-3(a) (1995), DFARS 252.227-7013(c)(1)(ii) (Oct. 1988), FAR 12.212(a) (1995), FAR 52.227-19, or FAR 227-14, as applicable.

WARRANTY DISCLAIMER

This document and its associated software may be used as stated in the underlying license agreement. 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 or fitness for a particular purpose or arising from a course of dealing, usage, or trade practice.

Contents

Introduction	1
Welcome to Rational Rose RealTime	1
Release Notes	1
Installation Guide Updates	2
Overview of Rational Rose RealTime Capabilities	2
What's New?	3
New Features to Enhance Model Navigation and Simplify User Workflow	3
RQA-RT Enhancements	4
Build and Target Enhancements	4
Documentation Improvements	5
How to Get Help	5
Contacting Rational Customer Support Through the Help Menu	5
Contacting Rational Customer Support by Email or Telephone	5
License Support Contact Information	7
Evaluation and Ordering Information	8
Rational Web Site	8
Directory Contents	8
Accessing the Online Help System	10
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	15
Real-time Operating System	16

Referenced Host Configurations	16
Creating Executables for Hosts without Toolset Support	19
Generating an Executable Without a Common File System	19
Adding a Printer on UNIX.	20
Installing Rational Rose RealTime on Windows	23
Before You Install	23
CCient Installation Tasks.	24
Administering Licenses.	24
Preparing for a Rational Rose RealTime Installation	25
Upgrade Information	25
Administrative Installation Tasks	26
Using the Rational Software Setup Program	26
Performing a Client Installation	26
Performing an Administrative Installation	30
Performing a Client Installation from the Network	32
After You Install	33
Updating Batch Files	33
Configuring Your Environment	33
Installing Professional Edition Software	34
Testing your Environment.	34
Installing Rational Rose RealTime on UNIX.	37
Before You Install	37
Installing in Secure Environments	38
Installing Multiple OS Versions of Rational Suite DevelopmentStudio RealTime (UNIX)	38
Stopping and Restarting an Installation	38
Upgrade Information	38
Upgrading to New Version Only (Uninstalling Earlier Version)	39
Upgrading to 6.4 While Maintaining an Earlier Version	40
Installation Instructions	40

After You Install	44
Source to Setup Script	45
Set Connexis Variable	45
Unmount the CD-ROM Drive	45
Install the Professional Edition Software	45
Starting Rational Rose RealTime (UNIX).	45
Understanding Rational Rose RealTime Licenses	47
How Licenses Work	47
Types of Licenses.	48
Node-Locked Licenses.	48
Floating Licenses	48
Permanent Licenses and Temporary License Keys	49
Emergency and Evaluation Keys	49
Suite Licenses and Point Product Licenses	49
Returning License Keys	49
Upgrading Licenses	50
Requesting License Keys.	50
Receiving and Importing License Keys	51
Requesting License Keys by Fax	51
Receiving Permanent License Keys.	52
Converting a Temporary License to a Permanent License	52
Licenses for Windows	53
The License Manager - UNIX.	53
License Manager Commands	54
Additional Licensing Commands	55
License Manager Daemon (lmgrd)	55
Vendor Daemon.	55
License Key File.	56
Application Program.	56
Configuring a UNIX Workstation to Point to a FIEXIm Server	56
License Activation Process	57
Licensing on UNIX	57
Running the lmgrd from a Command Prompt	57
Example.	58
Administration Commands.	58

The License File	59
Format	59
UNIX Licenses	60
Start-up or Emergency keys	61
Node-Locked keys	61
Floating keys	61
TLA	61
Frequently Asked Questions	62
Installing License Keys	63
Before You Begin	63
Installing a Startup or Permanent License on Windows	63
Installing a Permanent License on Windows	65
Installing the License Key	66
Installing a Floating License Key on a UNIX server	67
Installing a Startup or Permanent License on UNIX	67
Installing a Startup License on UNIX	67
Installing a Permanent License on UNIX	68
Installing the License Key	70
Integration With Rational Suites Licensing	70
Troubleshooting	71
Windows	71
UNIX server	72
UNIX	72
Migration	75
Migrating from Rational Rose	75
User Interface Differences	75
New Modeling Language Elements	77
Code Generation, Building, and Running	77
Opening Models from Rational Rose	78
List of Importation Log Messages	78
Limitations and Restrictions	79
Importing Rational Rose Generated Code	80
Limitations and Restrictions	80

Migrating from ObjecTime Developer 5.2/5.2.1	81
Terminology	81
User Interface Differences	83
Compilation	83
Migrating from Rational Rose RealTime 6.0/6.0.1/6.0.2/6.1	84
File Format Changes	84
Source Control Migration	84
Migrating customized CM scripts	85
Language Add-in Changes	86
Running Two Different Releases of Rational Rose RealTime	86
Workspace Files	86
RRTEI Changes	86
C Language Migration	88
Converting a C++ Model to C	88
ObjecTime Developer for C Migration	89
Importing models	90
Converting global signals to local signals	90
Timing service	91
C++ Language Migration	91
Backwards Compatibility Mode	91
Migrating in two steps	92
What does backwards compatibility do?	92
Compiler will find all errors	92
Building a Model in Backwards Compatibility Mode	93
Full migration	95
Changes	95
C++ UML Services Library	95
Code generation and compilation	96
New classes for protocols, signals, and ports	96
Type safety explained	96
How has this been changed?	97
API Changes Summary	97
Asynchronous Sends	98
Synchronous Sends	99
Message Reply	99
Defer, Recall, and Purge	100
Port Indexes	101
Discriminating in Code the Signal of a Received Message	102

Forwarding	102
RTPortRef Operations	104
RTTimespec Pameters	106
RTSignalNames	106
Macros	106
External Layer Service (ELS)	107
Code Generation	107
Components	107
Directory structure	108
Parameters available in transition code	108
Port cardinality cannot be unspecified	109
Makefile override changes	109
Model Properties	109
Advanced property editors	109
Integration Notes	111
Overview	111
Configuration Management (CM) Tools Integration	111
ClearCase on a UNIX Server and Clients on both NT and UNIX	112
Migrating from Rational Rose and ObjecTime Developer	112
Requirements Management Tools Integration	113
Rational SoDA for Word	113
Rational RequisitePro	113
Unit Testing Tools Integration	114
Rational Purify	114
Adding options to Purify on UNIX	114
Microsoft Development Environment	115
Integration with Rational Robot	115
Naming Directories	115
Starting Rational Rose RealTime	117
Starting Rational Rose RealTime on Windows	117
Starting Rational Rose RealTime on UNIX	117
Start-up options for UNIX	118

Rational Rose RealTime for UNIX and the X Window System	118
X clients	119
X servers	119
X window managers.	119
Input focus (active window) policy	120
Window order policy.	120
Automating Rational Rose RealTime	120
Command Line Options	121
Add-Ins	123
Web Publisher	123
Suggested Workflow	123
Limitations	124
Model Integrator	125
Suggested Workflow	125
Rose C++ Analyzer	126
Suggested Workflow	126
Limitations	128
Uninstalling Rational Rose RealTime	129
Windows.	129
UNIX.	129
Index.	131

Contents

This chapter is organized as follows:

- *Welcome to Rational Rose RealTime* on page 1
- *Overview of Rational Rose RealTime Capabilities* on page 2
- *What's New?* on page 3
- *How to Get Help* on page 5
- *Directory Contents* on page 8
- *Accessing the Online Help System* on page 10

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 **Documentation** link.

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

What's New?

Welcome to Rational Rose RealTime Version 2002.05.02.305.000. Based on extensive customer consultation and feedback, this service release contains a wide range of 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:

- *New Features to Enhance Model Navigation and Simplify User Workflow* on page 3
- *RQA-RT Enhancements* on page 4
- *Build and Target Enhancements* on page 4
- *Documentation Improvements* on page 5

New Features to Enhance Model Navigation and Simplify User Workflow

- Find/Show references: On Specification dialogs and in the **Model View** tab in the browser, you can now search the model and code for references to the selected capsule, class, attribute, operation, protocol, or signal. The results appear in the **Find** window.
- Resizable show usage window.
- Moving model elements feature to simply moving capsules, protocols, classes, and packages.
- Smart resizing of Specification dialog list columns.
- C, C++ and Java language-specific Attribute, Operation and Aggregation tools to simplify UML adoption. Fine-grained computation and control over class dependencies.
- Simplified Java modeling with import Java tool, called Add External Java, that imports class, operation and attribute descriptors into UML model directly from class and jar files.
- New **Diagrams** tab on Specification dialogs for faster navigation to, and manipulation of, model element diagrams.

