Rational the e-development company™

UNIX/LINUX

Release Notes

Rational Rose

VERSION: 2001A.04.00

PART NUMBER: 800-024947-000

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

COPYRIGHT NOTICE

Code known as pdksh is included in this software in object code form and in an unmodified state for the installation of Rose 2001 for Linux x86. Information concerning pdksh, including legal disclosures can be found at http://www.cs.mun.ca/~michael/pdksh. The following applies only to the pdksh code and no other portions of this software:

pdksh is provided AS IS, with NO WARRANTY, either expressed or implied. The vast majority of the code that makes pdksh is in the public domain except that it contains sigact.c and sigact.h which are covered by copyrights by Simon J. Gerraty; the copyright notice for these files is as follows: This is free software. It comes with NO WARRANTY. Permission to use, modify and distribute this source code is granted subject to the following conditions. 1/ that the above copyright notice and this notice are preserved in all copies and that due credit be given to the author. 2/ that any changes to this code are clearly commented as such so that the author does get blamed for bugs other than his own. aclocal.m4: This is covered by the GNU General Public Licence (GPL) as it contains modified versions of macros that come with GNU autoconf. As this is used solely for configuration, the pdksh code itself is not covered by the GPL.

The following is taken from autoconf 2.x documentation (info autoconf questions distributing) concerning use of autoconf in programs: There are no restrictions on how the configuration scripts that Autoconf produces may be distributed or used. In Autoconf version 1, they were covered by the GNU General Public License. We still encourage software authors to distribute their work under terms like those of the GPL, but doing so is not required to use Autoconf.

U.S. GOVERNMENT RIGHTS NOTICE

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

TRADEMARK NOTICE

Rational, the Rational logo, Purify, Visual PureCoverage, Visual SourceSafe, Visual Quantify, ClearQuest, Rational Visual Test, and Rational DevelopmentDeskTop are trademarks or registered trademarks of Rational Software Corporation in the United States and in other countries.

Visual C++, Visual Basic, Windows NT, Developer Studio, and Microsoft are trademarks or registered trademarks of the Microsoft Corporation. All other names are used for identification purposes only and are trademarks or registered trademarks of their respective companies.

Rose 2001 for Tru64 Unix, includes the Compaq C++ Run-Time Library Redistribution kit (from Compaq Computer Corporation) which provides the necessary 6.2 C++ Run-Time library (libxcc). Copyright 2000 Compaq Computer Corporation.

U.S. Registered Patent Nos. 5,193,180 and 5,335,344 and 5,535,329. Licensed under Sun Microsystems Inc.’s U.S. Pat. No. 5,404,499. Other U.S. and foreign patents pending.

Printed in the U.S.A.

Rational Rose for UNIX/Linux Release Note Contents

Preface

Rational Rose for UNIX/Linux® is a comprehensive, integrated programming environment that supports the development of complex software systems. These release notes contain information about last-minute changes to Rose and the Rose Add-Ins.

Audience

This manual is intended for:

- Anyone who installs Rose
- Anyone who uses Rose

Other Resources

- Online Help is available for Rational Rose.
- All manuals are available online, either in HTML or PDF format. The online manuals are on the Rational Solutions for Windows Online Documentation CD.
- For more information on training opportunities, see the Rational University Web site at http://www.rational.com/university.

Contacting Rational Technical Publications

To send feedback about documentation for Rational products, please send e-mail to techpubs@rational.com.

Contacting Rational Technical Support

Contact Rational Customer Support if you have questions about installing, using, or maintaining this Rational Software product. The following table describes how to contact Rational Customer Support.

Your Location	**Telephone**	**Fascimile**	**E-mail**
North America	(800) 433-5444 (toll free) (408) 863-4000 Cupertino, CA	(781) 676-2460 Lexington, MA	support@rational.com
Europe, Middle East, Africa	+31 (0) 20-4546-200 Netherlands	+31 (0) 20-4545-201	support@europe.rational.com
Asia Pacific	+61-2-9419-0111 Australia	+61-2-9419-0123 Australia	support@apac.rational.com

For up-to-date contact numbers and addresses visit the support contacts web page at http://www.rational.com/support/contact.

Note: When you contact Rational Technical Support, please be prepared to supply the following information:

- Your name, telephone number, and company name
- Your computer's make and model
- Your operating system and version number
- Product release number and serial number
- Your case ID number (if you are following up on a previously-reported problem)

Contacting Rational Licensing Technical Support

If you have questions about acquiring license keys for your Rational Software products, contact Rational Licensing Support as follows:

Your Location	Telephone	Facsimile	E-mail
Asia Pacific	+61-2-9419-0111	+61-2-9419-0123	license@apac.rational.com
Asia Pacific (Japan)	+61-2-9419-0111	+61-2-9419-0123	license@japan.rational.com
Asia Pacific (Korea)	+82-2-556-9420	+82-2-556-9426	license@apac.rational.com
Asia Pacific (Mainland China, Hong Kong, and Taiwan)	+61-2-9419-0111	+61-2-9419-0123	license@china.rational.com
Europe Middle East Africa	+31 20 4546 200	+31 20 4546 202	license@europe.rational.com
North American (East Coast) South America	800-433-5444 (toll free)	781-676-2460	license@rational.com
North America (West Coast) Central America	800-433-5444 (toll free)	408-863-4001	license@rational.com

Introduction

1

Thank you for selecting Rational Rose for UNIX/Linux. The release notes provide last-minute changes and reference information about Rational Rose and all of the Rational Rose Add-Ins. The release notes contain a listing of enhancements, known problems, and other information pertaining to the operation and functionality of Rational Rose.

Updates for Registered Owners

From time to time, service packs and additional utilities may be made available for download to registered owners. Please periodically check the Rational Rose web page at www.rational.com/rose/.

Rational Rose for UNIX/Linux Documentation Set

Rose for UNIX/Linux comes with the following printed documents:

- *Installation Guide* - A guide to installing the Rose software and setting up licenses.
- *Release Notes* - Updated technical information on Rose.

Note: Printed versions of the above two documents are not provided in Beta releases of Rose software.

For a complete printed documentation set, contact your Rational Sales office.

The complete documentation set is supplied in electronic format with this release.

Rose for UNIX/Linux Beta Help Books may include information describing functionality which will not be completely available until this product becomes Generally Available. If you should have questions in relation to the information presented in the Help Book, contact Rational Technical Support.

Installation and Licensing

2

Please refer to the *Rose for UNIX/Linux Installation Guide* that is provided as your primary guide for installing and setting up licensing for this product.

If you encounter any problems while installing or licensing the Rose product, please check here first to see if it is a known problem. If a problem is not yet documented, please contact Rational Technical Support so we can investigate it, provide you with a workaround, and track the problem for future action.

This chapter contains the following sections:

General Install Information

As a precaution, always make back ups of any Rose model files that you will be accessing with this software.

This product includes FLEXlm, a licensing product from Globetrotter, Inc. A startup license key that allows you to start using your product immediately is shipped with this product. You must obtain a separate permanent license from Rational in order to continue using this product beyond this startup key expiration date. Please refer to the licensing section of the *Installation Guide* for detailed instructions on obtaining and installing permanent licenses. Also, see *Licensing Information* on page 19 for any last minute licensing notes.

Note: Rose 98i, 2000e, and 2001 for UNIX licenses are valid for this release. You will not need to utilize the startup licenses or obtain new permanent licenses if you are already utilizing Rose 98i, 2000e, or 2001 for UNIX licenses.

Pre-Installation Information

You may want to back up your Registry prior to running the installation program. Specifically, we recommend you make backups of the following files:

- You should always have a backup of all your model files. These can include files that end with .mdl,.cat,.ptl, .red, and .sub.
- Custom property files, which can include files that end with .pty and .prp

  Note: There is no need to save .pty files that ship with Rose since these will be re-installed.
- Custom script files, which can include files that end with .ebs and .ebx

  Note: There is no need to save .ebs or .ebx files that ship with Rose, since these will be re-installed.
- Path Maps (For Rose 98i, and 2000e path map variables are saved in the registry.)

Follow these steps to back up path maps:

1. Run **regedit** and navigate to **HKEY_CURRENT_USER\Software\Rational Software\Rose\Virtual Path Map**
2. Click **Registry > Export Registry File...**

Follow these steps to restore path maps:

1. Run **regedit** and navigate to **HKEY_CURRENT_USER\Software\Rational Software\Rose\Virtual Path Map**
2. Click **Registry > Import Registry File...** and import the file you exported.

System Requirements

Before installing this software, please verify that your system meets the minimum system requirements listed below. If the requirements are not met, the product may not install and/or operate correctly.

Note: All platforms may not be available at this time. Contact Rational technical support for platform availability information.

The following table contains the hardware requirements for installation and use of this product. Note that the recommended amount of memory is only the suggested minimum amount. Additional memory may improve performance. Your memory requirements will also be larger if you are constructing large models.

Table 1 Hardware Requirements

Hardware Item	Recommendation	For
Memory	64 + (32 * *N*) MB of RAM (where *N* is the number of users running Rose simultaneously on a given workstation)	Use
Server	A midrange UNIX/Linux server or client UNIX/Linux workstation (Sun servers: Sparc Station 20 and above)	Use
Client	An AIX, Solaris, HP-UX, SGI IRIX, or Tru64 Unix workstation running the required operating system level as described in the following table under operating system software requirements. The UNIX/Linux workstation must be capable of displaying X Windows or a PC running eXceed version 6.1or higher to display X windows or for Rose on Linux x86, a PC running with Xfree and Xi Graphics X servers on Linux. It is recommended to use more than 256 colors in your graphics card / monitor configuration.	Use
Disk space	270 MB for loading release + 1-3 MB for each Rose model	Installation, Use
CD-ROM	Mounted as a UNIX/Linux file system	Installation
Display	Color display	Use

The next table contains the software requirements for installation and use of this product. Note that this is also only a suggested amount of swap space. Your requirements may be larger if you expect to construct very large models. If all of your models are small, you may need less than the amount shown.

Table 2 Software Requirements

Software Item	Requirement	For
Operating system[a]	AIX 4.3.2 Solaris 2.5.1, 2.6, 7, 8 (kernel revision January 2000 and later) HP-UX[b] 10.20 (ACE release[c]), 11.00, and 11.11* SGI IRIX 6.5.5 Tru64 Unix 4.0f Linux x86 Red Hat 6.2 and 7.0	Use
C++ Compiler #include files	AIX CSet ++ 3.6.6 Solaris SC 5.0 HP-UX 10.20: HP aC++ B3910B A.01.21 HP aC++ B3910B A.01.19.02 Language Support Library HP-UX 11.00: HP aC++B3910B A.03.13 HP aC++B3910B X.03.11.10 Language Support Library IRIX C++ 7.3.1.1m Tru64 for Unix C++ 6.2 Linux x86 g++ 2.95.2	Analyzer
Java	Rose J supports the following Java IDE's. Use the JDK appropriate to your IDE.[d] - IBM VisualAge for Java Professional Edition (Linux only) - IBM VisualAge for Java Enterprise Edition (Linux only) - Forte for Java Internet Edition 2.0 (Solaris Linux only) - Forte for Java Community Edition 2.0 (Solaris Linux only) - JBuilder 4.0 Professional Edition (Solaris Linux only) - JBuilder 4.0 Foundation Edition (Solaris Linux only)	

Table 2 Software Requirements

Software Item	Requirement	For
JDK, JFC, and J2EE	JDK 1.1.6 JFC 1.1 (swing classes for 1.1) JDK 1.2 (based on 1.2.1) J2SE 1.2 (based on 1.2.2) J2SE 1.3 J2EE 1.2 or any of the major Java IDEs*	
Port map daemon	Must be running	Installation, Use
TCP/IP	Must be running	Installation, Use
Host names	Must be configured	Installation, Use
Page/Swap space	150 + (50 * N) MB	Use
X Window System	Native X11R5 or X11R6 (except HP 10.20) X news on Sun with OpenWindows 3 PC X server eXceed version 6.1 or higher In addition, when running Rose for Linux x86, XFree and Xi Graphics X servers on Linux	Use

Table 2 Software Requirements

Software Item	Requirement	For
Window Manager	MWM (Motif Window Manager) version 1.2 vuewm on HP-UX 4dwm on SGI IRIX dtwm (CDE (Desk Top) version 1.0) olwm (Open Look Window Manager) on SUN is supported, but icons of Rose for UNIX/Linux may not integrate well with other icons on the screen In addition to those listed above, Rose for Linux x86 will run with the GNOME desktop environment with the Enlightenment window manager and the KDE desktop environment with the lcwm window manager. The following Window Managers are not supported: olvwm (olwm with virtual screens) Rose for UNIX/Linux will not remain on original virtual desktop, but rather will follow when other desktops selected. twm (shipped with X11) tvtwm (twm with virtual screens) fwm gwm fvwm	Use

a. All platforms may not be available at this time. Contact technical support for availability information.
b. HP9000 Series 700
c. Rose requires that an ACE release be installed to run properly on HP-UX 10.20. To determine if an ACE release is installed, run **/usr/sbin/swlist | grep ACE**.
d. See the Rose_IDE_Link_README file in the release for Rose UNIX/Linux IDE Link Integration setup information, prior to using the IDE Link integration.

