Pathfinder Solutions

PathMATE Modeler's Guide

Model Driven Architecture with Rational Rose

Version 1.7

December 31, 2004

PathMATE™ Series

Pathfinder Solutions LLC
33 Commercial Drive, Suite
Foxboro, MA 02035 USA

www.PathfinderMDA.com

508-543-7222

Table of Contents

Preface

Audience

The *PathMATE Modeler's Guide* is for software engineers who want to use PathMATE to create high performance systems. Users of this guide should be familiar with the Unified Modeling Language (UML).

Related Documents

These PathMATE documents are available at www.PathfinderMDA.com, or from your Pathfinder account manager:

- *PathMATE Transformation Engine User Guide*
- *Platform-Independent Action Language*
- *PathMATE Quick Start Guide*

Conventions

The *PathMATE Modeler's Guide* uses these conventions:

- **Bold** is for clickable buttons and menu selections.
- *Italics* is for screen text, path and file names, and other text that needs special emphasis.
- `Courier` denotes code, or text in a log or a batch file.
- A **NOTE** contains important information, or a tip that saves you time.

How to Use this Guide

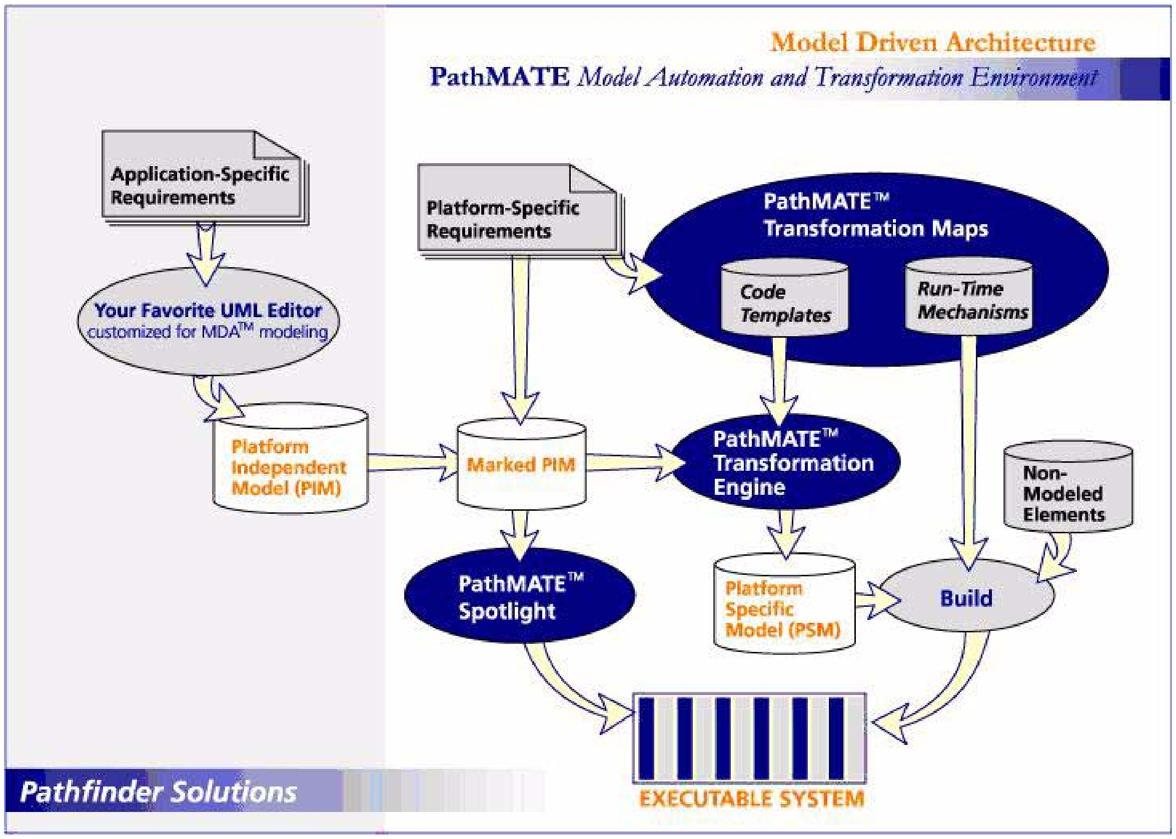
If you are not familiar with the PathMATE toolset, read the overview that begins on page iv. If you have not installed the PathMATE toolset on your computer, obtain a password from your account manager and download the software from www.PathfinderMDA.com. After installation, use the *PathMATE Quick Start Guide* to become familiar with the software tools.

PathMATE Overview

This overview introduces Model Driven Architecture (MDA) and the PathMATE™ tools that make MDA work. MDA and PathMATE move you from writing and debugging code to developing and testing the logic of a high performance system. Over years of rigorous refinement in several industries, PathMATE tools have proven their value in rapid and effective software systems development.

PathMATE Toolset

The PathMATE Model Automation and Transformation Environment includes all the tools required to transform your MDA models into high-performance systems. See the PathMATE workflow in the figure below.