- Faster model navigation with referenced class hyperlinks on Specification dialogs supporting operation return type, argument type, classes, operations, port protocol class, capsule role capsule class, message operation, message signal, and more.
- New shortcut menus to navigate directly to Class specifications from operation, attribute, argument, and role types.
- Full operation signature display on Specification dialogs and Code window.
- Additional protection with optional prompt for: closing Specification dialogs; deleting model elements; quitting the toolset.
- Inheritance browser now supports rearrangement of inheritance hierarchies without diagrams.
- Specification History keeps track of the last Specification dialogs you visited. Cycle through the specification sheets with hot keys. Open selected, close selected, Specification dialogs. Lock frequently visited Specification dialogs in the history.
- Diagram placement improvements and new larger minimum size for diagram graphics.
- Better display of inherited operations.
- More keyboard shortcuts and improved visibility of shortcuts
- Optional line wrapping on internal editor.
- New menu items, **Help > Email Technical Support > Problem Report**, **Help > Email Technical Support > Feature Request**, and **Help > Email Technical Support > Support Request**, to simplify communication and feedback between users and the Rational Team.

RQA-RT Enhancements

- Performance improvements. Suppress display of Sequence diagrams and Trace during test execution.
- Improved test results overview with date and time stamp and label generated test harnesses, test results and purloined test run pass and fail diagrams.

Build and Target Enhancements

- Automatically save the output on the **Build Log** tab in the **Output** window to a file.
- Filtering build errors and sorting information by column.
- Show/Hide build warnings.

- The TargetRTS Wizard simplifies customization and adaptation to operating systems and compilers.
- New target reference configurations for ITRON 3 and targets without operating systems (NoRTOS).
- Improved C target support - target observability for single-threaded targets.
- Microsoft Visual Studio 7.0 source debugger integration.
- Reuse target build environment when building models using target setup script, `setup.pl`, and `vssetup.pl` to get the Microsoft Visual Studio environment directly from the registry.

Documentation Improvements

- New extensibility API to extend existing build, model, configuration management APIs through to target control and source debugger integration.
- New example models and animated demonstrations to accelerate learning.
- Comprehensive keyboard shortcut overview for quick reference.
- A more comprehensive index including troubleshooting topics in the online help.

How to Get Help

This section describes procedures for interacting with Rational Customer Support services.

Contacting Rational Customer Support Through the Help Menu

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

For details on how to use this feature, see the Technical Support chapter of the *Rational Rose RealTime Release Notes*.

Contacting Rational Customer Support by Email or Telephone

When contacting Rational Customer Support 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, or HP-UX 10.20)
- Computer make and model
- Your case 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

If your organization has a designated, on-site support person, please try to contact that person before contacting Rational Customer Support.

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 detailed description of your problem.

When sending email concerning a previously-reported problem, please include in the subject field: "[SR# XXXXXXXXXX]", where XXXXXXXXXX is your Service Request number. For example:

[SR# 111222333] Rational Rose RealTime installation issues

Occasionally, Rational Customer Support 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 Support" and add your fax number to the information requested above.

Telephone and fax numbers for Rational Customer Support are contained in the following table. If you have problems or questions regarding licensing, please see *License Support Contact Information* on page 7.

Table 1 Support Telephone and Fax

Region	Telephone Number	Fax Number
Americas	800-433-5444	408-863-4300
Asia Pacific (includes support for Japan, China, India, Korea, Taiwan)	+61-2-9419-0111	+61 2 9419 0123
Europe, Middle East, and Africa (includes support for Israel)	+31 (0)20 4546 200	+ 31 (0) 20 4546 202
Other worldwide locations	408-863-5000	

Email addresses for Rational Customer Support are as follows:

Table 2 Support Email

Region	Email Address
Americas and other worldwide locations	support@rational.com
Asia Pacific (includes support for Japan, China, India, Korea, Taiwan)	support@apac.rational.com
Europe, Middle East, Africa (includes support for Israel), and Scandinavia	support@europe.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 3 License Support Telephone and Fax

Region	Telephone Number	Fax Number
Americas	800-433-5444	781-676-2510
Europe, Israel, and Africa	+31 (0)20 4546 200	+ 31 (0) 20 4546 202
North Asia Pacific (Mainland China, Hong Kong, Taiwan)	+852 2143 6382	+852 2143 6018
Korea	+82 2 556 9420	+82 2 556 9426
South Asia Pacific Australia, New Zealand, Malaysia, Singapore, Indonesia, Thailand, The Philippines, Vietnam, Guam and India	+612 9419 0111	+612 9419 0123
Japan	+81 3 5423 3611	+81 3 5423 3622

Email addresses for license support are as follows:

Table 4 License Support Email

Region	Email Address
Americas	lic_americas@rational.com
Europe, Israel, and Africa	lic_europe@rational.com
North Asia Pacific Mainland China, Hong Kong, Taiwan, and Korea	lic_apac@rational.com
South Asia Pacific Australia, New Zealand, Malaysia, Singapore, Indonesia, Thailand, The Philippines, Vietnam, Guam and India	lic_apac@rational.com
Japan	lic_japan@rational.com

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 \$ROSE_HOME on UNIX (HP-UX and Solaris) and %ROSE_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 \$ROSE_HOME/Help directory must be Read/Write for all.

The 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:

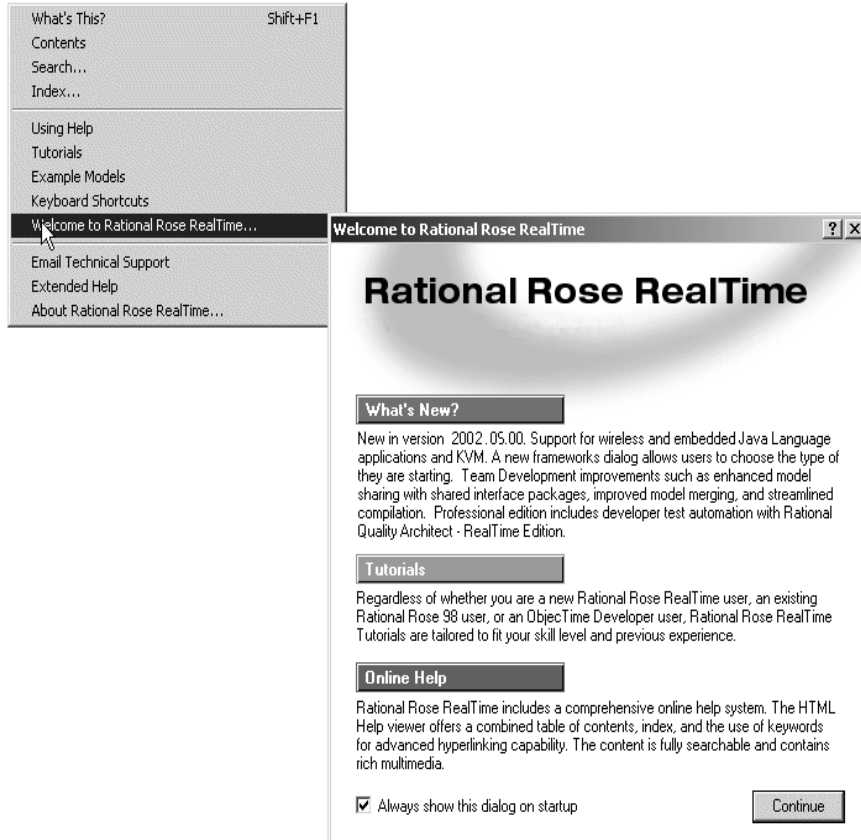
Directory	Description
ROSERT_HOME	This is the top level directory.
AddIns	Contains the configuration information required by Rational Rose RealTime Add-ins.
bin	Contains the Rational Rose RealTime executable and various scripts. The bin directory also contains subdirectories for each of the supported 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> .
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 tutorial additional information.
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 Microsoft HTML Help format. You can load the online Help Viewer from the Rational Rose RealTime toolset.

Figure 1 Help menu and Welcome screen



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.

If you choose not to have Internet Explorer as the default browser, you will need to run **Hhupd.exe** (in **redist**). This file is the distribution executable that installs the run-time components needed for an HTML Help Project, such as **Hh.exe**, **Hhctrl.ocx**, **ltss.dll**, and **ltircl.dll**. **Hhupd.exe** is in the **Redist** folder of the HTML Help Workshop folder.

Most PDF versions of the guides are available in the **\$ROSERT_HOME/Help** directory, unless specified otherwise.

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 or Windows 98.

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
- 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 5.01 or 5.5 or Netscape Navigator 4.7 or 6.0. We recommend Internet Explorer 5.01 or 5.5

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 1
- 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 5.01 or 5.5 or Netscape Navigator 4.7 or 6.0. We recommend Internet Explorer 5.01 or 5.5

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:

- 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 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 5.01 or 5.5 or Netscape Navigator 4.7 or 6.0. We recommend Internet Explorer 5.01 or 5.5

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 HP-UX 10.20
 - For Solaris operation, the minimum workstation is an UltraSparc 10. We recommend an UltraSparc 60 with 512 MB of RAM. We recommend the Solaris 2.8 operating system.
 - For HP-UX operation, we support installation of the HP 700 series architecture
 - 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.
- 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 disk space for the Rational Rose RealTime installation.

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 3.02 or later) be configured on your computer. For details, see *Accessing the Online Help System* on page 10.

Compiler

You must have a C++ compiler installed on your system to make use of the code generation 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.

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 *Referenced Host Configurations* on page 16. 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 5 shows the referenced host configurations for this release of Rational Rose RealTime.