Hardware/Software Patch Requirements

Please see “Operating System Runtime and/or Patch Requirements” in the *Installation Guide* for system patch requirements.

When installing and running Rose for UNIX/Linux for the first time, you will be alerted if hardware or software patches are necessary. If running on a Solaris or HP-UX server, Rose for UNIX/Linux will attempt to automatically determine if patches are necessary on your server and/or Solaris display, and will advise accordingly.

You may also run the following anytime to determine if your computer requires operating system patches to run Rose for UNIX/Linux.

- SUN Solaris Server (computer running Rose for UNIX/Linux)

  cd *rational_dir***/releases/rose.***I.J.K***/bin**

  ./check_rose_reqs sol2_server rose

- SUN Solaris Display (computer displaying Rose for UNIX/Linux)

  Check that the DISPLAY environment variable is set to your display, and then:

  cd *rational_dir***/releases/rose.***I.J.K***/bin**

  ./check_rose_reqs sol2_client rose

- HP-UX, IBM, or SGI Server

  cd *rational_dir***/releases/rose.***I.J.K***/bin**

  for HP-UX **- ./check_rose_reqs hp_runtime rose**

  for IBM **- ./check_rose_reqs aix_runtime rose**

  for SGI **- ./check_rose_reqs sgi_runtime rose**

  for Tru64 Unix **- ./check_rose_reqs axp_runtime rose**

Configuring Window Manager So Pop-Up Windows Will Remain On Top of their Parent Windows

Configure your Window Manager such that pop-up windows will remain on top of their parent windows, in compliance with the Windows API. Under X11, Window Managers may implement different strategies for maintaining the Z-order of pop-up windows.

In order to keep pop-ups on top, if possible:

- For MWM - Motif Window Manager

There is no specific configuration to keep pop-up (transient) windows on top. Pop-up windows are kept on top by default.

- For DTWM - Desk Top Window Manager (shipped with CDE)

  Depending on your system’s configuration, the default behavior of dtwm may not keep pop-up (transient) windows on top of their owner.

  For example, it may be that the window stacking behavior for dialogs is specific, such as “Dtwm*secondariesOnTop = False”. In such a case, here are some possible solutions for keeping the pop-ups on top:

a Change Dtwm*secondariesOnTop to True in the system-wide dtwm configuration file. On Solaris, the dtwm configuration file is located at **/usr/dt/app-defaults/C/Dtwm**. On platforms other than Solaris, the path to the dtwm configuration file may be different.

b Add

```
Dtwm*secondariesOnTop: True
```

to your **.Xdefaults** file.

c Be more specific with the program name, and add, for example:

```
Dtwm*ssexp*secondariesOnTop: True
```

to your **.Xdefaults** file.

Note: The **.Xdefaults** file needs to be reloaded with:

```
xrdb ~/.Xdefaults
```

- For OLWM -- Open Look Window Manager

Add the following line to your **.Xdefaults** file:

```
OpenWindows.KeepTransientsAbove: True
```

Running Multi-threaded GUI Applications on Multi-processor Solaris 2.6 Computers

If Rose for UNIX/Linux hangs when running on a multi-processor Solaris 2.6 computer, verify that the Solaris 2.6 server and display (if applicable) includes the Sun Solaris XSun Patch 105633-32 or higher. This patch includes a fix for Sun Defect 4233280, Sun SO#3918581 (multi-threaded applications crash/hang on 2.6 multiprocessor computers). To check for this patch, run

```
showrev -p | grep 105633
```

on both the server running Rose and the display (if display Solaris 2.6).

When installing or running the application for the first time, Rose will advise if this patch is installed or required.

Displaying on Ultrabook, Using PGX Graphics Card

A CDE crash may occur when displaying Rose for UNIX/Linux on a Ultrabook using a PGX Graphics Card, if SUN Solaris patch 105362-26 is installed.

If using a PGX Graphics Card, we recommend that SUN Solaris patch 105362-20 be installed as this is the latest version tested and determined to be compatible with this release.

If you encounter the CDE crash, uninstall patch 105362-26 and if possible install “105362-20”. If SUN Solaris patch “105362-20” is not installed, the following may occur:

- GUI Hanging Issues
- Paint Problems (i.e. black / white regions may appear in menus, dialog boxes, toolbars, etc.)

If you uninstall patch 105362-26 and encounter any of the issues described above, contact Rational Technical Support if patch 105362-20 is not available.

Running Rose for UNIX/Linux on HP-UX 11.00 with HP-UX OS Patch PHSS_21493

Rose for UNIX / Linux requires HP-UX OS Patch PHSS_17327 (X / Motif 2.1 Dev Kit Mar99 Periodic Patch). This Patch requires PHSS_17326 (X / Motif2.1 Runtime Mar99 Periodic Patch). PHSS_17326 has been superseded by HP-UX OS Patch PHSS_23823.

It is important to use PHSS_23823 as Rose for UNIX / Linux will not start (display) if an earlier version of this patch is installed (i.e. PHSS_21493).

If it is necessary to have a Motif Xlib runtime patch prior to PHSS_23823 and you encounter this issue, refer to the following instructions to workaround this problem:

1. Make sure your Rose process is no longer running. (Use **ps** to determine if the process is running, and “kill -TERM [pid]” if it is necessary to kill the process.)
2. As the owner of these files,

   ```
   cd rational_dir/releases/rose.7.1.9659/install/patches/hppa_hpux/hp11
   ```

3.
   ```
   cp libX11.3 rational_dir/releases/rose.7.1.9659/hppa_hpux/hp11/lib/.
   ```

4.
   ```
   chmod 555 rational_dir/releases/rose.7.1.9659/hppa_hpux/hp11/lib/libX11.3
   ```
5. Run **rose_cleanup**.
6. Run **rose**.

MainWin Runtime Components

When Rose for UNIX / Linux runs for the first time, an **$MWUSER_DIRECTORY** directory is created which includes MainWin files and directories. Rose is created utilizing MainWin (developed by Mainsoft) and requires specific MainWin components to operate correctly. **$MWUSER_DIRECTORY** may be set to serve as the windows directory. For example, **$MWUSER_DIRECTORY** may be set to **$HOME/.windows**. If **$MWUSER_DIRECTORY** is not set, the default location will be **$HOME/windows**.

Within the **$MWUSER_DIRECTORY** directory, you will find the following type of files and directories:

- **mwfc-* -** Font caches, are created incrementally over the duration of the application process; activating when a user chooses a new font. The font cache is built each time Rose for UNIX/Linux is run on a new X server, or if the server's font path is changed. Each font cache file has a unique name:

  mwfc-*xxxx* (where *xxxx* is dependent on the specific display used)

  This file can be removed if it is suspected of interfering with font operations. If removed, it will be regenerated the next time Rose is invoked.

  When Rose for UNIX/Linux is invoked by the user on the same X server, the application can quickly look up the font cache file and select the closest X font available.

- **registry** directory **-** This directory includes a binary "registry" file which is created the first time the user runs Rose for UNIX/Linux (for each version of the Operating System). For example, a "registry.5.6" registry is created for a user running the release on a SUN Solaris 2.6 computer. The registry stores Rose for UNIX/Linux configuration and initialization data.

  The registry directory also includes **rose_reg_check**, an ascii file which includes the version of Rose for UNIX/Linux and it's associated add-ins. This file is created the first time a user runs Rose for UNIX/Linux.

  A specific registry file may be utilized by setting the following environment variable:

  setenv USE_MWREGISTRY */path/myregistry*

- **help -** This directory includes files that get created and stored every time the user runs help and creates the help keyword database.

- **win.ini** file **-** This is a default configuration file. It includes such information as print device/port settings.

  During a Rose session, Rose for UNIX/Linux ***.ini** files may also be created in the user's "windows" directory. These files include Add-In specific settings.

Rational Rose for UNIX/Linux includes "mw" directory

Rational uses MainWin (by Mainsoft) to rehost Rose for UNIX/Linux platforms. Specific MainWin runtime components are included in Rose. Below is a description of the type of MainWin files included in Rose for UNIX/Linux under the *rational_dir***/rel*/rose*/mw** directory.

- **afm** directory - Includes support fonts.
- **bin** directory - Includes programs which perform such functions as determining hardware configuration, removing unreleased X resources, showing X resources, and converting old registries to new.
- **bin-<os>_optimized** directory - Includes binaries designed to register libraries and controls, provides registry browser and support, and type library browser, and font server.
- **fonts** directory - Includes release fonts and font files compatible on both PC and UNIX/Linux displays. See *Installing and Utilizing Fonts Compatible on Both PC and UNIX/Linux Displays* on page 13.
- **fontserver.cfg** file - Font configuration file used by font server.
- **helpfile** directory - Includes help files provide by Mainsoft.
- **lib** directory - Includes scripts to order library paths correctly, and to setup MainWin configuration.
- **lib-<arch>_<os>** directory - Includes an nls directory which holds ***.nls** (Nation Language Support) files that are used upon application startup. It also includes **mwperl** (an executable called upon by the MainWin scripts).
- **lib-<os>** directory - Includes files to manage the font cache, provide support for type library browser and registry browser.
- **lib-<os>_optimized** directory - Includes MainWin support libraries and resources.
- **system** directory - Includes default binary registry and system MainWin registry files.
- **setmw*** and **setup*** files - These files provide appropriate MainWin environment settings.
- **win.ini** is the default Windows configuration file (which gets copied in **$MWUSER_DIRECTORY** the first time Rose is run).

Installing and Utilizing Fonts Compatible on Both PC and UNIX/Linux Displays

Rose for UNIX/Linux now includes the following release fonts which are compatible and may be used on both the PC and UNIX/Linux displays:

- MS Sans Serif
- Arial

- Times New Roman
- Courier New
- Tahoma

As it now is possible to display these same fonts on UNIX/Linux and the PC - models, etc. created on the PC (using these fonts) should display easily and accurately in Rose for UNIX/Linux on PC and UNIX/Linux displays. Furthermore, utilization of these fonts will provide the following additional enhancements:

- Improved dialog scaling on all platforms
- Resolves defects when these PC fonts were not always available on UNIX/Linux displays. Previously, a model created on the PC would utilize an Arial font (not available on UNIX/Linux). Opening the same model on a UNIX/Linux display could produce unpredictable results when the font would map to “Application”.
- Resolves defects when users were creating/sharing models to be displayed on PC and UNIX/Linux displays. Display difficulties could occur when models created or edited on PC display (i.e. using eXceed) were then opened on a UNIX/Linux display.

Font files (in *.pcf - portable compiled format), are now included in the *rational_dir*/**releases/rose.***I.J.K*/**mw/fonts** directory of your release. This directory also includes the following fonts* files:

- **fonts.dir -** The X11 fonts dir file.
- **fonts.mwd -** MainWin fonts dir file. This file is accessed by this release and contains, for each font, the WIN32 metrics, the X11 fontname, a mask stating whether it is a “stock” font, and a “used on unix” flag.

A **fontserver.cfg** file is included in the *rational_dir*/**releases/rose.***I.J.K*/**mw** directory. This file is used when starting the font server to provide the new fonts to the Rose application. A font server (xfs) is also provided in the *rational_dir*/**releases/rose.***I.J.K*/**mw/bin_**<*os*>**_optimized** directory.

Making The Release Fonts Available on Your $DISPLAY

Prior to running Rose for UNIX/Linux for the first time, run *rational_dir*/**releases/rose.***I.J.K*/**bin/rose_cleanup**. Upon starting Rose for UNIX/Linux, you will notice some new messages in relation to the availability of the new fonts. Rose will attempt to determine if the new fonts are available for your $DISPLAY, by running the new Rose for UNIX/Linux **rose_fonts** utility.

UNIX/Linux DISPLAY with visibility to Rose fonts

rose_fonts will make the fonts available to your DISPLAY by including the path to *rational_dir*/**releases/rose.***I.J.K*/**mw/fonts/** in your DISPLAY's font path.

DISPLAY other than UNIX/Linux (i.e. PC using eXceed) or UNIX/Linux DISPLAY without visibility to Rose fonts

In this case, fonts must be made available by using a font server and including the font server in the DISPLAY's font path.

rose_fonts will determine if a font server is providing access to the Rose fonts, as described below:

- Determines if a font server is already running (on the server used to run Rose). There may already be a font server running on the designated font server's port# 7100. On SUN systems 2.5.1 and above it is likely that a font server is already running and defined in the **/etc/services** file, as "xfs" or "fs". The existing font server's default **fontserver.cfg** file will not include a path to the Rose fonts (unless it has been configured to do so).
- If a font server is running, **rose_fonts** determines whether the existing font server is already providing access to the **rose_fonts** (i.e. it determines if the fontserver.cfg includes the path to the Rose fonts provided in the release).
- If a font server is not running, **rose_fonts** attempts to start a font server "xfs".
- If a font server is running and does provide access to the Rose fonts, **rose_fonts** adds the font server to the DISPLAY's font path.