Table 5 Host configurations

Toolset Host	Requirements
HPUX 10.20	See <i>Requirements for UNIX</i> on page 15
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
Windows NT 4.0 (Service Pack SP6a)	See <i>Requirements for Windows NT</i> on page 13
Windows 2000 (Service Packs SP1 and SP2)	See <i>Requirements for Windows 2000</i> on page 14
Windows XP Pro	See <i>Requirements for Windows XP Pro</i> on page 14

Note: Java generation on HPUX is not supported.

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. 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
- Compiler version
- Processor type

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 6 shows the referenced configurations and targets.

Table 6 Referenced configurations and targets

Host Configuration(s)	Target RTOS	Compiler/Processor	RTS Library	Connexis DCS Library
Solaris	Same	Gnu 2.95.1, SPARC Gnu 2.8.1, SPARC Gnu 2.7.2.3, SPARC Sun C++ 5.0, SPARC Sun C 5.0, SPARC	C++ C & C++ C++ C++ C	C++ C++ - C++ -
HPUX	Same	Gnu 2.8.1, HPPA HP C++ 10.11, HPPA	C & C++ C++	C++ -
Windows	Same	Visual C++ 6.0, x86 Visual C++ 7.0, x86	C & C++ C & C++	C++ -
Solaris Windows	pSOS 2.5	Diab 4.2b, ppc	C++	C++
Solaris, HPUX	VRTX 4.AB	Microtec 1.3C, ppc	C++	n/a
Windows	VRTX 4.Baa	Microtec 1.4, ppc	C++	-
Solaris, Windows	OSE 4.1.1	Diab 4.3f, ppc GreenHills 1.8.9, ppc GreenHills 2.0, ppc	C & C++ C C	C++ - -
Solaris	OSE 4.1.1 SoftKernel	Gnu 2.95.1, SPARC	C & C++	-

Table 6 Referenced configurations and targets

Host Configuration(s)	Target RTOS	Compiler/Processor	RTS Library	Connexis DCS Library
Windows	OSE 4.1.1 SoftKernel	Visual C++ 6.0, x86	C	-
Solaris, Windows, HPUX	Tornado 2.0 (VxWorks 5.4)	Cygnus 2.7.2.960126, M68040 Cygnus 2.7.2.960126 Cygnus 2.7.2.960126, ppc GreenHills 1.8.9, x86c GreenHills 2.0, ppc	C & C++ C & C++ C & C++ C & C++ C & C++	- C++ - C++ C++
Solaris	Tornado 2.0 Sim	Cygnus 2.7.2.960126, SPARC	C++	C++
Windows NT	Tornado 2.0 Sim	egcs 2.90.29, x86	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 Cygnus 2.7.97r1, ppc	C++ C++	C++ C++
Solaris	Chorus Classix 4.0	egcs-2.91.66, ppc	C++	-
Windows	Windows CE sh3	eMbedded Visual C++ 3.0, sh3	C++	C++
Windows	eCos ITRON	gnu 2.95.3, x86	C	-
N/C - native compilation only	AIX 4.2.1	gnu 2.8.1	C++	-
N/C - native compilation only	Nucleus 1.1	Diab 4.2b	C++	-
N/C - native compilation only	Red Hat Linux 6.1	Egcs 2.91.66	C++	C++
N/C - native compilation only	QNX 4.2.2	Watcom C++ 10.6	C++	-
N/C - native compilation only	UnixWare 7.0.1	SDK 3.0	C++	C++

Creating Executables for Hosts without Toolset Support

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

Note: The following steps assume that you use a common file system hierarchy 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 computer that you use to compile does not have a common file system with the generated 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 build 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 generate an executable.

To generate an executable without a common file system:

- 1 On the target host, a visible copy of the TargetRTS must be available.
- 2 Copy the component directory into the target host 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 host 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 line-up.
- 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 the following directory:

```
$ROSERT_HOME/bin/mw
```

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:

- *Before You Install* on page 23
- *After You Install* on page 33
- *Using the Rational Software Setup Program* on page 26
- *Performing a Client Installation* on page 26
- *Performing an Administrative Installation* on page 30
- *Performing a Client Installation from the Network* on page 32
- *After You Install* on page 33
- *Testing your Environment* on page 34

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.

To perform a network installation, you must have the Rational Suite for Windows installed. For instructions on how to install Rational Suite, see the *Installing Rational Suite* guide.

There are three types of installations that you can perform:

- Client Installation (a local client installation) - see *Performing a Client Installation* on page 26
- Administrative Installation - see *Performing an Administrative Installation* on page 30
- Client Installation from the Network - see *Performing a Client Installation from the Network* on page 32

Table 6 Administrative Installation Tasks

To	See
License Rational software	<i>Administering Licenses</i> on page 24
Ensure that the servers meet the minimum or recommended system and software requirements	<ul style="list-style-type: none"> ▪ <i>Requirements for Windows NT</i> on page 13 ▪ <i>Requirements for Windows 2000</i> on page 14 ▪ <i>Requirements for Windows XP Pro</i> on page 14
Upgrade from earlier versions of Rational software	<i>After You Install</i> on page 33

Client Installation Tasks

This topic outlines the general tasks you must perform before you install Rational Suite on a client computer (a local installation). Table 6 directs you to information in this manual that can help you perform client installation tasks.

Table 7 Client Installation Tasks

To	See
License your Rational software	<i>Administering Licenses</i> on page 24
Ensure that your system meets the minimum or recommended system and software requirements	<ul style="list-style-type: none"> ▪ <i>Requirements for Windows NT</i> on page 13 ▪ <i>Requirements for Windows 2000</i> on page 14 ▪ <i>Requirements for Windows XP Pro</i> on page 14
Prepare for installing Rational Rose, RealTime	▪ <i>Preparing for a Rational Rose RealTime Installation</i> on page 25
Install optional software (the Rational Rose RealTime Companion Products CD)	<i>Installing Professional Edition Software</i> on page 34
Upgrade from earlier versions of Rational software	<i>After You Install</i> on page 33

Administering Licenses

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. In the Rational Software Setup program, a green check mark next to a Rational product indicates you have a license key configured for that product. If you

do not see a green check mark next to the product you want to install, you may want to install a license key before you install the product. To configure a license key, click the **Configure Licenses** button to launch the Rational License Key Administrator (LKAD) and the License Key Administrator Wizard. Both tools provide online Help. If you do not install the license keys before you install the product, the LKAD will appear at the end of the installation.

The *Rational Suite License Management Guide* describes the web-based license manager Rational AccountLink, Rational licensing terms, and the Rational License Key Administrator. The Setup program installs the LKAD automatically with each point product within Rational Suite. The *License Management Guide* also provides instructions for requesting, installing, upgrading, and configuring floating and node-locked license keys.

Note: If you plan to upgrade the Rational license server, and this server has other Rational products installed on its system, remove the Rational products from the system or upgrade the Rational products to the current release. Older Rational products may not work with an upgraded license server if they are on the same system.

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 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 34.
- To deploy your model in 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 129 for your specific platform.

Models created in earlier versions of Rational Rose RealTime can be loaded directly into version 6.4. Rational Rose and ObjecTime Developer models should be converted as described in *Migrating from ObjecTime Developer 5.2/5.2.1* on page 81.

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 6.4 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 50.

If you upgrade to Rational Rose RealTime 6.4 from Rational Rose RealTime releases 6.1, 6.1.1, 6.2, or 6.3, your license keys are valid.

For more information on license keys, see *Installing License Keys* on page 63.

Administrative Installation Tasks

Table 6 directs you to information that can help you perform administrator installation tasks.

Using the Rational Software Setup Program

From the Rational Software Setup program, you can install selected Rational Software products, such as Rational Rose RealTime, or the Rational Suite. Your Rational product shipment includes two *Rational Solutions for Windows* installation CD-ROMs. Depending on the Rational products you select during the installation process, you may have to insert more than one installation disc.

Performing a Client Installation

The following general requirements are necessary to run the Rational Software Setup program on your system.

- You need to install Rational licenses to use Rational products. You can install the Rational license keys before or after you install Rational software.
- Ensure that other programs are not running before you start the installation.
- Turn off all virus protection software. These programs often run in the background and interfere with the installation and file decompression process.
- Ensure that your system meets the minimum requirements.

- Ensure that you have administrator privileges before installing Rational products.
- To use the Rational Software Setup program on a Windows NT, 2000, or XP Pro system, you must have Windows administrator privileges on the local system. 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
- The Rational Software Setup program uses C:\Program Files\Rational as the default installation path. The program installs software system files in the Windows directory or Windows systems directory.
- The Setup program installs Microsoft Core Components and some additional files on the same drive as the operating system (often the C:\ drive), even if you specified an alternate drive during the installation.

To perform a Client Installation of Rational Rose RealTime:

- 1 Insert the *Rational Solutions for Windows Disc 1* into your system's CD-ROM drive. The **Rational Software Setup** program starts automatically.
If autorun is disabled on your system, click **Start > Run** and type the following:
drive: \Setup.exe
where *drive* is the letter of the CD-ROM drive.
- 2 The **Choose Product** screen appears. Choose **Rational Rose RealTime** as the product to install.
Note: If you are planning on using floating licenses, the FLEXlm license manager must be installed before you install. If you do not have FLEXlm installed, select **Install FlexLm** and follow the instructions and prompts.
- 3 Click **Next**.
Note: If you have not configured a license for Rational Rose RealTime, a dialog box appears. To install Rational Rose RealTime without licensing configured, click **OK**. To configure the licensing, click **Cancel** and then click **Configure Licenses**. You can configure licenses after installing Rational Rose RealTime. For information on installing your license keys, see *Installing License Keys* on page 63.
The Rational License Agreement appears.