If Necessary To Configure Existing Font Server

If a font server is running on the Rose server and it is not already providing access to the Rose fonts, contact your System Administrator to perform this setup or refer to the man pages for xfs(1). The following advises how to configure an existing font server on SUN Solaris.

Note: Configuring an existing font server to provide access to the Rose fonts may produce less than optimal font results, based on the font paths included in the fontserver.cfg's catalogue setting. This is due to a potential Rose 2000e for UNIX/Linux font mapping issue that is currently under investigation. If you elect to continue with the following instructions, and you encounter poor font results please notify Rational Technical Support and remove the Rose font support from your fontserver.cfg file.

If you elect not to add the Rose font support (as described next) or elect to remove the Rose font support from an existing font server Rose will continue to operate correctly, however the font enhancements listed in *Installing and Utilizing Fonts Compatible on Both PC and UNIX/Linux Displays* on page 13 will not be available and you may encounter the following defects:

Defect 10386
Defect 10387

Adding Rose fonts to Existing Font Server on SUN Solaris

To add Rose font support to an existing font server on SUN (it may be necessary to su to root to perform these actions):

- Edit the default fontserver.cfg file "/usr/openwin/lib/X11/fontserver.cfg" to add the location of the Rose fonts in the "catalogue" path (being careful not to add or remove extra spaces).

  A catalogue path may appear as:

  /usr/openwin/lib/X11/fonts/F3bitmaps/,/usr/openwin/lib/X11/fonts/Type1/

  Add the Rose font path, by prepending the path to the catalogue path.

  For example:

  rational_dir/releases/rose.*I.J.K*/ mw/fonts/,/usr/openwin/lib/X11/fonts/F3bitmaps/,...

  Do not include spaces. Each path is separated by a colon.

- Use **ps** to determine the process ID of the "fs" or "xfs" process, and then stop (kill) the "fs" or "xfs" process, if running.

- Set the DISPLAY font path to include the path to the existing font server by running:

  xset +fp tcp/<*fontserver*>**:7100**

  and then

  xset fp rehash

- Determine if the Rose fonts are available and restart font server:

  fslsfonts -server <*fontserver*>**:7100 | grep -i mainwin**

  The font server must be the current host for it to restart the font server ("fs" or "xfs"). In many cases these x* utilities are located in /usr/openwin/bin.

Rose (upon starting) should indicate that the Rose fonts are now available.

Upon performing this setup, it will be important to update the **fontserver.cfg** file if Rose is uninstalled or moved to a different location (path changes).

Starting a Font Server

If a font server is not running (on the server running Rose), **rose_fonts** will start a font server provided in the release, if necessary. This is necessary if the DISPLAY is not a UNIX/Linux display, or if the display does not have direct visibility to the fonts provided in the Rose release.

X Font Commands Utilized by rose_fonts

The following commands are typically located in one of the following directories: /usr/openwin/bin, /usr/bin/X11, /usr/X11R6/bin, /usr/openwin/lib/X11, or /usr/openwin/include/X11. The font server "xfs" is provided in your release in *rational_dir*/releases/rose.*I.J.K*/mw/bin-*$arch*_optimized.

To Start a Font Server

xfs -config *rational_dir*/**releases/rose.***I.J.K*/**mw/fontserver.cfg -port 7100**

To Set the Font Path

xset +fp *rational_dir*/**releases/rose.***I.J.K*/**mw/fonts/** (direct path to fonts in release)

xset +fp tcp/$host:7100 (font server path)

To Determine if Rose Fonts Available

xlsfonts | grep -i mainwin

xlsfonts may hang if a font path includes a path to a font server which is not running. **rose_fonts** will not let **xlsfonts** run for longer than a specified period of time (typically 30 seconds), to safeguard against this event. If an **xlsfonts** hang occurs, **rose_fonts** will provide status with recommendations.

rose_fonts Overview

In summary, **rose_fonts** will provide the Rose fonts to your DISPLAY by either setting the font path to point to the fonts in the release (UNIX/Linux display with visibility to the fonts), or by setting the font path to a running font server if the font server is providing access to the Rose fonts. **rose_fonts** will use "xset" to set the font path, "xfs" to start a font server (if necessary) and "xlsfonts" to determine if the Rose fonts are available after the setup has been performed. If a font server is already running and does not already provide support for the Rose fonts, manual setup is necessary (i.e. see *Adding Rose fonts to Existing Font Server on SUN Solaris* on page 16) to complete the Rose font setup. In this case, manual set up is required as it is often necessary to su to root to perform this action. **rose_fonts** will provide messages describing the current status each time Rose is started.

Contact Rational Technical Support if you should have any questions or encounter any difficulties accessing or utilizing the new Rose fonts.

If Rose Fonts Not Available

There may be occasions when the Rose fonts will not be available. **rose_fonts** will alert you if, when starting Rose, if the Rose fonts are not available. If this occurs, Rose will continue operate correctly, however the font enhancements listed in *Installing and Utilizing Fonts Compatible on Both PC and UNIX/Linux Displays* on page 13 will not be available and you may encounter the following defects:

Defect 10386

Defect 10387

To Eliminate Output from "rose_fonts" When Running Rose

Each time that you start Rose, **rose_fonts** determines whether the Rose Fonts are available, attempting to make the fonts available when necessary. Each time Rose is started, **rose_fonts** displays messages advising current Rose Font status. In order to eliminate output from **rose_fonts**, set the following environment variable prior to running Rose.

setenv ROSE_NO_FONT_MSG 1

Setting this environment variable will allow **rose_fonts** to run, but will refrain from printing Rose Font status messages.

Licensing Information

There are three license types supported by Rose for UNIX/Linux:

- Startup - This is a time-locked temporary license to allow you to use the software until you receive a production key.
- Node-locked - This allows you to run the software on a single node (computer).
- Floating - This is server-based licensing, where a client logs onto a license server to be assigned a key.

For a full description of license types, installation and usage, please consult the *Rational Rose for UNIX/Linux Installation Guide.*

The software you receive includes a temporary startup license. You must install the software and follow the steps detailed in the *Installation Guide* to acquire node-locked or floating license for your product. For users who are upgrading from Rose 98i for UNIX/Linux, you may use your existing license key to run Rose for UNIX/Linux.

Rational Rose for UNIX/Linux

3

Rational Rose, the world's leading visual modeling tool, allows you to define and communicate a software architecture, resulting in accelerated development, improved quality, and increased visibility and predictability.

The Rational Rose portion of the release notes describes new features and enhancements, known problems and limitations, and fixed defects since the last release. You will also find the Rose Extensibility Interface information in this section.

New Rational Rose Features and Enhancements

This release of Rational Rose includes these new features and enhancements:

- Construction indicators on sequence diagrams. You can drag an object vertically on a diagram to indicate when it was constructed. Once you place the object, you can add a message from the calling object to the new object.
- Destruction indicators on sequence diagrams. A new toolbar icon enables you to add a destruction marker to an object. When you add a destruction marker, a return message to the calling object is created and the object's lifeline and focus of control are ended.
- Support for additional return messages on sequence diagrams. These include procedure call or other nested flow of control, flat flow of control, asynchronous flow of control, and return from a procedure call.
- Docking for the Log Window. You can now dock and undock (float) the Log window. When docked, the Log window is positioned along the border of the application window. If docking is not enabled, or if you drag the window outside the application frame, the window is floating. Use the shortcut menu (right mouse click) to enable / disable docking. To show or hide the Log window, on the **View** menu**,** click **Log** to toggle the display

Note: New and changed Rose add-in features are covered under separate add-in sections of this Release Note.

New Rose Internal Editor

This release of Rose also includes a new internal Editor. The internal editor supports text-editing, language-specific syntax formatting, editor style preferences, and compilation of displayed code for the following types of generated files:

- Rose J .java files
- Rose CORBA .idl file
- Rose XML DTD .dtd files
- Rose ANSI C++ .c, cc, .cpp, ..cxx, .h, hh, .hpp, and .hxx files

The Editor's syntax highlighting defaults to ANSI C++ for files without file extensions.

In addition, for Java, CORBA, and ANSI C++ language elements, the Editor supports automatic synchronization between model elements and their source code.

Rose.ini file replaced by rose.reg

Any settings previously in the Rose.ini file are now in the registry file, rose.reg. This new registry file is located in the rational_dir/releases/rose.I.J.K/registry directory of the release. The settings in this file are entered in your registry, after running “rose_cleanup” and then “rose”. Any changes made to the release rose.reg file will apply to all users running rose. If you wish to make changes specific for your environment, a rose.reg.template file is available in rational_dir/releases/rose.I.J.K/registry. Copy this file to your $HOME directory and name it $USER.reg. Edit this file with the settings you prefer, run rose_cleanup and then rose. Rose will automatically register your $HOME/$USER.reg file and your settings will be available in your Rose session.

Known Rose for UNIX/Linux Problems and Limitations

The following table contains known Rose problems and limitations.

Table 3 Known Rose Limitations

Defect	Description
1718	Add “use qualified name” to all “Add Relation” descriptions.
1719 16674	F1 Help for keywords does not work in the Script Editor. Use the online Help table of contents or index to find the help you require.
1859	Fonts change when sharing units with team members using different default fonts. Default font settings are currently stored with the model, and not with the .cat files. Items will only store font settings in a .cat file if the font is different from the default. As a result, text for model elements in units will be formatted, by default, according to the settings of the model. If one team member formats diagrams relying on a specific default color and font, that formatting will not be rendered for other team members with different default settings. Workaround: Explicitly format text using a font that is different than the default setting.
1977	A business actor loses (changes) its fill color if it is moved around in the browser. For example, change the fill color of the business actor from yellow to green and then move the business actor to a different package.
1981	**Report > Show Usage** does not work correctly.
1998	Problems dragging association point on a diagram.
2062	**Print Preview** is disabled for specifications.
2090	**Print Preview** does not work reliably for all diagram types.
2101	Print Specifications omits classes from the output report, without apparent reason or pattern.
2242	Drag and Drop of multiple selection from browser not supported. The browser can be used to select multiple items, and the context menu can be used for that selection, but drag and drop of multiple items from the browser to a diagram is not supported.
2746	Rose hangs when checking out a controlled unit named with a special character.

Table 3 Known Rose Limitations (continued)

3057	If you change the font size (through **Tools > Options > Diagram** tab) of a model element that is already on a diagram, the model element may not correctly resize. This happens frequently on activity and sequence diagrams. Workaround - You can clean up the diagram using the **Edit > Refresh** command.
3078	Rose is inconsistent in how it handles user attempts to overload element names in different contexts.
3093	It is not possible to drag the “History” icon from the browser to the diagram.
3728	CDE window menu options do not appear when clicking in the top left hand corner menu of Rose. Workaround: Minimize Rose and right click on the icon. CDE options, such as “Occupy Workspace” will then appear.
3861	Scroll bar in Language Properties dialog box may disappear. Workaround: If this occurs, clicking in the scroll bar area will cause the scroll bar to reappear.
3877	Having a <return> character in the language properties (**Tools > Options** “language tab” property fields) can create a code generation problem. In addition, it is difficult to determine if a <return> character exists in language properties as it does not display.
3881	If a Java or Corba specification dialog is displayed and the Rose browser is not docked, Rose will lose focus when the specification dialog is closed. To proceed the user must terminate the Rose process and all unsaved data will be lost.
6397	CDE crash and technicolor affect when running on RDI Ultrabook if PGX graphics patch 105362-20 or higher installed. (May occur with previous PGX patches as well.)
6924	REI: RoseModel.FindItems() does not find attributes or operations.
7156	Encapsulated postscript file created by Rose which includes either multiple diagrams or diagram which exceeds one page size may print on one page creating distorted images.
7871	Text within the brackets on messages are lost on all sequence diagrams when the Apply button is clicked after changing the Message Signature option.
8718	The Fit to Page print feature will occasionally not leave enough space at the bottom of the page for the last element on a diagram. This happens when the diagram, as normally displayed, is just a little too large to fit on one page

Table 3 Known Rose Limitations (continued)

8845	The Test Dialog option in the Dialog Editor is not supported in this version of Rose for UNIX/Linux. Selecting the F5 shortcut key or the Test Dialog Icon will have no affect.
9168	Some custom stereotype icons added-in to previous versions of Rational Rose do not display correctly in Rose for UNIX/Linux.
9229	To change the ROSE_CPP path map variable, you must first manually remove its entry from the registry. In the Windows Registry, this pathmap variable is located at two different locations: ▪ HKEY_CURRENT_USER\Software\Rational Software\Rose\Virtual Path Map ▪ HKEY_LOCAL_MACHINE\Software\Rational Software\Rose\Virtual Path Map
9535	Collaboration diagram numbering is thrown off by deleting and undeleting a link message. Workaround: Turn Collaboration Numbering off and then on again to reset the numbering. (Select **Tools > Options**. Go to the Diagram tab and click Collaboration Numbering off. Click Apply. Click Collaboration Numbering back on and click Apply again.)
10388	Arial font size not consistent. Certain icons with Arial font style does not display the correct font size. If 14pt Arial font size is selected, certain icons (i.e: usecase/business/interface) icons may display in 12pt size.
10622	Print output may not appear the same as the screen display. This issue only occurs when printing from a UNIX/Linux display. This does not occur when printing from a PC display using Exceed.
11133	Fit/Undo Fit in Window may result in undesirable layout. Undo Fit in Window may not result to its original display/layout.
11416	Font scaling difficulties when using Rose fonts. Problem with scaling occurs when doing print preview.
11615	Help button on title bar missing. The **?** button on the title bar of various dialogs (located left of the **X** button) does not appear initially, unless: 1 The area where the **?** button should appear is clicked on. 2 The window area containing the spot where the **?** button should appear is moved off screen, and then moved back on screen.

Table 3 Known Rose Limitations (continued)

11800	You may encounter an unresolved model warning if there is a view of an item on a diagram, but the underlying item is not loaded or could not be found in the current model. Items and relations are correctly displayed and reported by Check Model as unresolved references when the view to the item is in a diagram that is in the current model but the item resides in an unloaded package (controlled unit). The reference becomes resolved and the unresolved indicator is removed when the unit containing the item is loaded.The unresolved model warning occurs occasionally during model loading if the model was created by a previous version of Rose and that version of Rose allowed an invalid model to be created. Some known cases are: Circular Generalize / Realize Relationship Combinations (allowable in Rose2000e but no longer valid in this release). Note that if you attempt to create a circular generalize / realize relationship combination, you will get an error and the relationship will not be created.Circular generalize / realize relationship combinations that exist in models files created with earlier versions of Rose are detected on model load and one of the relationships involved in the circularity is left unresolved. You may delete the unresolved view and then add new ones as appropriate. Running **Tools > Check Model** will provide in a list of unresolved views in the Rose log. If the relationship has no view on a diagram in the model, then an “unresolved relation” warning is placed in the Rose log. The warning is also generated by **Tools >Check Model**. To correct the unresolved relations: 1 Load the model into the Rose 2 Check the log file for “Unresolved relation” warnings 3 Open the spec of the “from” class to the relations tab 4 Note that the unresolved relation (generalize or realize) is the one with the class name in brackets in the name column 5 Determine which classes are involved in the circularity 6 Delete the undesired relationship via the class spec relations tab Shared Objects - Shared activity diagram objects (allowable in Rose2000e but no longer valid in this release). A shared object is an object that belongs to one state machine and has a view on an activity diagram belonging to another state machine. The drag and drop of the object is now disabled if the object does not belong to the same state machine as the diagram.In general, the unresolved object can be deleted from the diagram and a new object can be created on that diagram. The model does not lose integrity or validity for doing this since an object is not actually a model item - it doesn't define anything, it is only an instance of a class used for visualizing state or behavior.

Table 3 Known Rose Limitations (continued)

11810	Printed diagrams can differ from what is displayed on the user's screen. For example, some longer operations do not print completely. If the text on the screen is: **`provideWorkflowClosure(WorkItemClosureData) : void`** the printout might only show: **`provideWorkflowClosure(WorkItemClosureData) : v`**
11879	Display problem with Message\More dialog and long names.
14249	A new **Creates a Form** class diagram toolbar is available, but you must manually add it to the toolbar.
14739	If you use the inline-editing capability on a diagram to rename an operation parameter name, then any OVERRIDDEN model properties associated with that parameter are reset to the DEFAULT value. Note that if you rename the RoseParameter via REI or through the specification dialogs, then the model properties are not reset.
15037	Role Name will not remain on diagram if context menu is used to create it. Workaround: Define the Role Name using the Association Specification dialog.
15230	If R2Editor terminates incorrectly, it will be necessary to exit and re-start Rose in order to bring the R2Editor.
15261	Changing the Syntax Coloring in the R2Editor removes current font setting information for diagram. After changing the Syntax Coloring, no font or point size is assigned.
18698	Rose may crash when selecting a context menu from the Class diagram, after closing the Java Reverse Engineering dialog. This issue is under current investigation. Contact Rational Technical Support if you encounter this issue. See *Contacting Rational Technical Support* on page viii.
18733	Infrequently, a modal dialog may move behind Rose. In this event, you will need to move Rose to access the modal dialog. Contact Rational Technical Support if you encounter this issue. See *Contacting Rational Technical Support* on page viii.
18890	Rose may crash when Web Publishing a model if the following environment variables are not set, *and* csh **unlimit** has been run, prior to running Rose: ▪ MWVISUAL_CLASS=TrueColor ▪ MWVISUAL_DEPTH=24 Workaround: If this occurs, unsetenv MWVISUAL_CLASS and MWVISUAL_DEPTH prior to running Rose.

Table 3 Known Rose Limitations (continued)

145958	After RTE Update the font in Rose is changed to 9 pt.
149496	Virtual Desktop capability not supported on OLVWM.
15230	If the R2Editor process is killed, the user will not be able to open an R2Editor in the same session.
17061	If a state or activity diagram contains a view of an element from another state or activity diagram / model, the query expand feature will not work for that element.
18168	Deleting package with a class attached to a read-only object in an activity diagram will cause Rose to crash
18734	It is not possible to delete an unloaded controlled unit from a model. To delete a unit from a model, first load that unit and then delete it.
23950	When running Rose for Linux x86 on GNOME, Rose may maintain focus. If you encounter this issue, contact Rational Technical Support.
25410	Editor status bar for "Ln" and "Col" may show incorrect text
25408	Unable to open Help file when utilizing the 'More Info...' of Option dialog window
25404	Some diagrams can lose or have additional vertical toolbars.
30165	When a message is created between two existing messages and the sequence numbering is on, the new number is drawn at the very bottom of the diagram. **Workaround**: Manually reposition the message sequence number label.

Fixed Defects in Rose for UNIX/Linux Version 2001A.04.00

Table 4 Fixed Defects

Defect	**Description**
1273	Choosing a pathname with foreign characters may cause Rose to crash.
1724	Add links to REI help from list of shortcut menu events to event detail topics.
1893	On Sequence diagrams, messages do not show up after a paste. The diagram is updated properly when one of the objects is relocated.
3065	On the Component view package specification Detail tab, the diagrams are listed in the Title field but their graphical depictions are absent.

Table 4 Fixed Defects (continued)

3910	Some Context help does not work. Context Help in some Specification dialog boxes are not functioning. When selecting context help in the main Rose window, certain topics are not available. Workaround: Search for Help topics using the Index tab.
3912	The tooltips associated with the Dialog Editor may produce a flashing effect causing screen to also flash.
3921	A Rose pop-up dialog box (requiring user input) may hide behind the parent Rose window. If this occurs Rose will hang, as the pop-up dialog is awaiting input. Workaround: Refer to the suggestions *Configuring Window Manager So Pop-Up Windows Will Remain On Top of their Parent Windows* on page 9 for possible workarounds for this issue.
8128	Window focus issue can occur if using Exceed in XDMCP mode. Contact Rational Technical Support if you encounter this issue. See *Contacting Rational Technical Support* on page viii.
9259	If you delete a model element that has nested elements from a model (deep delete), you cannot undo the action.
10405	Attempting to run help from Rose on some HP-UX 10.20 computers may fail with a “Failed to launch help” message. In addition, a crash may occur if a subsequent attempt is made. Contact Rational Technical Support if this occurs.
10581	Selecting Edit Compartment on a object in a state diagram and then pressing enter will change the compartment of the object to have no items listed. This is because the edit compartment dialog defaults to no items selected and pressing OK accepts that default. To avoid this, if the user wants to keep his previous selections, they should click Cancel on the edit compartment dialog.
11883	Rose crash on dangling role manipulation.
12574	Realize and dependency relationships are not shown in the Relations tab in the use case spec.
12616	The visibility icons for operation do not print correctly.
12630	In the use case view, the drawing of the arrows is not correct, that is, an arrow connected to a use case ellipsis is drawn at the border of the ellipsis. When the model is closed and then opened, the arrow is connected in the center of the ellipsis.
12734	Flows are broken between an activity and object if the activity is in a unit and is unloaded.

Table 4 Fixed Defects (continued)

140938	24-bit stereotype bitmaps don’t work in toolbars. If you use non-standard colors in a 24-bit stereotype bitmap, the icon becomes distorted when you define a stereotype toolbar button or a list (browser) icon.
14249	A new class diagram toolbar button called “Creates a Form” is available, but you must manually add it to the toolbar.
15796	Rational SoDA will show "Implementation" access types for class attributes and operations appear as "Private"
18196	Reduced number of fonts available in R2Editor.
18735	Running **rs_help** and clicking on a link to a PDF format file may cause Netscape to attempt a “Save As...” on the file rather than launching Adobe Acrobat to read the file. Workaround: The best method to view the PDF files provided in this release is to enter: *rational_dir***/releases/rose.7.1.9701/rs_help -pdf** Once Acrobat is visible, select **File > Open** to open the document of choice.

Rose Extensibility Interface (REI) Release Notes

Rose Extensibility Interface Features and Enhancements

The Rational Rose Extensibility Interface (REI) provides several ways for you to extend and customize Rose's capabilities to meet your specific software development needs. Using REI capabilities, you can:

- Customize Rational Rose menus
- Automate manual Rational Rose functions with Rational Rose Scripts (for example, diagram and class creation, model updates, document generation)
- Execute Rational Rose functions from within another application by using the Rational Rose Automation object (RoseApp).
- Access Rational Rose classes, properties and methods right within your software development environment by including the Rational Rose Extensibility Type Library in your environment.
- Activate Rational Rose add-ins using the Add-In Manager

There are no new features or enhancements for this release of the Rose Extensibility Interface (REI).

Netscape V4.7x Issues

Netscape 4.7*x* is provided in the *rational_dir*/base/cots/netscape.4.7*x*/*$arch* directory of this Release. **rs_help** uses Netscape to display Installation Guide and Release Note information. Rose for UNIX/Linux will occasionally use a browser. Rose will first use the browser defined in the users "BROWSER_PATH" environment variable or Netscape included in this release, if BROWSER_PATH is not defined.

If you have not used Netscape 4.7*x*, we recommend you consult the Netscape README file located at

rational_dir/base/cots/netscape.4.7*x*/*$arch*/README

This file includes a warning, recommending that an existing **$HOME/.netscape** file or directory be renamed prior to running Netscape 4.7. Otherwise, data may be lost (i.e. old preferences, bookmarks, cookies, etc.). The README file includes other helpful information such as platform specific issues as well.

ANSI C++ Add-In

4

Rose ANSI C++ is the new Rational Rose language add-in in support of the C++ programming language.

Because of its many benefits, users are encouraged to convert their models from Rose C++ to Rose ANSI C++. Rose ANSI C++ is designed to be powerful enough to handle large projects, with particular emphasis on scalability and completeness of language support. The user interface has been completely overhauled to make it easy to understand and use.

A model converter is included as an add-in with this release, and the ANSI C++ online help provides instructions for this easy conversion.

Rose ANSI C++ Features and Enhancements

Rose ANSI C++ provides:

- Design, modeling and visualization of all C++ constructs including classes, templates, namespaces, inheritance and class members functions.
- Support for the C++ language, without being restricted to a single vendor’s C++ compiler
- Support for large frameworks
- Automatic generation and reverse engineering of C++ source code
- User-controlled code generation using patterns of default constructors, destructors and class members. One-step reverse engineering from Rose, without a separate import step
- Style sheet mechanism to allow custom formatting of generated code
- Simplified, yet complete round-trip engineering support, which synchronizes models and generated C++ code across multiple iterations. After assigning files or classes to a project, you can forward or reverse engineer the entire project using a single context menu command.
- Update-in-place mechanism to determine the differences between model and code and make only those changes required to maintain consistency between the two.

Batch File Load

- This release adds a Batch Load feature to the **Files** page of the the ANSI C++ **Component Specification** dialog. The purpose of this feature is to allow you to specify a file containing an arbitrary list of files to be added to the component. The Add Files button, by contrast, brings up a standard file selection dialog, and requires you to add files individually, or by selecting multiple files within the same directory with the mouse. Add Files does not support adding files from multiple directories or sub-directories.

Progress Bar/Cancel

- A progress bar is now created when performing forward and reverse engineering. This progress bar gives an approximate indication of the amount of processing remaining. The **Cancel** button is displayed along with the progress bar and allows you to stop the code generation or reverse engineering operation. The **Cancel** action terminates any reverse engineering or code generation operation, which is in progress; it will not undo operations that have already completed.