- 4 Click **Yes** to accept the terms and conditions of the license agreement,

Note: You must accept the license agreement to proceed. If you do not agree with the terms of the license agreement, the installation should be aborted. All software and documentation should be returned to Rational Software.

- 5 Choose either **Typical**, **Custom**, or **Compact**.

Table 8 Installation Types

Type	Description
Typical	<ul style="list-style-type: none">■ Installs the most commonly used features for a product.■ Use this option for standard installations.
Custom	<ul style="list-style-type: none">■ Allows you to add or remove product features for installation.■ Defaults to all features in a Typical installation.
Compact	<ul style="list-style-type: none">■ Installs a subset of the standard configuration. May omit optional files, including online documentation or online Help. To find out which files will be installed, read the product's release notes.■ Use this option for installations on systems with limited disk space.

Note: The **Network** configuration option is not offered with the Rational Rose RealTime point product installation.

- 6 Click **Next**.

An Information Summary dialog shows the shared components.

Update Shared Components

The **Update Shared Components** dialog appears if the Setup program needs to update shared files or components on your system. Click **Next** to have the Setup program install these files for you or **Cancel** to install these files yourself.

Note: The Setup wizard does not recalculate the disk space required for your updated selections.

Upgrade Compatibility

The **Upgrade Compatibility** dialog appears if you have older Rational products installed on your system. For each of the older products listed, we strongly recommend that you do one of the following:

- Upgrade it: Complete this installation, and then re-start the Setup program to upgrade the listed products.
- Remove it: Complete this installation, and then remove the listed products from the system.

7 Click **Next**.

A confirmation dialog appears, listing your selected settings and options.

8 Click **Next**.

The installation begins.

Note: If any errors occur during installation, such as insufficient disk space or inadequate permissions, an error summary dialog is displayed.

9 When the installation is finished, the **Setup Complete** dialog box appears, prompting you to restart your computer. We strongly recommend you click **Yes, I want to restart my computer now**.

10 Click **Restart**.

11 The installation dialog appears, indicating that your computer restarted.

12 Click **Finish**.

If you have not configured your product license, the **License Key Administrator** appears.

13 Install the License Key, if required. For information on how to install your license key, see *Installing License Keys* on page 63.

Now, you will want to review the topic, *After You Install* on page 33 for additional post-installation activities.

Performing an Administrative Installation

The following general requirements are necessary to run the Rational Software Setup program on your system.

- You need to install Rational licenses to use Rational products. Install the Rational license keys before or after you install Rational software.
- The Rational Software Setup program requires that you install all Rational products in the same directory. If you already have Rational products installed on your system, the Setup program installs additional Rational products in the same directory.
- Ensure that you have administrator privileges before installing Rational products.
- To use the Rational Software Setup program on a Windows NT, 2000, or XP Pro system, you must have Windows administrator privileges on the local system. 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
- Ensure that you have a current backup of your Registry and system directories.
- Ensure that other programs are not running before you start the installation.
- Turn off all virus protection software. These programs often run in the background and may interfere with the installation and file decompression process.
- Turn off any user interface managers or desktop environments that run on top of Microsoft Windows.
- Ensure that your system meets the minimum requirements.
- The Rational Software Setup program uses C:\Program Files\Rational as the default installation path. The program installs software system files in the Windows directory or Windows systems directory.
- 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. The Software Setup program will report the amount of space required on all drives for your installation.

- The Setup program 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.
- Initially, the Setup program requires at least 2 GB of space to perform file transfers to the server. After installation, only the required files remain on the server.

To perform an Administrative Installation of Rational Rose RealTime:

- 1 Create a shared folder on the server.

This shared folder must be visible and accessible by a client.

- 2 Map a network drive to the shared server location.

Note: Ensure that you select the **Reconnect at Logon** option in the Windows **Map Network Drive** dialog box.

You do not have to use the local server to store files. You can map a network drive to another location on another computer; however, the other computer must have a shared folder that is mapped, and it must be visible and accessible.

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

Note: The Rational Software Setup program starts automatically, but you do not want to start the Setup program this way.

- 4 Click **Cancel**.

- 5 Launch a Command Prompt (DOS window).

- 6 Change directory to the location of your CD-ROM drive.

- 7 Change to the setup directory.

- 8 To configure the server to accommodate Network installations, type the following and press ENTER:

```
rssetup -admin:<drive_letter>
```

where <drive_letter> is the name of the mapped drive created when you mapped the shared drive in step 2.

Note: Ensure that you include the colon after you specify the drive letter. Also, after you press **ENTER**, there is a time delay prior to the display of the Rational Software Setup window.

9 Click **Next** to begin.

Note: Your Rational product shipment includes several *Rational Solutions for Windows* installation CD-ROMs. You may have to insert more than one installation disc into the drive during the Setup process.

10 Click **Finish** when installation is complete.

Now that installation process copied and decompressed the product files to a shared network folder, you can perform a Client installation from a network. To perform client installation over the network, see *Performing a Client Installation from the Network* on page 32.

Note: You cannot perform upgrades with Network installations.

Performing a Client Installation from the Network

A network installation installs the required files on your system to run Rational Rose RealTime from a shared network folder. You need 240 MB disk space to accommodate the shared system DLLs and local configuration settings the Setup program installs on your system.

Use this option to run a program from a centrally managed location.

You can perform a client installation only after an Administrative Installation. An Administrative installation is performed by a system administrator who copies and decompresses product files to a shared network folder. To perform an Administrative Installation, see *Performing an Administrative Installation* on page 30.

Note: You cannot perform upgrades with network installations.

To perform a client installation from the network:

1 Log in to the Client computer as Administrator.

The domain should be the name of the local machine.

2 Open Windows Explorer.

3 From the **Tools** menu, select **Map Network Drive**.

4 Map a new drive to the shared folder found on Server.

Note: Ensure that you select the **Reconnect at Logon** option in the Windows **Map Network Drive** dialog box.

5 Using Windows Explorer, browse to this mapped drive and launch the Setup program from that location.

- 6 Click **Next**.
- 7 In the **Product** box, select **Rational Rose RealTime**.
- 8 Click **Next**.
- 9 In the **Select Configuration** window, click **Network**.
- 10 Click **Next**.

A network installation of Rational Rose RealTime is installed on Client computer.

Now, you will want to review the topic, *After You Install* on page 33 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 ROSERT_HOME variable, as well as the launch command path:

```
set ROSERT_HOME=C:\Program Files\Rational\Rose RealTime
set ROSERT_HOST=win32
set ROSERT_LICENSE_FILE=%ROSER_T_HOME%\license\license.dat
set path=%ROSER_T_HOME%\bin\%ROSER_T_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.

Environment Variables

With Rational Rose RealTime, you need to specify environment variables. Set the environment variable %ROSER_T_HOME% to the new installation directory and add %ROSER_T_HOME%/bin to your path.

Installing Professional Edition Software

If you have the Rational Rose RealTime Companion Product CD, you may want to install the Professional Edition software. To install the software on this CD, see *Installation in the Release Notes and Installation Guide - Rational Rose RealTime Professional Edition*.

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 installed on your system and configured to be run from the DOS prompt to make use of the code generation 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 37
- *Upgrade Information* on page 38
- *Installation Instructions* on page 40
- *After You Install* on page 44

Before You Install

Before you install Rational Rose RealTime on UNIX, refer to the items in Table 9 to direct you to information in this manual that can help you perform pre-installation tasks.

Table 9 **UNIX Pre-Installation Tasks**

License your Rational software	<i>Administering Licenses</i> on page 24 and <i>UNIX Licenses</i> on page 60
Ensure that your system meets the minimum or recommended system and software requirements	<i>Requirements for UNIX</i> on page 15
Install optional software (the Rational Rose RealTime Companion Products CD)	<i>Install the Professional Edition Software</i> on page 45
Upgrade from earlier versions of Rational software	<i>Upgrade Information</i> on page 38

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 Technical 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) (Solaris or HP-UX) 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 you 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 39
- *Upgrading to 6.4 While Maintaining an Earlier Version* on page 40

Upgrading to New Version Only (Uninstalling Earlier Version)

Ensure that you remove past releases of Rational Rose RealTime from your system prior to installation. Please see *Uninstalling Rational Rose RealTime* on page 129 for your specific platform.

You can load models created in earlier versions of Rational Rose RealTime directly into 6.4 (also referred to as 2002.05.00). To convert your existing Rational Rose and ObjecTime Developer models, see *Migrating from ObjecTime Developer 5.2/5.2.1* on page 81.

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 6.4 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 50.

If you upgrade to Rational Rose RealTime 6.4 from Rational Rose RealTime releases 6.1, 6.1.1, 6.2, or 6.3, your license keys are valid.

For more information on license keys, see *Installing License Keys* on page 63.

Upgrading to 6.4 While Maintaining an Earlier Version

Your Unix environment can continue to have a Rational Rose RealTime 6.4 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.xxx
```