Macros

- The Macros dialog allows you to predefine macros, which will be expanded when the code is reverse engineered. Macros are entered in table format on the last page of the ANSI C++ Component Specification dialog. Each ANSI C++ component has its own list of macros and associated values. You can define an arbitrary number of macros.
- There are three ways of filling the table with macros and their values:
    - Enter the macros in the table manually. Use the Enter key to move to the next row, and the Tab key to move to the next column. You may also use the keyboard arrows to navigate the table. When the last row of the table is filled in, a new row is automatically created.
    - Press the **Open File** button and selecting a single file to parse any #define preprocessor directives in the file and add the macros to the table.
    - Press the **Auto Load** button. This will parse all the files currently assigned to the component, find all the #define directives, and add those macros to the table.
- You can save the macros currently listed in the table to a file by pressing the **Save As** button. Using **Save As** and **Open File** is the best way to copy macros from one component to another.

- When source files are parsed, their text is partially preprocessed, causing the macros defined in that particular component (and only those macros) to be expanded. The model that results from reverse engineering that component will correspond to the text of the expanded macros. While parsing, the add-in will not recognize #defines in the code; all macros must be explicitly defined in the component. When generating code, the add-in will generally not attempt to change code that was the result of a macro expansion.

AutoSync

- While in AutoSync mode, most changes to ANSI C++ classes are automatically propagated between the model and the code. Changes in the model will be forward engineered as they are made. Code changes are picked up and reverse engineered when the user saves a C++ file in the source code editor provided with Rose. AutoSync mode is turned on or off in the ANSI C++ **Preferences** dialog, accessible by clicking **Tools** > **ANSI C++** > **Preferences**. During AutoSync, Rose behaves as usual except for the automatic propagation of changes. All commands, including the traditional batch mode code generation and reverse engineering operations, may be performed. Code generation and reverse engineering caused by AutoSync behave normally, except that the dialogs allowing the selection of classes to be synchronized do not appear. You must use the Rose internal editor if you want code changes to be automatically synchronized. Code changes made with a different editor must be synchronized with a reverse engineering operation. To use the internal editor, in the ANSI C++ **Preferences** dialog, select **Default** as the **Editor**.

Known ANSI C++ Defects

Defect	Description
8350	Reverse engineering with the ANSI C++ add-in does not create a diagram that contains all reverse engineered classes. Although they do not appear on a diagram, the reverse engineered classes do exist in the model. You can see them in the browser and can add them to a diagram with drag and drop, or by using the Query | Add Classes menu item.
9071	The ANSI C++ add-in displays any errors encountered during code generation or reverse engineering in the Rose log window. However, there is no message telling the user to check the log window for errors.
9084	The ANSI C++ add-in will create two unidirectional associations on reverse engineering when the model originally has a bi-direction association between two classes and both roles of the association are named.
10091	If the stereotype of an operation in the model is changed to <<const>>, and the return type of that operation is a pointer to a function, then the "const" will be inserted into the code in the wrong place.
10776	When generating code, Rose ANSI C++ does not make backup copies of files it changes, nor does it provide an undo mechanism for such changes. You are strongly encouraged to backup your source files or have them under an effective source code control mechanism before generating code.
11115	If a typedef defines more than one name, the ANSI C++ add-in does not reverse engineer the typedef. For example, reverse engineering the following code completes without error:

11115

```
>>>
typedef struct myStruct {
            DWORD                   Version;
            DWORD                   Size;
} myName, * lp_myName;
<<<
```

However, the struct myStruct appears in the model, but the typedef does not.

Defect	Description
14630	The ANSI C++ add-in ignores comments in source code.
15120	After you convert a class or model from Classic C++ using the ANSI C++ Model Converter, you generate code for the class or model before attempting reverse engineering. Attempting to reverse engineer before generating code will cause this message to be displayed: Class cannot be reverse engineered because it does not exist in the project files.

Defect	Description
16714	**Apply** button on component dialog is never enabled. Workaround: "OK" button is available and enabled.
17114	When a template class is deleted from the code, and then the code is reverse engineered, the corresponding template class in the model is deleted; however, anonymous classes that are instantiations of that template are not deleted.
17513	When indicating "vi" as the editor of choice in the **Tools > ANSI C++ > Preferences** dialog, it is necessary to invoke vi within the context of it's own window, by entering the following command in the preferences. **xterm -e vi command_line_args** Entering "vi", instead of the command listed above will start vi if an xterm window is currently open and will display the source but it then receives an 'input read error' and exits vi. emacs and gvim will work as expected.
17844	The initial values of static attributes are not updated during either code generation or reverse engineering. The initial values are used when an item is created for the first time in either the code or the model, but are not changed if the item already exists.
18360	Creation order of classes affects code generation from class (but not from component). If a class containing a data member is created before the class being contained and code is generated by selecting both classes on a class diagram and choosing **Tools > ANSI C++ > Generate Code**, the #include for the contained class is not generated. If code is generated from the component (not the classes), the #include is generated. If the contained class is created first and code is generated from the class diagram, the #include is generated.
20639	Types nested within template instantiations may not be used as base classes or the ends of assocations, since such types are not modeled. Reverse engineering such code can result in an incorrect model. For example, if the following code is reverse engineered, the base class of class c will be "a<b>", not "a<b>::z".

```
template <class t> class a {
     typedef t z;
 };
 class b { };
 class c : public a<b>::z { };
```

21809	Automatically generated `#include` directives are always relative to the source file root directory. There is no easy way to force them to always be simple names or to make them relative to some other directory.
23938	Multiple code generations on code including the “friend” type will cause the “friend” keyword to appear multiple times.

Defect	**Description**
24206	With AutoSync enabled, when a class in the model is changed from being at package scope to being a nested class, its new definition is correctly added to the code of its containing class, but its old definition in the code is not deleted.
25046	Rose ANSI C++ does not work with components whose stereotype is one of Main Program, Package Body, Subprogram Body, or Task Body. Components with such stereotypes cannot be used for code generation or reverse engineering with Rose ANSI C++.
25564	ANSI C++ dialogs may contain incorrect dialog title of “libcppdll.so”.
31276	Initial value for role added to header file during subsequent code generation.
31350	Argument type cannot be updated when Autosync is turned on.

Fixed Defects in ANSI C++ Version 2001A.04.00

10743 29762	When a function parameter is changed in the model and code is generated, the corresponding change is only made in the member function declaration - the *.cpp file is not updated.
11549	If an existing class does not have any member functions or static data members in an implementation (.cpp)file, the ANSI C++ add-in generates the function body for any new functions in the header (.h) file.
11689	Automatic code generation does not create new #included statements when existing code elements are modified via model edits. It only creates them when new items are added.
11961	The ANSI C++ add-in does not create directories during code generation. If you try to generate code into a directory that does not exist, a message will be displayed that says the file cannot be created.
12569	In a .cpp file containing only a #include statement is reverse engineered, the code generator will place this #include statement in a header file on forward engineering.
15144	When attempting to generate code for a collection of classes, if the Root Directory field in the code generation dialog remains empty, a warning dialog is presented once for each of the selected classes
16428	The ANSI C++ add-in does not create directories during code generation. If you try to generate code into a directory that does not exist, a message will be displayed that says the file cannot be created.

16429	If a typedef defines more than one name, the ANSI C++ add-in does not reverse engineer the typedef. For example, reverse engineering the following code completes without error:

```
>>>
typedef struct myStruct {
        DWORD                   Version;
        DWORD                   Size;
} myName, * lp_myName;
<<<
```

However, the struct myStruct appears in the model, but the typedef does not.

16430	The ANSI C++ add-in displays any errors encountered during code generation or reverse engineering in the Rose log window. However, there is no message telling the user to check the log window for errors.
16432	Reverse engineering with the ANSI C++ add-in does not create a diagram that contains all reverse engineered classes. Although they do not appear on a diagram, the reverse engineered classes do exist in the model. You can see them in the browser and can add them to a diagram with drag and drop, or by using the **Query > Add Classes** menu item.
16434	If an existing class does not have any member functions or static data members in an implementation (.cpp)file, the ANSI C++ add-in generates the function body for any new functions in the header (.h) file.
16449	The ANSI C++ add-in ignores comments in source code.
16452	The Class Customization feature can generate two get or set operations for the same attribute if the user invokes Class Customization twice, once generating Get by Reference and once just generating Get.
16457	After you convert a class or model from Classic C++ using the ANSI C++ Model Converter, you generate code for the class or model before attempting reverse engineering. Attempting to reverse engineer a class before generating code will cause this message to be displayed: Class cannot be reverse engineered because it does not exist in the project files.

16463	A previous version of Rose could create multiple module lines for C++ classes in certain cases. These extra module lines cause the Convert from Classic C++ to fail with this message: The Class / Interface <class name here> is assigned to an unloaded Component, which has the language C++. Classes / Interfaces may only be assigned to components with the same language. To convert a model that has multiple module lines, open the model in a text editor and remove the offending lines. The lines that should be removed contains two words: "module" and the name of the module. For example: module "ABC" module "ABC" module "ABC" module "ABC" module "ABC"
16550	The ANSI C++ add-in cannot expand pathmaps that are defined using the reserved symbol "&". . Using a pathmap that is defined with "&" for the code generation root directory or the source file root directory will cause this error message when you attempt to close the ANSI C++ Component Specification dialog: Directory <Pathmap symbol here> does not exist. Please reenter.

16552

The add-in can create the wrong operation signature in the model as a result of reverse engineering. This occurs if the code contains non-const, on-abstract member function declarations (i.e. one where the closing parenthesis is followed by a semicolon), where the last two parameters are abstract declarations(i.e. they don't have an identifier) that contain a single type name.

For instance, when reverse engineering:

```
class a { };
class b { };
class c { };
class x {
  public:
    void f(a, b, c);
};
```

the operation x::f in the model has the signature "f(:a, c:b): void",

which is wrong. The signature should be "f(:a, :b, :c): void".

On code update, the original signature is preserved, so the code is not changed. Only the model is incorrect.

Workaround: Change the code so that the function parameters are named.

For example, change

```
void f(a, b, c);
```

to

```
void f(a arg1, b arg2, c arg3);
```

Since parameter names are optional in function declarations, this will not affect the meaning of the code.

17848	The TYPENAME keyword is not supported.
19170	AutoSynch does not remove the data member in the header file when removing its corresponding attribute in the Rose class.
19210	When AutoSynch is enabled, adding an operation to the class produces the correct declaration of the operation, but the operation body is also defined in the header file instead of in the .cpp file as one would expect.
19309	Rose is unable to browse the .cpp file with the editor after generating code and then reverse engineering.
21101	Converting from Classic C++ fails if a class is assigned to **C++ Package Body and Specification** components.

26191	Rose may crash when ANSI C++ reverse engineering is performed, using the following scenario:
	1 Start a new Rose (ANSI C++) model
	2 Create a class named "test"
	3 Create a component and assigned "test" to the component
	4 Mouse right click on the component and select ANSI C++ -> Open ANSI C++ Specification
	5 Enter the path for "Code generation root directory:" and click OK to exit
	6 Mouse right click on the component and select ANSI C++ -> Generate Code
	7 Start a new Rose (ANSI C++) model
	8 Create a component select ANSI C++ -> Open ANSI C++ Specification
	9 Click "Add Files" button.
	10 From the "Project Files:" list view window, click on the "..." button.
	11 Use the dialog to locate and select test.h and test.cpp produced earlier.
	12 Click OK to exit ANSI C++ Specification dialog
	13 Mouse right click on the component and select ANSI C++ -> Reverse engineer
	14 Drag and drop reverse engineered "test" class to the diagram. Note that there should no attributes/operations at all.
	15 Open test.h from i-editor and add an operation to test class as follows:
	class test
	{
	myFoo();
	};
	16 Save test.h and mouse right click on the component and select ANSI C++ -> Reverse engineer. Click OK to the class list dialog. Observe that myFoo() operation has been added to the class
	17 Open test.h from i-editor and change myFoo() operation to yourFoo() as follow:
	class test
	{
	yourFoo();
	};
	18 Save test.h and mouse right click on the component and select ANSI C++ -> Reverse engineer.
	Click OK to close the class list dialog.
	Rose may crash at this point.

Rose C++ Add-In

(Not Available for Linux)

5

Note: While Rose C++ is still supported, there are many advantages to working with the new Rose ANSI C++ add-in, and we encourage all Rose C++ users to consider using it. A model converter comes with this release, providing an easy transition from Rose C++ to Rose ANSI C++. For details, see the release note information on the Rose ANSI C++ add-in. **This will be the final release of Rose C++.**

Using the Rational Rose C++ add-in, you can produce C++ source code from the information contained in a Rational Rose model. The code generated for each selected model component is a function of that component's Specification and code generation properties, and the project's properties. These properties provide the language-specific information required for mapping your model onto C++, and allow you to control the code generated for each component.

The following sections provide information on the Rose C++ Add-in:

- *C++ Code Generator Versioning* on page 44
- *C++ Version Comment Orphaned* on page 45
- *C++ Annotation Upgrade* on page 45
- *C++ Annotation Downgrade* on page 46
- *C++ Minor Version Downgrade* on page 46
- *C++ Add-In Known Problems and Limitations* on page 47

Upgrading from a Previous Rose C++ Add-In Release

When upgrading from a previous release, it is important to understand the various versioning considerations associated with the upgrade. To help you stay in control when regenerating code of a different version, Rose C++ generates a version stamp and includes it in the generated code. Refer to the *Rose C++ Code Generator Versioning* online help to learn more about this feature.

C++ Code Generator Versioning

The C++ code generator uses annotations (protected regions) in the code in order to support non-destructive code regeneration (code updating). The C++ Analyzer takes advantage of the annotations as well.

As the code generator advances using code generators of different versions, the same code base becomes problematic. This is the case when the annotations format changes. A code generator versioning stamp and a surrounding mechanism has been introduced to address this problem.

A particular code generator will produce a version stamp that characterizes the annotations format used and the version of the code generator relative to this annotation format. They are called annotation version and minor version. The version stamp itself has the form of a protected / preserved region. It is not meant to be modified or deleted by the user in any way because it is used differently than other preserved regions. Text added in the region will not be maintained. In addition, the preserve tag must not be set to false.

A particular code generator version will only write the newest version of annotations. However, it can read and interpret older versions of the prior annotation version. How far the backward compatibility goes depends on the kind of change introduced. The code generator will issue a warning message when an annotation upgrade occurs. Refer to the C++ Annotation Upgrade topic to read more about the implications of upgrading the annotation format. Upgrading to a higher minor version is the recommended path and does not cause any problem because it does not require any transformations. The annotation format has not changed.

Downgrading code, which is updating code produced by a higher version code generator, is a risk. The exact risk depends on the usage of newer code generator features and how models that produce the code are going to be shared. In order to allow the user to decide what downgrading risk to engage, two Boolean project properties AllowGenerateOverNewerMajorAnnotations and AllowGenerateOverNewerVersions have been introduced. The default property sets assumes that no risk should be taken. They are both set to false. If the property does not allow overwriting newer code, the code generation displays an error message and skips the module. Otherwise, it will issue a warning and continue.

There are some additional considerations when using an older code generator that is not aware of version stamps. The first version of the version stamp aware code generator is version 1.1. If it does not find a version stamp, it will assume that the file had been written by an older code generator. The behavior is as if found a stamp denoting version 0.0. The C++ version comment orphaned topic contains more information about the effects when using a version 0.0 code generator on newer version stamped code.

C++ Version Comment Orphaned

If you are using an older, version stamp unaware code generator to update version stamped code (C++ code generator versioning) the version stamp comment will end up in the orphaned section at the end of the module. This raises the same issues as described in the C++ annotation downgrade topic. The recommended action is to use the newer code generator.

It is likely that other preserved regions will also end up in the orphaned section. If you decide to undo this operation, you will have to manually revive the written backup files.

C++ Annotation Upgrade

A major annotation upgrade happens when code is updated that had been written with an older code generator that supports an older annotations format.

An annotation upgrade is introduced to put more information in the code in order to find preserved regions properly in complex situations. For example, support for recognizing the preserved regions for property generated operations (such as assignment operators) copy constructor when the class name changes.

Changing the annotation format has disadvantages, which is the reason why they occur only after careful consideration of alternatives. One disadvantage is that upgrading to a new format can create problems if the upgrade occurs at the wrong time. Whether this is true or not depends on the particular format change introduced. In addition, upgrading is in general restricted to a particular number of versions.

A particular code generator will support upgrading from the next lower annotation version under stable conditions. Stable conditions means that the upgrade should occur on code that has been successfully updated using the code generator from which to upgrade. If the current code does not fit the model, upgrading and synchronizing at the same time may fail. The steps to assure stable conditions are:

1. Update the code with the code generator version you want to update from.
2. Do not change the code and the model.
3. Update the code with the target code generator version.

C++ Annotation Downgrade

A major annotation downgrade occurs when you update code that had been written with a newer code generator supporting a newer annotation format.

Refer to the C++ annotation upgrade topic for reasons why the annotations format may change. It must be expected that the older code generator may not be able to recognize the code regions with the newer annotation. It is particularly problematic to share code using code generators, which support different annotation versions. The exact risk depends on the usage of newer code generator features and how models and especially the produced code are going to be shared.

Downgrading annotations is neither recommended nor automatically supported. You may have to do manual work.

However, you may have valid reasons to dismiss all these considerations. Setting the Boolean project property AllowGenerateOverNewerMajorAnnotations to True will allow you to perform the downgrade. The code generator will still insist to issue a warning though.

The default setting is false, which makes the code generator to issue an error and to skip this module.

C++ Minor Version Downgrade

A minor version downgrade occurs when you update code that had been written with a newer code generator, which supports the same annotation format.

Minor downgrades may introduce problems if your current model/code depends on capabilities that have been introduced in the newer code generator. Note that minor upgrading will remedy those problems. Therefore minor downgrades will not raise problems which may require manual work. However, you should allow the downgrade only if you are sufficiently certain that you are not causing problems.

The capability to honor a minor version attempt can be controlled in the same way as the annotation version downgrade. So even though minor downgrading is far more reasonable, the behavior regarding error/warning messages and default setting is the same. Setting the Boolean project property AllowGenerateOverNewerVersion to True will allow you to perform the downgrade. The code generator will still issue a warning in order to make you aware of the potentially unintended downgrade. The default setting is false, which makes the code generator issue an error and skip this module.

C++ Add-In Known Problems and Limitations

The following table contains known Rose C++ problems and limitations in Rose for UNIX/Linux.

Table 5 Known C++ Defects

Defect	Description
3861	Scroll bar in Language Properties dialog box may disappear. Workaround: If this occurs, clicking in the scroll bar area will cause the scroll bar to reappear.
3877	Having a <return> character in the language properties (**Tools > Options** "language tab" property fields) can create a code generation problem. In addition, it is difficult to determine if a <return> character exists in language properties as it does not display.
6707	Argument, parameter, and result type specifications are textual even when they are references to elements in the model. This gives you the impression that they will change as you change the elements referenced (like other fields in the model). Because they are just text, they do not.
145958	After a round-trip engineering update the font in Rose is changed from whatever it was to 9 pt.

Fixed defects in C++ Version 2001A.04.00

Table 6 Fixed Defects

Defect	Description
6209	The Analyzer does not handle most template specializations. How they should map to a UML model has not been determined. They probably should be put in protected regions by Action:CodeCycle, but they are not.
21233	Preserved region data not retained correctly when generating Ada83, Ada95 or C++ code.

Rose J Add-In

6

The Rose J Add-In supports true round-trip engineering and has support for large frameworks.

The following sections provide information on the Rose J Add-In:

- *New Rose J Add-In Features and Enhancements* on page 49
- *Known Rose J Add-In Problems and Limitations* on page 51
- *Fixed Rose J Add-In Defects* on page 54

New Rose J Add-In Features and Enhancements

The Rose J add-in supports true round-trip engineering and provides support for large frameworks. This release of the Rose J add-in provides the following new capabilities:

- Enhanced comment support
- Javadoc is fully supported; controlling Javadoc tags is easier and more complete, and includes simplified controls for user-defined tags
- Default return line generation
- Classpath definition is simplified and includes autosearch capabilities
- Link integrations with IBM VisualAge for Java (Linux x86), Forte for Java (Solaris and Linux x86) , JBuilder (Solaris and Linux x86)
- For J2EE:
    - Greatly simplified EJB and Servlet capabilities
    - EJB 2.0 support including message-driven beans
    - EJB method helpers

Rose J Design Pattern Support (Not Available on HP-UX 10.20)

When you apply a design pattern to your model, Rose J adds it in the form of new classes and components for each participant in the design pattern. Rose J places new Rose elements (classes, methods, packages, relations, etc.) in the Logical View and adds a component in the Component View for every added or updated class. The component name also appears in parentheses beside the class name in the Logical View.

When you forward engineer your model into Java code, Rose J populates the design pattern participant classes with methods that have complete code bodies, as well as methods with empty code bodies to be filled in by you.

Rose J includes support for the following design patterns covered in the book, *Design Patterns, Elements of Reusable Object-Oriented Software* (ISBN 0-201-63361-2), by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, often referred to as the "The Gang of Four."

- Creational Patterns
    - Abstract Factory
    - Factory Method
    - Prototype
    - Singleton
- Structural Patterns
    - Adapter
    - Bridge
    - Composite
    - Decorator
    - Facade
    - Flyweight
    - Proxy
- Behavioral Patterns
    - Chain of Responsibility
    - Command
    - Iterator
    - Observer

- State
- Strategy
- Template Method
- Visitor

Known Rose J Add-In Problems and Limitations

The following table contains known Rose J problems and limitations in Rose for UNIX/Linux.

Defect	**Description**
10988	Get/set operations are not visible in Sequence Diagrams.
11721	Unable to create directory on an NFS mounted disk when defining source directory using Java or CORBA project specification.
12237	Code generation error when placing a static attribute in the Java inner interface.
12950	Rose J uses a code generation property to indicate if an operation is abstract, rather than the model property used by the Rose specifications. For this reason when you specify an operation as abstract in the Standard Specification this is not reflected in the Custom dialog box. The reverse is also true.
13935	Utilizing tabs in the Rose J style specification causes errors when round trip engineering. Workaround - Utilize spaces instead of tabs.
14628	Reverse engineering of very large Java files, can cause memory issues.
15230	If the R2Editor process is killed or crashes, user will be unable to restart the R2Editor in the same session. Workaround: Close and restart Rose in order to restart the R2Editor.
15420	Code generating for EJB's does not prompt user for all classes at the same time (only prompts for one at a time).
15481	No warning dialog to indicate there are warnings in log if in Java Auto Sync Mode.
15487	Java code gen does not begin when class is created with Auto Sync already enabled.
15554	When reverse engineering .cab files (which include .java and/or .class files), the method body contents of the source files is missing.
15583	The R2Editor displays carriage returns and line feeds as ascii characters. This occurs in the output window only, not in the R2Editor, itself.

Defect	**Description**
15891	Reverse engineering of .cab files may produce unpredictable results. When reverse engineering .cab files (which include .java and/or .class files), the resulting Rose model may not be correct. The Rose log records errors, but does not alert the user.
16018	No context sensitive help for EJB & Servlet Class Properties treeview items.
16021	No context sensitive help for object finders labels in EJB Persistence Properties window.
16032	Utilizing the "Generate Fully Qualified Type" option on Container Class type, causes incorrect code generation to occur. Workaround: Do an import for the package in which the Container Class type is defined and do not use the "Generate Fully Qualified Type" for the Container Class.
16037	If you have a package in your model that has a space in its name, for example "w p" and a class attribute references a type defined in that package, the generated code will include the space and will be incorrect.
16295	JavaDoc tags are not generated correctly.
16391	In the Rose J online help, obsolete Java frameworks are mentioned. When you start Rose, check the Frameworks dialog to see the extant Java frameworks.
16490	When performing compilations using the R2Editor, a message indicating "Compile Operation Completed successfully", will occur when the compilation was actually unsuccessful due to compilation errors.
16517	The online Help states that the `EJBHomeMethod` is updated using the Update EJB to Interface and Update Interface to EJB. However, the stereotype with this version of the specification is an abstract stereotype that is specialized by `EJBCreateMethod` and `EJBFinderMethod` where an update of the model is available. This stereotype is not related to the concept of Home Method introduced by version 2.0 of the EJB specification.
16593	EJB DocComment - extra newline erroneously inserted and inconsistent indentation in code. To reproduce: Create a class and convert it to EJB. Enter multiline comments in the DocComment fields for the class, for the operation and attributes. Generate code. Extra new lines are inserted between consecutive lines of comment in the model. The indentation is also inconsistent.
16663	No context sensitive help for EJB & Servlet Class Properties treeview items.

Defect	**Description**
16665	Incorrect EJBHomeMethod information in Rose J help. The online help states that the EJBHomeMethod is updated using the Update EJB to Interface and Update Interface to EJB. However, the stereotype with this version of the spec is an abstract stereotype that is specialized by EJBCreateMethod and EJBFinderMethod where an update of the model is available. This stereotype is not related with the concept of Home Method introduced by the 2.0 EJB spec. As a reminder the version supported by the Add-in is the 1.1 EJB spec.
16809	Browsing source (R2Editor) which includes a line exceeding 16384 characters may crash the R2Editor.
16881	The R2Editor's compiler will not detect major modifications in Java source code.
17256	Unable to create new EJB / Servlet from statechart icon.
18021	Putting EJBPK in the New EJB dialog Remote field causes terminal loss of focus.
18195	The Java code generation OK prompt (modal dialog) can infrequently go behind Rose. If this occurs, it will be necessary to move Rose to respond to the modal dialog - before proceeding with Rose activities.
20257	Pattern Support -> Singleton - Singleton interface should implement java.lang.RunTimeException when exception is thrown.
21096	For J2EE, EJB classes and the diagrams that contain them must be in the same package in order for the EJB **Update** to work.
24389	The `rjavarelay.exe` will not exit unless you properly shut down the Rose VAJ Link Plugin Toggle. To ensure proper shutdown, do one of the following before exiting Visual Age: ▪ Use the Quick Start Menu (F2) to turn offthe Rational Rose VAJ Link Plugin Toggle. ▪ In Workbench, click **Workspace** > **Tools**. Similarly, when you want to use the Rose VAJ Link, turn on the toggle, either from the Quick Start Menu (<F2>) or the Workbench **Tools** menu before you select **Rational Rose Update Model** or **Rational Rose Show Model** from the context menu under **Tools** in Visual Age Java.
26170	Inserting new EJB method when autosync is activated may cause Rose to crash.
31374	Unable to delete Finalizer method from browser after turning on Autosync.
31620	Changing a class attributes Bean Property after saving the file in the editor does not update the code correctly.