(where .xxx represents .csh or .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.

rs_install welcome, license agreement phase

sets up install directory structures under an entered directory name
other directories created

install options and license server phase

1. Use existing server
license file or server already setup
e.g. <port>@<servername>

2. Permanent or TLA license
imports license_XXX.upd into a rational.dat file and stops & restarts the license server

3. Temporary license

installs applications
based on selection, installs application into appropriate directories

rs_install automatically creates setup scripts which should be sourced in the .cshrc or .profile or .login as appropriate

RoseRT environment variables
RoseRT depends on environment variables which are set up by sourcing the setup files created during install

Use the appropriate setup script for the shell being used and the installed setup file

e.g. for csh
source <rational_dir>/rosert_setup.csh

e.g. for ksh
.<rational_dir>/rs_setup.sh

Entered dir name < rational_dir >
e.g. /ratlserver/Rel2002/RRT.2002

/base
/config
/releases
/DevelopmentStudioUNIX.2002.05.00

1. modifies setup files when installed

2. /config/rational.dat created

3. /config/temporary.dat created

created and filled
/releases/RoseRT.2002.05.00
/releases/purify.2002.05.00
/releases/RUP_Gen.2002.05.00

rs_setup.csh, rs_setup.sh (suite install)
or
rosert_setup.csh, rosert_setup.sh
(RoseRT only) install

rs_setup.xxx
ROBERT_HOME
ROBERT_LICENSE_FILE
ROBERT_HOST
Purify, RUP and other variables env settings are also set

rosert_setup.xxx
ROBERT_HOME
ROBERT_LICENSE_FILE
ROBERT_HOST

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 HP-UX 10.20)

Note: You should be **root** to install the product.

- 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
```

▫ **HP-UX 10.20**

```
# mkdir /cdrom
# mount -r -F cdfs /dev/dsk/c0t2d0 /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	Set up permanent or counted license(s). <ul style="list-style-type: none">▪ Request Node-Locked or 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.
3	Set up a temporary license file.

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:

- *Source to Setup Script* on page 45
- *Set Connexis Variable* on page 45
- *Unmount the CD-ROM Drive* on page 45
- *Install the Professional Edition Software* on page 45

Source to 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.xxx
```

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

```
source <rational_dir>/ rosert_setup.xxx
```

where *.xxx* is *.csh* or *.sh*.

Set Connexis Variable

Set the following:

```
CONNEXIS_HOME to $ROSSERT_HOME/Connexis
```

Unmount 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.

Install the Professional Edition Software

If you have the Rational Rose RealTime Companion Product CD, you may want to install the Professional Edition software. To install the software on this CD, see *Installation* in the *Release Notes and Installation Guide - Rational Rose RealTime Professional Edition*.

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*.

Understanding Rational Rose RealTime Licenses

5

Contents

This chapter is organized as follows:

- *How Licenses Work* on page 47
- *Types of Licenses* on page 48
- *Requesting License Keys* on page 50
- *Converting a Temporary License to a Permanent License* on page 52
- *The License Manager - UNIX* on page 53
- *License Manager Commands* on page 54
- *Licensing on UNIX* on page 57
- *The License File* on page 59
- *UNIX Licenses* on page 60
- *Frequently Asked Questions* on page 62

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 48
- *Floating Licenses* on page 48
- *Permanent Licenses and Temporary License Keys* on page 49
- *Emergency and Evaluation Keys* on page 49

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 54.

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 51 or call Rational Licensing Support. See *Contacting Rational Customer Support by Email or Telephone* on page 5 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) 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 Support by Email or Telephone* on page 5 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 *How to Get Help* on page 5 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 *How to Get Help* on page 5 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 50.

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. In the Rational Software Setup program, a green check-mark next to a Rational product indicates that you have a license key configured for that product. If you do not see a green check-mark next to Rational Rose RealTime, you may want to install a license key before installing. 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 at 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 HP-UX:

```
% su
# cp <rational_dir>/config/start_lmgrd_on_server-name \
  /sbin/init.d/S98Rational
# ln -s /sbin/init.d/S98Rational /sbin/rc2.d/S98Rational
```

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)
```

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:

```
$ROBERT_HOME/bin/sun5/lmstat -c $ROBERT_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 `$ROSERT_HOME/bin/<arch>`, where `<arch>` is the host that Rational Rose RealTime is installed on (`sun5` or `hpux10`).

- **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 $ROSERT_LICENSE_FILE`. Otherwise, include the `$ROSERT_LICENSE_FILE` environment variable in a semicolon ("`;`") separated list.
- **logfile** is the path to a log file. `$ROSERT_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:

- HP-UX: `/sbin/init.d/SImRational.sh`
- Solaris: `/etc/rc2.d/SImRational.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 $ROSERT_LICENSE_FILE -l /apps/logs/logRRT
```

or

```
lmgrd -c $ROSERT_LICENSE_FILE;$LM_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 or hpux10).

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

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 63
- *Installing a Startup or Permanent License on Windows* on page 63
- *Installing a Startup or Permanent License on UNIX* on page 67
- *Integration With Rational Suites Licensing* on page 70
- *Troubleshooting* on page 71

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. See *License Support Contact Information* on page 7.

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

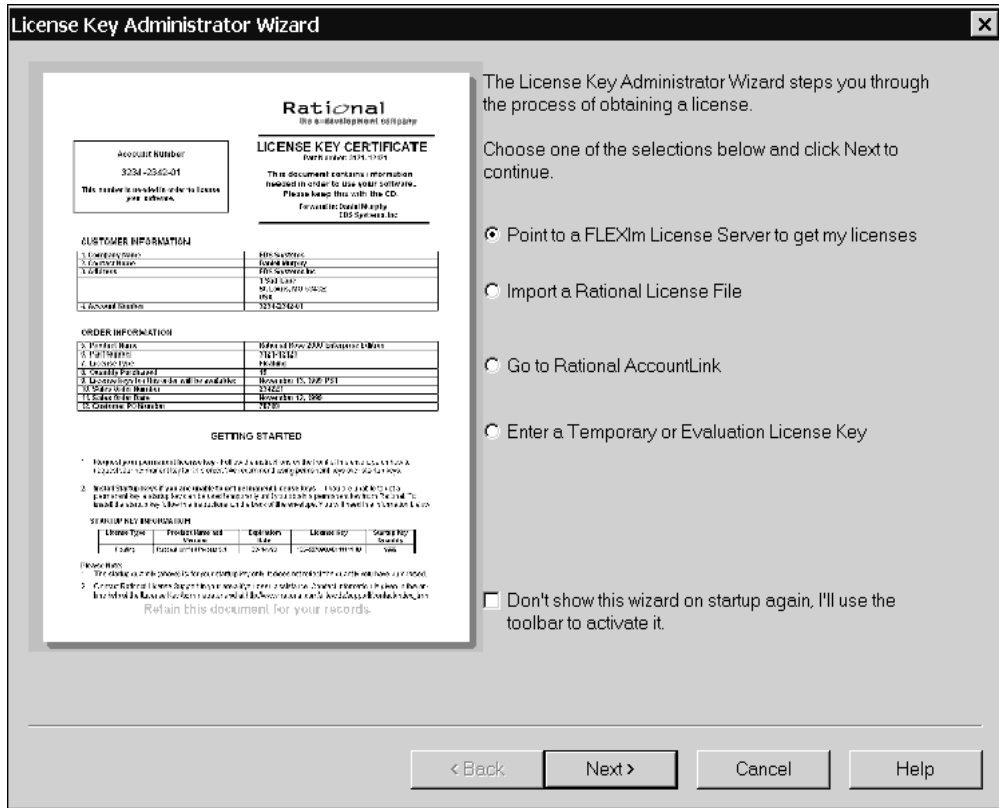
You can install Rational license keys before or after you install a Rational product. If you want to install a license key before you install a Rational product, open the Rational License Key Administrator by clicking the **Configure Licenses** button in the **Choose Product** dialog box. Use the Rational 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.

After the Rational Rose RealTime product installation is complete, the LKAD wizard appears.

Figure 2 License Key Administrator (LKAD) Wizard



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
- 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. See *License Support Contact Information* on page 7
- 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 65.
- 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 53, 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

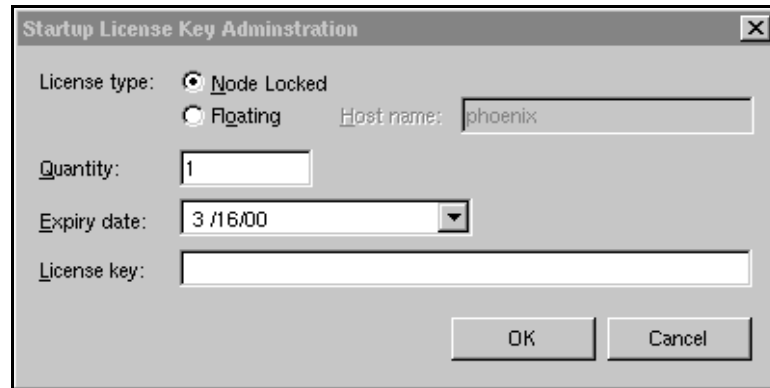
The startup license keys are time-limited and allow you to start using Rational Rose RealTime immediately.

Installing a Startup License on UNIX

To install a startup license on UNIX:

- 1 Go to the \$ROSSERT_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.
- License key:** An 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 47.

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
- 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. See *License Support Contact Information* on page 7.

- 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 70.
 - To not integrate Rational Rose RealTime with any other Rational products, see *The License Manager - UNIX* on page 53, 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 67.

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 ROSERT_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 47, 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 67.

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 47, 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 67.

Contents

This chapter is organized as follows:

- *Migrating from Rational Rose* on page 75
- *Migrating from ObjecTime Developer 5.2/5.2.1* on page 81
- *Migrating from Rational Rose RealTime 6.0/6.0.1/6.0.2/6.1* on page 84
- *C Language Migration* on page 88
- *C++ Language Migration* on page 91

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 3 Layout menu - right-click on any diagram

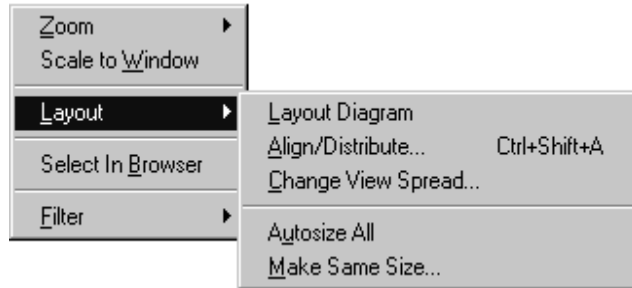
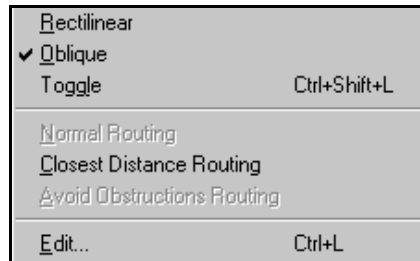


Figure 4 Line attributes menu - Edit > Line Attributes



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 and 98i (.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*.

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**.

Note: Files from Rational Rose versions newer than Rational Rose 98 and 98i must be opened in Rational Rose 98/98i and saved using the 4.5 or 6.0 format before opening in Rational Rose RealTime.

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:

- State diagrams and Activity diagrams

The model opens, but the state diagrams and activity diagrams are not present in Rational Rose RealTime.

- Importing Rose models containing controllable units is not supported.

If the Rational Rose model file contains controllable units, you should export the model from Rational Rose98 into a single .ptl petal file (**File > Export Model**), that can then be opened with Rational Rose RealTime (**File > Open**, and select **All Files...** in the combo box to display .ptl files).

- Three-tier class diagrams are not supported in Rational Rose RealTime.

If the Rational Rose model file contains a three-tier class diagram, you should create a copy of the Rational Rose model that does not contain a three-tier diagram to import into 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 78.

- 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.

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

Context/update

In ObjecTime Developer, contexts 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

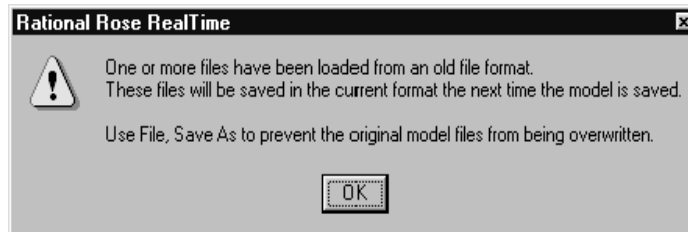
Models from these previous versions of Rational Rose RealTime are compatible with this version. However, there are some changes in team development and language add-ins that require you to plan some changes to your model.

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 5 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 Technical 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 88 and *C++ Language Migration* on page 91 for instructions on migrating existing models to either of these Language Add-Ins.

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

Running Two Different Releases of Rational Rose RealTime

Windows NT

If you need to run both versions of the tool while you are converting your models to the new release, you need to start the 6.02 release with a batchfile to reset your environment settings to the 6.02 defaults. This script is available on the Rational Rose RealTime support website in the patches and updates section:

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

Note: Add-Ins and other product integrations may not work with the 6.0.2 release after you have installed the new release on your workstation because of the new registry settings. We recommend that you remove the 6.0.2 release from your workstation as soon as your 6.0.2 model has been converted to the new release.

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.

For additional information, see *Upgrading to 6.4 While Maintaining an Earlier Version* on page 40.

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 unfortunately 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 91. Then, you can convert to the new syntax described in *Changes* on page 95.

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 number local to each protocol class defining the signal. Signals with the same name in different protocols do not share the same integer value.

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.

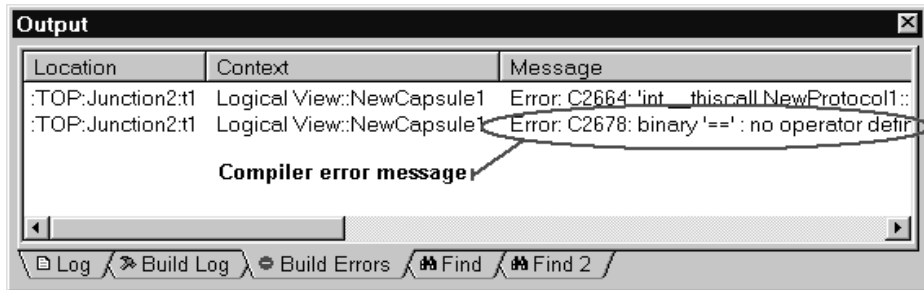
What does backwards compatibility do?

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 6 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).

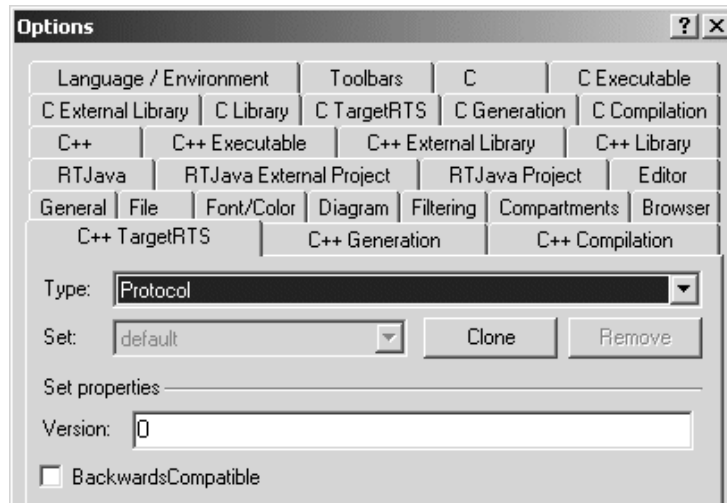
If you want to continue to use the **SEND_SCALAR** macro you should turn off the **TypeSafeSignals** property on these protocols.

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 102 and *Discriminating in Code the Signal of a Received Message* on page 102.

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 95 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 95 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 96
- *New classes for protocols, signals, and ports* on page 96
- *API Changes Summary* on page 97
- *Macros* on page 106
- *External Layer Service (ELS)* on page 107

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 Technical 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 been 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 98 (to one or all port instances)
- *Synchronous Sends* on page 99 (to one or all port instances)
- *Message Reply* on page 99
- *Defer, Recall, and Purge* on page 100 (one or all signals to one or all port instances)
- *Port Indexes* on page 101
- *Discriminating in Code the Signal of a Received Message* on page 102
- *Forwarding* on page 102 (potentially from one protocol to another and to one or all port instances)
- *RTPortRef Operations* on page 104

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 108.

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 the a generic **RTPProtocol** 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 signal numbers are no longer global (that is, a signal with the same name and data class in two protocols won't have the same signal number) 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 NewProtocol1::Base::rti_start:
        outport.start(*rtdata).send();
        break;
    case NewProtocol1::Base::rti_stop:
        outport.stop(*rtdata).send();
        break;
    default:
        log.log("Unexpected message");
}

// or you could have 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:

RTEndPort ** 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 **RTEndPort** appeared previously. For example, before you would have:

```
RTEndPort ** 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.

RTEndPort ** 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.

RTEndPort * 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 );
RTEndPort * port = (RTEndPort *)0;
if( aid.isValid() ) {
    port = replicatedportref.incarnationTo( aid );
    if( port != (RTEndPort *)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 Pameters

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 107. 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 99.

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 111
- *Configuration Management (CM) Tools Integration* on page 111
- *Requirements Management Tools Integration* on page 113
- *Unit Testing Tools Integration* on page 114
- *Microsoft Development Environment* on page 115
- *Integration with Rational Robot* on page 115
- *Naming Directories* on page 115

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
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 76.1.1.1 on HPUX
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 75.

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.

Note: The Rational Requisite Pro integration does not support the association of a Rational Rose RealTime package with a RequisitePro project. Use Case and Model association is supported.