Fixed Rose J Add-In Defects

The following table contains Rose J Add-In defects that have been fixed since the previous release.

Defect	**Description**
3881 19639	Rose may hang if a Rose J or CORBA Operation and Attribute Specification dialog box is viewed while the browser is undocked. Rose is unable to gain focus, once the Specification dialog is closed.
12122	If you define a component hierarchy that is different from the logical hierarchy, Rose J will generate incorrect code. The package is specified according to the component hierarchy; the return type is according to the Logical View.
12323	Comments will be missing after forward engineering with Rose J. If a user has an instance variable and defines an in-line comment "//" after the line or a multi-line comment "/**/" above the line, then when forward engineering is done these comments are silently lost. This is true for comments anywhere, as they are not associated with a Rose element.
13318	Utilizing the Java Specification window may cause component to lose it's stereotype.
14927	"Remote Exception" should automatically be placed in the Detail Exception.
15493	Selecting **Show Tabs** in the Options dialog does not cause the editor to show tabs.
15518	Generating the `Setter` method for the `Bean` property does not properly update the `Setter` method when there is a change in the `Bean` property type.
15794	Using an interface from a class with the same name, but created in a different package will cause incorrect code to be generated.
15853	Jar tool location and creation of *.jar and *.war file name must be entered using the browse "..." button.
16258	**APPLY** button not enabled for Java class, operation, and attribute specifications.
16417	Reverse engineered Java code that does not contain a visibility indicator (public, private, etc), will be represented as package visibility in Rose
16433	Java AutoSync feature does not appear to receive all events from Rose model. The AutoSync events need to be first updated by Rose, and then the Rose J Add-In needs to be updated to perform correct processing.
16482	Rose J code generation does not remove the link in an inner / nexted class. Workaround: Do not nest the classes or do not enable AutoSync.

Defect	**Description**
16485	Code generation incorrectly occurs when an inner class includes static members. Rose displays an error "... - an inner class cannot have any static members", but proceeds to generate code.
16492	A model is not automatically updated via Autosync when a new J2EE action is performed. Some code generation aspects may therefore be missing in the model.

16498	Rose J reverse engineering error "Parser Error: near token }" occurs with the following code:
	public interface test
	{
	public char opname();
	
	/**
	* comment to see if will get an error while parsing
	*/
	}
	Workaround: Do not use "/**" as a comment, "/*" will work.

16524	Comments are not generated correctly for EJB/Servlets.
16526	Error occurs in specification of "throws" classes. When specifying an operation via the "Method Specification" dialog that is obtained by double clicking on the method itself in the model browser window, the user can only enter exceptions in the "throws" list box by explicitly specifying the exception class with its package. It is not possible to pick the exception in the popup browser dialog because this only displays packages and not actual classes within them.
16664	No context sensitive help for object finders labels in EJB Persistence Properties window.
16764	J2EE - Servlet and DoGet Method causes a duplicated signature for the doGet method. This occurs by creating a new Servlet, adding the methods "init" and "doGet" and adding a value for the buffer size. Workaround - Add a "Servlet Content Type" or create a normal class and configure it to be a servlet (the Servlet content Type is added by default).
16780	Java code generation does not code the relation to an inner class correctly.
16902	Servlet global import statements do not reverse engineer properly from .class files. Upon creating a Servlet with an operation (for example, `doPost`) global imports may be missing.

Defect	Description
17174	Rose adds changed interface method instead of changing the current method. If a class implements an interface, the interface methods are added to the implementation class automatically by Rose during code generation. If the interface signature changes, Rose does not change the implementation class but adds again the changed interface methods without warning or messages. This means the old interface method becomes a regular method still containing the implementation and the "new" interface method is empty. Workaround: To prevent this behavior, delete the methods from the implementation class before starting a new code generation.
17275	Parser error occurs when attempting to reverse engineer a class which contains an anonymous inner class that extends a non anonymous inner class, belonging to another class.
17375	Dragging and dropping a .jar file into Rose in order to reverse engineer Java files will prevent the user from forward engineering unless the .jar file is removed from the path statement in the Java Project Specification directory.
17438	When using 'Update Remote Interface' from J2EE for a Bean class, the corresponding method interface is created in the remote interface, however the documentation and exception properties are missing.
17684	After a method name in an abstract class is changed ,the former method remains in a sub class after code generation.
17811	Once a container class has been defined using the custom Rose J specifications, it cannot be removed.
17925	The Java add-in uses the <install dir>\Rose\java folder to store the temporary Code.xml file. If the user running Rose does not have administrator privileges, this will fail on Windows 2000.
18043	A modal dialog from the Java and CORBA Project Specification goes behind Rose and still holds focus. ▪ When you enter a non-existent path in the **Classpath** portion of the dialog, you are prompted to createthe directory. If you click on something else, the prompt goes behind Rose. ▪ On the PC, the **Classpath** prompt appears on the toolbar and can be brought back to the front by selecting it there.
18591	On reverse engineering Rose will incorrectly report a syntax error when an opening brace '{' is followed by an opening C style comment '/*'
21178	When using IBM VAJ, if the IDE crashes, Rose will be unable to auto-start the IDE. **Workaround**: Manually start the IDE.
21203	Rose does not generate default return statements for certain J2EE methods.
23948	Rose on Linux x86 may hang when creating a jar or war file. If this occurs contact Rational Technical Support.

Defect	**Description**
31104	Changing the return type of an operation generates an error, rather than generating code.

Rose Ada Add-In

7

The Rose Ada add-in allows you to produce Ada source code from the information contained in a Rose model.

The code generated for each selected model component is a function of that component’s specification and code-generation properties, and the model’s properties. These properties provide the language-specific information required to map your model onto Ada.

Rose / Ada preserves user-supplied declarations and statements from one iteration to the next. You place such user-supplied code in protected code regions which are preserved whenever code is re-generated.

The Rose / Ada Code Generator:

- Substantially reduces the elapsed time between design and execution
- Produces uniformly-structured source code files, promoting consistent coding and commenting styles with minimal typing
- Supports both Ada 83 and Ada 95

Rose Ada Add-In New Features and Enhancements

There are no new Rose Ada features or enhancements for this version release of Rose.

Rose Ada Fixes With No Defect Associated

Rose SGI: Remove dependency on an OS call, the CodeMapper interpreter is returning a spurious status code. The OS call was for performing a file compare to determine whether an update was necessary.

For a role, where RecordFieldImplementation is set to Discriminant, the container declaration occurred after the type definition.

Rose Ada Defects Fixed in This Release

Defect	**Description**
3486	`preserve=yes` as default when generating Ada bodies. A new operation property `InitialCodeBody`, specifies code that should be emitted in the user generated subprogram body. The first time the code is emitted, the protected region is set with `preserve=yes`. The code will be preserved on subsequent code regenerations, until `preserve=no` is set, by the user. The name default refers to the value of the project property `DefaultCodeBody`. The default setting is `${default}`.
11093	Rose Ada should allow [statement] to be overridden. A new project property -- `DefaultCodeBody`, specifies what code should be emitted in generated subprogram bodies. The default setting is [`statement`].
11737	Ada 83 and Ada 95 -- A new value `RenamingAsBody` has been added to the `SubprogramImplementation` property. This effectively replaces `Renaming`, which is now used to generate a renaming-as-specification. **Note:** You must explicitly change any earlier setting of `Renaming` to `RenamingAsBody`, if renaming-as-specification is not desired.
12329	Code generation ignores the `Abstract` field for an operation.
12687	Formal parameters are not substituted for unconstrained parameterized implementation.
13429	Private inheritance defects
13504	`Decl` dependency not handled correctly
14965	Rose Apex needlessly reports circularities among nested packages.
15393	**Type Name** is not substituted in an association with an instantiated class.
15404	Code generated for inherited discriminants does not comply with the Ada 95 LRM
15791	Rose Ada 83 does not generate code if the class name contains the colon (:) character.
16141	Only some of the `ContainerDeclarations` are generated.
16416	Default implementation of equality recurses indefinitely.
16667	Illegal code is generated for bidirectional associations that has a `decl` relationship.
17059	Ada code generation objects to circular dependencies, even when legal.

Defect	**Description**
21639	Apex subsystem views are created incorrectly.
21774	Ada 83 - Visibility of static attributes is not working.
21919	Ada 83 code generation --The order of operation in the specification does not match the order of operations in the body.
21923	Ada 83 code generation - Protected operations were previously placed in the visible part of the package. Protected operations are now placed in the protected part of the package, just as they are in Ada 95 code generation.
21924	Ada 83 code generation - Class utility operations and attributes are incorrectly generated.
23596	Ada 95 code generation cannot generate a package body of a class utility, even when the operations are defined.
24880	Accessor functions are not generated for abstract classes

Rose Web Publisher Add-In

8

The Rational Rose Web Publisher Add-In allows you to publish Rational Rose models in HTML format. The Web Publisher Add-In exports your Rose model to HTML (including diagrams) so you can view them using a standard browser such as Netscape Communicator. The Web Publisher also produces meta tags for use by search engines.

You can find Web Publisher Add-In release note information at the following locations:

- *New Web Publisher Add-In Features and Enhancements* on page 63
- *Browser Data* on page 63

New Web Publisher Add-In Features and Enhancements

There are no new features or enhancements for this version of Rose Web Publisher.

Browser Data

In order for Netscape to find the correct Java jar files, you must have the browser on your PATH, or set the environment variable BROWSER_PATH to the absolute path of your browser.

The Web Publisher on UNIX supports only .jpg file format.

The Web Publisher has only been tested with Netscape versions 4.6 and 4.7*.

XML DTD Add-In

9

The Rational Rose XML DTD Add-in provides visualization, modeling, and code generation to create XML document type definitions (DTD) for use in your application. You can reverse engineer an existing XML DTD to visualize a DTD's structure using a Rose class diagram. After modeling an XML DTD, you can use the Syntax Checker to validate your XML DTD. From the valid XML DTD model, you generate code to create a new XML DTD.

In addition to reverse engineering an existing XML DTD, you can model a DTD using Rose class stereotypes for XML elements, element attributes, entities, and notations. By modeling an XML DTD, you can visualize the structure of the document to see which element definitions to change or remove. You can also share your XML DTD model with members of your development team to verify that you have captured the XML document requirements.

An XML DTD syntax checker finds errors in the XML DTD model, allowing you to make corrections before the DTD before is implemented. Use the Generate Code feature to produce an XML DTD that you can test or implement in your environment.

To support mapping XML to UML, Rational Rose extends UML with stereotypes for XML elements, element attribute lists, entities, and notations. Stereotypes also represent XML operator symbols, sequence lists, choice lists, and element and element attribute multiplicity.

Note: DTDElementANY cannot be nested. Use this DTDElement at the root level.

Known XML DTD Add-In Problems and Limitations

The following table lists known XML DTD problems or limitations in Rose for UNIX/Linux.

Table 7 Known XML DTD Add-In Limitations

Defect	Description
11382	File extension case sensitive on UNIX after generating code. Reverse engineering and then generating code on *.DTD file, does not generate code to existing *.DTD but creates a new *.dtd file (case sensitive) on UNIX.
31719	**Apply** tab remains unabled.

Fixed XML DTD Add-In Defects

The following table list defects that have been fixed in this release.

Table 8 Fixed XML DTD Add-In Defects

Defect	Description
11524	When an XML DTD is reverse engineered and then forward engineered, the order of the data may be changed.
15062	Two copies of the R2Editor will start if browsing source when multiple classes are selected. This occurs from the XML and CORBA Add-In's.

Additional Add-Ins

10

This chapter contains information on the following add-ins:

Rose Model Integrator Add-In

The Rose Model Integrator Add-In lets you compare model elements from up to seven contributor files, discover their differences, and merge them into the base model. There are no new features for this release of Rose Model Integrator.

Known Model Integrator Add-In Problems and Limitations

The following table contains known Model Integrator problems and limitations in Rose for UNIX/Linux.