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

Once a component is built and a component instance has been created, the instance can then be run and observed. Purify detects errors in your own code as well as the components your software uses. For information, see the Running and Debugging section in the *Rational Rose RealTime Toolset Guide*.

Adding options to Purify on UNIX

The toolset looks for an installation of Purify by checking for an environment variable named **PURE_HOME**. This environment variable is not set up by installing Rational Purify.

You must set this environment variable manually. The variable need not point to a directory containing Purify, nor is it required to point to a directory. The variable may contain anything, but must be set.

Occasionally, you may need to add options during a Purify'd build on UNIX. For example, Purify on HP needs to know the name of the linker or collector used by **Gnu g++**.

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++** on HP, 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 2002.05.00 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 2002.05.00 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 Technical 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 117
- *Starting Rational Rose RealTime on UNIX* on page 117
- *Rational Rose RealTime for UNIX and the X Window System* on page 118
- *Automating Rational Rose RealTime* on page 120
- *Command Line Options* on page 121

Starting Rational Rose RealTime on Windows

To start Rational Rose RealTime on Windows, on the **Start** menu, choose **Programs > 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 63.

For additional information on configuring your environment variables, see *Configuring Your Environment* on page 33.

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 this 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, in **.Xdefaults**, set the following environment variable:

Dtwm*secondariesOnTop: True

Setting this variable to True ensures that an opened secondary window in Rational Rose RealTime for UNIX, such as an external editing window, does not get caught behind the main primary window. Since the secondary window is the active window, you are unable to regain focus of this secondary window.

If **Dtwm*secondariesOnTop** is not set to True, since the secondary window is the active window, you may not be able to regain focus of this secondary window when it goes behind the primary window.

When using CDE as your XWindow manager, 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 on UNIX:

<filename>

A user option to load a model on startup.

-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 123
- *Model Integrator* on page 125
- *Rose C++ Analyzer* on page 126

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's Internet Explorer 4.0 or better. (www.microsoft.com)
- Netscape's Communicator 4.06 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)
- 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:

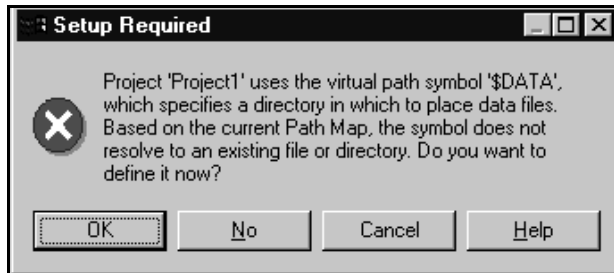
- 1 Set compiler settings.
- 2 Add Files.
- 3 Analyze.
- 4 Code Cycle.
- 5 Export to Rose.

From Rational Rose RealTime:

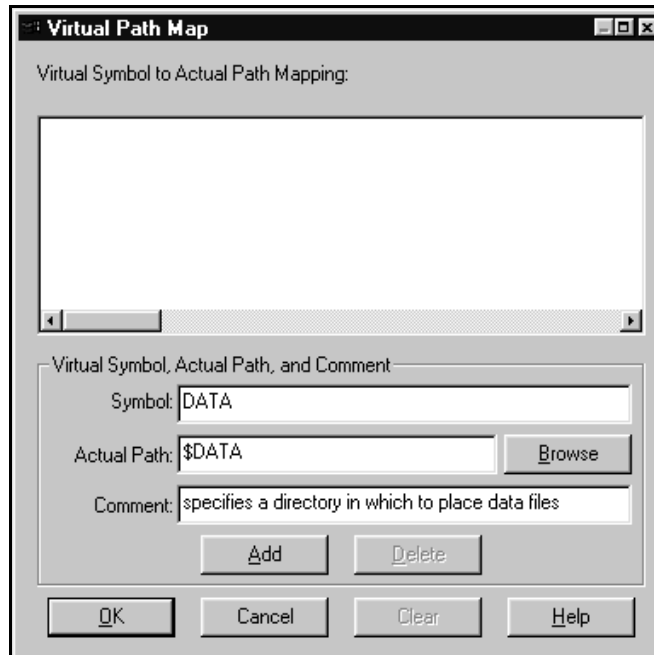
- 1 Select **File > Open** to load the Rose Model.
- 2 Select **Tools > Import Code** to import code from source files.

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.

Uninstalling Rational Rose RealTime

11

Contents

This chapter is organized as follows:

- *Windows* on page 129
- *UNIX* on page 129

Windows

To uninstall 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 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.

Index

A

- accessing
 - online help 10
- AccountLink 50
- activation process
 - licenses 57
- active window policy (X window system) 120
- adding
 - printer on UNIX 20
- add-ins
 - Model Integrator 125
 - Rose C++ Analyzer 126
- administering Licenses 24
- administering licenses 25
- administration commands
 - licensing on Unix 58
- Administrative Installation 30
- Administrative Installation Tasks 24, 26
- Administrator Installation Tasks 26
- Advanced property editors 109
- Advantages of backwards compatibility 92
- API Changes 97
- API Changes Summary 97
- asynchronous sends 98
- automating
 - Rational Rose RealTime 120
 - Rose RealTime 120
- Automating Rose RealTime 120

B

- backwards compatibility
 - advantages 92
 - disadvantages 92
- Backwards Compatibility Mode 91
- Batch Files
 - updating 33
- bin (directory) 9

- building
 - Model in Backwards Compatibility Mode 93
- Building a model in backwards compatibility mode 93

C

- C Language Migration 88
- C language migration 88
- C++ Analyzer 126
 - Limitations 128
 - Suggested Workflow 126
- C++ Language Migration 91
- C++ language migration 91
- C++ UML Services Library 95
- CD-ROM
 - mounting instructions 42
 - unmount 45
- classes 96
- cleanup 118
- ClearCase integration 85
- ClearCase on a UNIX Server 112
- client 32
- Client Installation 26, 27
- Client Installation from the Network 32
- client installation from the network 32
- Client Installation Tasks 24
- Code browser 76
- Code editors 76
- Code Generation 77, 107
- Code generation 96
- Command Line Options 121
- command line options 121
- commands
 - License Manager 54
 - license manager 54
- Compact 28
- compilation 96
- Configuration Management (CM) Tools
 - Integration 111

- configuration management tools integration 111
- configuration requirements
 - UNIX 15
 - Windows 2000 14
 - Windows NT 13
 - Windows XP Pro 14
- configurations
 - host 16
- Configure Licenses 25
- configuring
 - environment 33
- Configuring a Unix Workstation to Point to a FlexLm Server 56
- contacting Rational Technical Support 5
- convert an existing Rose RealTime model 89
- converting
 - C++ Model to C 88
 - global signals to local signals 90
- converting a C++ model to C 88
- converting temporary licenses to permanent 52
- creating
 - Executables for Hosts without Toolset Support 19
- creating executables 19
- Custom 28

D

- Defer 100
- DEMO FEATURE 60
- Devices 21
- Dynamic Forwarding 103

E

- ELS 107
- Emergency Keys 49
- Emergency keys 61
- Emergency License 49
- emulateREI 121
- Enable BackwardsCompatible protocol
 - property 93
- environment configuration 33

- environment variables
 - configuring for Windows 33
- Evaluation License 49
- exinstal 55, 58
- External Layer Service 107
- External Layer Service (ELS) 107

F

- file
 - license 53
- File Format Changes 84
- file format changes 84
- Fixing unresolved references 78
- fixing unresolved references 78
- FLEXIm
 - application program 56
- FlexLm
 - configuring for Unix 56
- FIEXlm Server 56
- Floating License 48
- floating license key for Unix 67
- Forwarding 102
- Forwarding Pattern 103
- Frequently Asked Questions 62

G

- generating
 - executable without a common file system 19
- generating executables 19
- getting help 5

H

- Hhupd.exe 11
- host configurations 16
- hosts
 - creating executables without Toolset support 19
- how to get help 5

I

- Imgrd 55
- import an ObjecTime Developer for C model 90
- Importation Log Messages 78
- importing
 - Generated Code 80
 - Limitations and Restrictions 80
 - log messages when migrating 78
 - models 90
 - Rational Rose Generated Code 80
 - Rational Rose generated code 80
- Importing license keys 51
- Input focus (active window) policy 120
- input focus (active window) policy 120
- install
 - types 23
- install program
 - run 43
- Install Rational Rose RealTime on UNIX 42
- install types
 - administrative install 23
 - client install 23
 - client install from Network 23
- installation
 - preparing for (Windows) 25
 - Rational Software Setup program 26
 - restarting (UNIX) 38
 - stopping (UNIX) 38
- Installation Guide Updates 2
- installation instructions
 - Unix 40
- Installation Types 28
- installation types 28
- installing
 - compiler environment setup 35
 - floating license key for Unix 67
 - instructions 23
 - license key 66, 70
 - license keys 63
 - Multiple OS Versions 38
 - on Unix 39
 - permanent license keys 63
 - permanent license on Unix 68
 - permanent license on Windows 63

- Rational Rose RealTime on Windows 25
- startup license on Unix 67
- startup license on Windows 63
- testing your environment 34
- upgrade information 25
- upgrade information for Unix 39