Table 9 Known Model Integrator Add-In Limitations

Defect	Description
1236 11454	Model Integrator does not support ClearCase elements. Model Integrator does support ClearCase versions for primary units specified on the command line or in the Contributors dialog, but it does not correctly parse subunit file names to create version-enhanced file names used by ClearCase. The workarounds are: 1. Merge subunit files individually using ClearCase. 2. Create separate views in ClearCase which contain all the proper versions of your model's subunits and then merge the models independently of ClearCase (start Model Integrator in the normal way and enter the model file names in the Contributors dialog). Check the resulting merged model back into ClearCase when you are finished. You will need to create 3 views: the base view containing the previously checked in version, the latest version, and the version you want to merge.
1614	Project specs in Java and CORBA. Directories specified for code generation may not be merged properly into the merged model. The workaround is to reenter these directories in Rose.
2169	Model Integrator allows user to select more than 7 contributors. Model Integrator only allows 7 contributors, but it allows the user to select an 8th contributor and then issues an error message.
7469	Model Integrator creates duplicate model elements. In a situation where one contributor has renamed a model element to new name and another contributor has added the same new name to the model without renaming the old model element, Model Integrator may produce a model with two model elements having the same name. The workaround for this is that when the model is loaded into Rose, Rose will report an error and rename one of the model elements automatically. Visit these elements in the Rose browser window and correct the names of the elements.
18159	In the Contributors dialog, the checkbox for the option "Compare/Merge against Base Model" should be grayed out until two contributors have been added.
25249	**Select All** does not select the main node in the tree view.
31427	"+" and "-" icons not working correctly in the browser.

ClearCase Add-In

Note: Prior to using the Rose for UNIX/Linux ClearCase Add-In, it is necessary to review and follow the steps outlined in the Add-Ins for Version Control help. To view the help, select the Rose **Help > Contents and Index** menu item, and then open the Add-Ins for Version Control help book.

The ClearCase Add-In provides a tight integration between Rational Rose and the Rational ClearCase version control system. This add-in is intended as a replacement for the ClearCase capabilities provided through the previous Rose/ClearCase integration. This has and will continue to allow Rational to provide more ClearCase specific customization to this add-in.

Known ClearCase Add-In Problems and Limitations

The following table contains known ClearCase problems and limitations in Rose for UNIX/Linux.

Table 10 Known ClearCase Add-In Limitations

Defect	Description
10530	If you select a unit whose parent is read only and try to run the uncheckout command on it, the ClearCase add-in warns that the parent unit is read only and does not uncheckout the unit. Workaround: Make the parent writable by checking it out.
15035	When using Rose/ClearCase Integration in Rose for UNIX/Linux, ClearCase output will appear in the terminal used to run Rose in addition to the ClearCase dialogs.
18738	After creating a branch and placing focus on Rose, there is no longer a dialog stating that the "...model has been externally modified...do you want to reload". The current correct behavior is to reload the model immediately.

Fixed ClearCase Defects

The following table lists ClearCase Add-In defects fixed since the last release.

Table 11 Fixed ClearCase Add-In Defects

Defect	Description
1240	Rose adds a model to version control without any error messages indicating that the model is write protected.

Table 11 Fixed ClearCase Add-In Defects (continued)

Defect	Description
1241	In general, icons in the browser should accurately reflect the ClearCase status for the associated unit. If, however, the Icons in the browser do not reflect the current ClearCase status (as might occur if the state is changed from outside of Rose), you can force a refresh of the browser icons by unloading and then reloading the ClearCase Add-In from the Add-In Manager.
1242	Removing from version control will fail if the parent unit is checked-in. Workaround: Before removing an element from version control, be sure to check-out the parent unit.
1243	Performing an undo checkout on a parent unit will cause the children to become unloaded. It will be necessary to reload the child units manually if desired. You should not select both a parent and child unit when using the undo checkout command as this will cause unexpected behavior when reloading the units. If this operation is performed inadvertently, you should reload the model and sub units afterward. It is recommended that you not use multi-selection with the undo checkout command.
1244	At present, the ClearCase Add-In does not fully support snapshot views. If you are using snapshot views, you may experience problems with the browser icons not being updated to reflect the current CM state. You can correct this problem by unloading and reloading the ClearCase Add-In from the Add-In Manager. In addition, you should not use the Get Latest menu item to update your snapshot view. Instead, you should update the snapshot view using one of the external ClearCase mechanisms (such as the ClearCase menu items in the Windows Explorer). These problems should be resolved in a future release. IMPORTANT NOTE FOR USING ROSE WITH CLEARCASE. In order to effectively control and use Rose models/units with ClearCase, you must set a Path Map symbol to the ClearCase view in which you are working. Otherwise, the full path to the units will be saved in the model making them unusable from any other ClearCase view. For example, if you are controlling Rose units in the view X:\MyView, you should set a Path Map variable, such as “CCVIEW”, to the value “X:\MyView” using the **File >Edit Path Map...** dialog. This will ensure that when the model and sub units are saved, references to the views path will be replaced with $CCVIEW. If you wish to work with the model from another view, such as “Y:\MyOtherView”, just bring up the **File >Edit Path Map...** dialog again and change the CCVIEW variable to the new view and reload the model.

Table 11 Fixed ClearCase Add-In Defects (continued)

Defect	Description
1596	Rose adds a package to Version Control despite of error message indicating that the .cat file is write protected.
1598	After creating a new VOB and a dynamic view, if you try to add any object to it, the version control (using the ClearCase Add-In) will fail. This happens only if no operations have ever been performed on the VOB.
5661	Browse button does not work in Select Activity dialog box.
10530	If you select a unit whose parent is read only and try to run the uncheckout command on it, the ClearCase add-in warns that the parent unit is read only and does not uncheckout the unit.
12982	Activation of ClearCase Add-In may cause delay when refreshing (redrawing) the browser. If this occurs, contact Rational Technical Support.
12983	ClearCase operations such as checkin and checkout may force reloading of units which have not changed. If this occurs, contact Rational Technical Support.
16244	ClearCase operations done outside of Rose on an open Rose model's units do not get reflected in Rose's browser if this ClearCase operation does not affect the content of the file unit nor its read / only status. The workaround is to reload the unit when this happens. It is highly suggested to do ClearCase operations from within the Rose ClearCase add-in to avoid these types of problem.
17559	Overlay not added to package correctly in browser when performing a ClearCase check-in.

DDL Add-In

The Data Definition Language (DDL) Add-In release note contains known problems and limitations as well as fixed defects since the last release.

The gap between object technology and relational databases is closed by providing a mapping interface to relational databases through generation of persistent classes to Structured Query Language (SQL) and Data Definition Language (DDL).

You can find DDL Add-In release note information in the following locations:

- Known DDL Add-In Problems and Limitations

Known DDL Add-In Problems and Limitations

There are no known problems and limitations for the DDL Add-In.

CORBA Add-In

Rational Rose CORBA Add-In allows you to forward engineer Rose model elements into CORBA-compliant IDL code and it allows you to reverse engineer CORBA IDL code into appropriate Rose model elements.

New CORBA Add-In Features and Enhancements

In support of the CORBA 2.3 specification, Rose CORBA now implements the use of value types. Value types allow you to pass objects by value rather than by reference, which is particularly useful when an object's primary purpose is to encapsulate data, or when an you want to explicitly make a copy of an object. Two new CORBA stereotypes make use of this enhancement:

- CORBAValue
- CORBAFixed

These stereotypes are not yet documented in the online help.

Rose CORBA continues to use the R2Editor, which provides a user-friendly environment for code generation and updating.

Known CORBA Add-In Problems and Limitations

The following table contains known CORBA Add-In problems and limitations:

Table 12 Known CORBA Add-In Limitations

Defect	Description
1770	You can create a nested class in CORBA typedef. You are not supposed to use nested classes in a typedef. This is described in the online documentation for CORBA.
1771	Reverse Engineering reports parse error on comment before }; CORBA reverse engineering reports an error when a comment is added to the last attribute of a struct.
1773	Some Include statements are generated incorrectly.

Table 12 Known CORBA Add-In Limitations (continued)

3881	Rose may hang if a Rose J or CORBA operation and attribute specification dialog box is viewed while the browser is undocked. Rose is unable to gain focus, once the specification dialog is closed.
6478	CORBA: interface generated before the typedefs it depends on.
9664	Syntax Checker should check to see whether a valid modifier is selected before reverse engineering idl.
10922	Unable to change font color in CORBA source editor.
11721	Unable to create directory over NFS in Java or CORBA Project Specification. Unable to create directory on an NFS mounted disk when defining source directory using Java or CORBA project specification.
16853	Browse source <ctrl+H> or <ctrl+K> shortcut key on CORBA classes fails. It will be necessary to use the **CORBA > Browse Code** menu option.

Fixed CORBA Add-In Defects

The following table shows the CORBA Add-In defects that have been fixed since the last release.

Table 13 Fixed CORBA Add-In Defects

Defect	Description
1778	Parser error while trying to reverse engineer CORBA when there is a comment after brace following enum.
3217	Reverse engineering #include with ".." causes incorrect component to be generated.

Framework Add-In

The Framework Add-In provides a library of frameworks that can be used as templates when creating new models. There are no new features or enhancements for this version of the Framework Add-In.

Apex Add-In (Not available for Linux)

The Apex Add-In of Rose for UNIX/Linux provides direct access to the Compile and Control commands in Apex. It also works with language add-ins for Rose, including Rose C++, Ada 95, and Ada 83 Add-Ins, to create Apex subsystems and views during the code generation from the model to the source files of the specified language.

- It is required to run Rose inside an Apex shell window to use the Apex Add-Ins.
- If the **Apex** submenu is not in the **Tools** menu, select **Add-Ins > Add-In Manager** to make sure the Apex Add-In is activated.

The following information is also located in your Apex Add-In online documentation.

Apex Compile and Control Commands

The Compile and Control commands of Apex are directly accessible from Rose through the **Tools > Apex** menu. The Compile commands compile and link the source files, generated into Apex views via Apex Add-In, that correspond to the selected items in the diagram. The Control commands perform Apex source code control operations on the source files generated in the same way.

To use these commands, select **Tools > Apex**, and then select Compile or Control to get to the list of commands, respectively.

For the descriptions on these Apex commands, please consult the online help in the Apex windows.

Code Generation With Apex Add-In

In general, the code generation operation for a regular language Add-In is based on a directory structure where top-level packages, nested packages, and classes (or modules) are mapped, respectively, into directories, subdirectories, and source files in a hierarchical way. The operation can create such directories or subdirectories if they do not already exist, and then generate source files into these (sub)directories. This works with the regular source code control system and build management.

Apex has a different style of source code control and build management. An Apex project has a set of subsystems which contain the source files and subdirectories of source files for development. Each developer has his own views, one for each subsystem, which collectively reflect the source base the developer uses for the project. For this reason, Apex-style code generation is different from the regular style. From a Rose model, the top level packages will be mapped into subsystems and the associated views for the developers; the packages nested inside another package will be mapped into the subdirectories inside an Apex view, or a subdirectory already enclosed by a view; the classes (or modules) will be mapped into the source files inside the views or the subdirectories.

The Apex Add-In provides a mechanism for the user to select which of the two code generation styles to use. When the property CreateApexSubsystemAndView of the Apex tool set is set to “Yes”, the code generation command for a language add-in will automatically create Apex subsystems and views for the top-level packages selected in the model, followed by the generation of subdirectories and source files into the views. If the property is not set to “Yes”, the code generation command does the regular style of code generation.

The language add-ins that work with Apex Add-In include C++, Ada 95, and Ada 83.

Properties for Apex Add-In

Refer to the online documentation for a list of the properties used by the Apex Add-In for Apex-style code generation. Some are properties for the project, and the others for the subsystems in the model. The code generation command will use the combination of these properties when it needs to generate code into Apex subsystems and views.

All the properties except CreateApexSubsystemAndView will take effect only when the CreateApexSubsystemAndView property is set to “Yes” and when the selected item is a top-level package. When any of these properties are specified differently from their default value, the Directory property for the top-level package should remain as the default value.

Utilizing Apex Add-In on HP-UX and SGI IRIX

In order for Apex to generate subsystems correctly, it is necessary for “tmp-mnt” to be placed in the automount field of the Virtual Pathmap dialog box. The “/tmp-mnt” entry should be removed from the Virtual Pathmap dialog box when not using Apex.

Utilizing Apex Add-In on AIX

When using the Apex Add-In to create Apex subsystems and views on AIX, Rose for UNIX/Linux might run into a hanging state. If that happens, look for an Apex “Job Queue” window. In that window, select all the entries and click **File > Run Job**. This will resolve the “hanging” state and resume code generation.

Known Apex Add-In Problems and Limitations

The following table shows known defects in the Apex Add-In.

Table 14 Known Apex Add-In Limitations

Defect	Description
3853	CM commands do not show on the context menu for packages in the browser. When this occurs, you are unable to use Apex CM integration on a package that does not appear in the diagram. Workaround: Add the package to the diagram.

Fixed Apex Add-In Defects

The following table shows the Apex Add-In defects that have been fixed since the last release.

Table 15 Fixed Apex Add-In Defects

Defect	Description
2967	Error: “..failed to exec the child process reverse_engineer; No such file or directory .." The above error may occur when attempting to reverse engineer Rose / Ada code from Apex, running: **Rose > Ada_Reverse Engineer** Workaround: For Rose, the Ada reverse-engineering code is done on the Apex side. (It is necessary to use Apex version 3.0.0b or greater.)