- Installing license keys
 - Before You Begin 63
- installing licenses 25
- installing on windows
 - types 23
- installing Professional Edition Software 34
- installing startup license keys 63

- integration
 - Microsoft Development Environment 115
 - naming directories 115
 - Rational Purify 114
 - Rational RequisitePro 113
 - Rational Robot 115
 - Rational SoDA for Word 113
- integration notes 111
- Integration With Rational Suites Licensing 70

K

- key file for licenses 56
- keys
 - emergency 49

L

- Language Add-in Changes 86
- Layout tools 76
- Library browser 82
- license
 - floating key for Unix 67
 - Node-Locked 48
- License Activation Process 57
- license activation process 57
- license daemon 53
 - start 54
- License File
 - format 59
- license file 53

- license file format 59
- license files 59
- license key
 - installing 66
- License Key Administrator 64
- License Key File 56
- license key file 52, 56
- License Keys 39
 - validity 26
- license keys
 - importing 51
 - installing 63
 - receiving 51
 - requesting 50
 - returning 49
- License Manager 53
 - commands 54
- license manager 53
 - verify 54
- license manager commands 54
- License Manager Daemon 55
- license manager daemon 55
- license server, upgrading 25
- license_check 55
- license_setup 52
- licenses
 - administering 24
 - Emergency 49
 - Evaluation 49
 - Floating 48
 - key file 56
 - License Manager 54
 - node-locked 48
 - on Unix 57
 - Permanent 49
 - Temporary 49
 - types 48
 - upgrading 50
 - Windows 53
- licensing
 - integration with Rational Suites 70
 - license usage order 25
 - troubleshooting 71
- licensing on Unix 57
 - administration commands 58

- Licensing Options (UNIX) 44
- line styles 76
- List headers 83
- LKAD 25, 64
- lmdiag 55, 58
- lmdown 55, 58
- lmgrd 53, 55, 57
 - running from command prompt 57
- lmhostid 55, 58
- lmread 55
- lmremove 58
- lmreread 58
- lmstat 55, 58
- log messages 78

M

- Macros 106
- Mainsoft 20
- MainWin 20
- makefile override changes 109
- Makefile overrides changes 109
- Message Reply 99
- migrating
 - building 77
 - C language migration 88
 - C++ language migration 91
 - code generation 77
 - Compilation 83
 - converting a C++ model to C 88
 - customized CM scripts 85
 - file format changes 84
 - from ObjecTime Developer 112
 - from ObjecTime Developer 5.2 and 5.2.1 81
 - from Rational Rose 75, 112
 - from Rational Rose and ObjecTime Developer 112
 - from Rose RealTime 6.0, 6.0.1, 6.0.2, 6.1 84
 - importation log messages 78
 - importing Rational Rose generated code 80
 - Limitations and Restrictions 79
 - limitations and restrictions 79
 - new modelling language elements 77
 - ObjecTime Developer for C migration 89

- opening models from Rational Rose 78
- RRTEI changes 86
- running 77
- source control migration 84
- terminology changes 81
- user interface differences 75
- Migrating customized CM scripts 85
- Migrating from Rational Rose 75
- migration 95
 - language Add-in Changes 86
 - language add-in changes 86
 - RRTEI Changes 86
 - running two different releases of Rose
 - RealTime 86
 - workspace files 86
- Model browser 82
- model browsers 76
- Model Integrator 125
 - Suggested Workflow 125
- Model Integrator add-in 125
- Model Properties 109
- Modeling Language Elements 77
- modelling language elements 77
- models
 - opening from Rational Rose 78
- mounting the CD-ROM 42
- multiple model browsers 76

N

- Naming Directories 115
- Network configuration option 28
- node-locked license 48
- Node-Locked Licenses 48
- noEmulateREI 121
- nologo 121

O

- ObjecTime Developer for C Migration 89
- ObjecTime Developer for C migration 89
- opening
 - Rational Rose model 78
- opening models from Rational Rose 78

- order policy for windows 120
- Ordering Information 8
- Output windows 76

P

- Parameters available in transition code 108
- passivation 82
- perform a Client Installation 27
- perform an Administrative Installation 31
- Permanent License 49
- permanent license key
 - receiving 52
- Permanent Licenses 49
- platforms (see referenced configurations) 13, 16
- Point to a FLEXlm Server 56
- Port cardinality 109
- port cardinality 109
- Port Indexes 101
- ports 21, 96
- PrinterPorts 21
- printers
 - adding on Unix 20
- printing
 - adding a printer on UNIX 20
- Project files 82
- property editors 109
- protocols 96
- PSCRIPT driver 20
- Purge 100
- Purify 114
- Purify on UNIX 114

Q

- q 118
- quiet 118

R

- rational
 - vendor daemon 53
- Rational SoDA for Word 113

- Rational Software Setup program 26
- Rational Web site 8
- rational_dir 38
- read
 - license file 55
- Recall 100
- Receiving License Keys 51
- recreate_registry 117, 118
- 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 118
- register 121
- regserver 121
- replication values 94
- Request a Copy of a License File 50
- Requesting License Keys 50
- requesting licenses 25
- requirements 15
 - referenced configuration 13
 - referenced configurations 14, 15
 - Toolchain 15
- Requirements Management Tools
 - Integration 113
- requirements Management tools integration 113
- RequisitePro 113
- Restarting an Installation 38
- Returning License Keys 49
- RGB values 124
- Robot 115
- ROOM 81
- ROOM_InSignal 90
- ROOM_PortSend 91
- ROOM_Signal 90
- Rose C++ Analyzer add-in 126
- Rose RealTime for Unix 118
- RoseRT -recreate_registry 117
- ROSERT_HOME 9
- RRTEI Changes 86
- RRTEI changes 86

- rs_hostinfo 51
- rs_install 43
 - license_check 55
 - license_setup 52
- RTPortRef operations 104
- RTSignalNames 106
- RTTimespec Parameters 106
- run
 - install program 43
- runScriptAndQuit 121

S

- send
 - synchronous 99
- SEND_SCALAR 93
- sends
 - asynchronous 98
- Setup Program 26
- Setup Script 45
- signals 96
- SoDA for Word 113
- softlink 40
- Source Control Migration 84
- start
 - license daemon 54
 - new vendor daemon 55
- start script
 - single server 54
- starting
 - command line options 121
 - Rational Rose RealTime (UNIX) 45
- starting Rational Rose RealTime on UNIX 117
- starting Rational Rose RealTime on
 - Windows 117
- starting Rose RealTime
 - Unix 117
 - Unix startup options 118
 - Windows 117
- Start-up keys 61
- Start-up options for UNIX 118
- startuplicense 118
- Static Forwarding Pattern 103

- status
 - feature usage 55
 - license daemons 55
- stopping an Installation 38
- synchronous sends 99

T

- targets 17
- Technical Support
 - contacting 5
- Temporary License 49
 - converting to permanent 52
- Terminology mappings (from ROOM to UML) 81
- test your environment 34
- Timing service 91
- TLA 61
- To 67
- Toolchain 15
- Toolchain requirements 15
 - Compiler 15
 - compiler 15
 - Help Viewer 15
 - RealTime Operating System 16
 - real-time Operating System 16
- Toolchan requirements 15
- transition code parameters 108
- troubleshooting
 - licensing 71
- type safety 96
- Type safety explained 96
- Typical 28

U

- UML 81
- Uninstalling 129
 - UNIX 129
 - Windows 129
- uninstalling
 - Rational Rose RealTime on Unix 129
 - Rational Rose RealTime on Windows 129

- Unit Testing Tools Integration 114
- unit testing tools integration 114
- UNIX
 - adding printer on 20
 - after you install 44
 - before you Install 37
 - configuration requirements 15
 - install the Professional Edition Software 45
 - installation instructions 40
 - installation Overview 40
 - set Connexis Variable 45
 - Setup Script 45
 - softlink 40
 - starting Rational Rose RealTime 45
 - unmount CD-ROM 45
 - upgrade Information 38

- Unix
 - adding a printer 20
- UNIX and the X Window System 118
- UNIX server 72
- unmount
 - CD-ROM 45
- unmount CD-ROM 45
- unregister 121
- unregserver 121
- unresolved references 78
- unsafe sends 94
- Updating Batch Files 33
- upgrade information (Windows) 25
- upgrading
 - license server 25
- Upgrading Licenses 50
- User Interface Differences 75, 83
- using the Setup Program 26

V

- v 118
- vcvars32.bat 35
- Vendor Daemon 55
- Vendor daemon
 - licenses 55
- vendor daemon 53, 55

- verbose 118
- verify
 - license manager operation 54

W

- Web Publisher 123
 - Limitations 124
 - Suggested Workflow 123
- web site
 - Rational 8
- window order policy 120
- window order policy (X window system) 120
- Windows
 - after you install 33
 - before you install 23
 - licenses 53
 - Toolchain requirements 15
- windows 20
- Windows 2000 14
- Windows NT
 - configuration requirements 13
- Windows XP Pro
 - configuration requirements 14
- Workspace browser 82
- Workspace Files 86
- workspace files 86

X

- X clients 119
- X servers 119
- X window managers 119
- X Window system 118, 119, 120