Model Driven Architecture
PathMATE Model Automation and Transformation Environment
Application-Specific Requirements
Your Favorite UML Editor customized for MDA™ modeling
Platform Independent Model (PIM)
Platform-Specific Requirements
Marked PIM
PathMATE™ Transformation Maps
Code Templates
Run-Time Mechanisms
PathMATE™ Transformation Engine
Non-Modeled Elements
PathMATE™ Spotlight
Platform Specific Model (PSM)
Build
EXECUTABLE SYSTEM
Pathfinder Solutions

PathMATE Workflow

The three parts of the PathMATE toolset cooperate to turn your models into executable systems:

- *Transformation Maps* – Generate C, C++, or Java software with off-the-shelf Transformation Maps, or create custom maps to drive output for other languages or specific platforms.
- *Transformation Engine* – The Engine transforms platform-independent models into working, embedded software applications.
- *Spotlight* – Verify and debug your application logic with Spotlight, the most advanced model testing environment available.

No other MDA transformation environment offers a more open or configurable set of development tools, designed to meet the requirements of systems engineers.

How PathMATE Works

Use Model Driven Architecture to build complex embedded systems that meet rigorous standards for speed and reliability. MDA works because it separates what the system does from its deployment on a particular platform. PathMATE adds these advantages:

- *Greatest architectural control* – A highly configurable Transformation Engine enables you to optimize output for resource-constrained platforms.
- *Clean separation of model and code* – Conforming to the MDA paradigm, PathMATE models contain no implementation code. That gives you fast and flexible deployment and migration capabilities.
- *Configurable, target-based model execution and testing* – Preemptively eliminate platform-specific bugs, minimize quality assurance resources, and accelerate development.
- *Lowest cost of ownership* – Integrate PathMATE with your existing UML editor. Build on your previous investment in training and software.
- *Speed* – Even large transformations take just seconds with PathMATE. That enables highly iterative model development, and rapid transformation and test cycles.

Try the demonstration software available at *www.PathfinderMDA.com* to get started quickly and easily.

1. Introduction

This guide explains how to use Rational Rose to create UML™ models that you can export to the PathMATE Transformation Engine. The Engine transforms the semantic information captured in the models into formatted documentation, HTML, XML, compilable source code, or any other ASCII form.

The following sections take you step by step from updating your Rose installation with the *PathMATE for Rose* add-in, through creating and maintaining a Platform Independent Model. This guide assumes that you are familiar with *Model Based Software Engineering: Rigorous Software Development with Domain Modeling*, available from www.PathfinderMDA.com.

In order to gain full benefit from the PathMATE toolset, you should become fluent with PathMATE's MDA process. This process achieves the goals of the OMG's Model Driven Architecture (MDA) initiative with a set of proven techniques supported by powerful and flexible technology. Pathfinder Solutions offers training in MDA and domain modeling for practitioners, and can provide highly experienced consultants to ensure your success.

2. General Procedures

This section contains general procedures for the following operations:

- Product installation
- Model conversion
- Model creation

Product Installation

Before you attempt to capture PathMATE models with Rose, install Pathfinder's *PathMATE for Rose*.

If you upgrade from Pathfinder's *UML Essentials* product, please see important information about how to convert models in the section below.

Set the Rational Rose CURDIR Symbol

The sample models and models that you create using Rose .cat files require that you define the CURDIR (current directory) symbol. To define the symbol:

1. Start Rational Rose.
2. Click **File > Edit Path Map...** in the top menu bar.
3. The Virtual Path Map box opens.
4. Enter *CURDIR* in the *Symbol* field.
5. Enter *&* in the *Actual Path* field and click **Add**.
6. The new mapping appears under *Virtual Symbol to Actual Path Mapping*.
7. Click Close to close the box.

NOTE

Not having this symbol set properly may cause PathMATE sample models to load incorrectly.

Text Editor for PAL and Type Files

The Rose menu items open text files with the .pal and .typ extension that define the action language and user defined types. By default, these files open in Notepad. The procedures below explain how to associate your favorite text editor with the .typ and .pal extensions in the common Windows operating systems.

Windows 98 and NT

In Windows Explorer, select **View > Options > File Types** in the top menu bar.

1. Click **New Type**.
2. In the New Type dialog, enter the file extension PAL and under the action box click **New**.
3. In the New Action dialog enter the action Open and specify the full path name of the editor program you would like to use for editing action text files.

NOTE

If you are not sure what to enter, find an existing extension that opens with your favorite text editor and use the same settings.

4. Repeat directions above with the TYP extension.
5. Click **OK** to close each open Explorer dialog box.

Windows 2000

In Windows Explorer, select **Tools > Folder Options > File Types**.

1. Click **New**.
2. In the Create New Extension dialog, enter the extension PAL.
3. Click the **OK** button to close the New Extension dialog.
4. Select the PAL extension in the Registered File Types list box.
5. Click the **Change…** button. Alternately, you may select the **Advanced** button if the selected application requires DDE settings.
6. In the Open With dialog, select the application that you would like to use to open PAL files.
7. Click **OK** to close the Open With dialog.
8. Repeat directions above for the TYP extension.
9. Click **Close** to close the Folder Options dialog.

Model Conversion

If you created models using *UML Essentials for Rose* version 4.03.020 or earlier, you will need to convert your models to PathMATE. Your models are converted automatically when you generate code or generate reports, or when you run the rose_convert program. On conversion, the following changes take effect:

- The PathMATE tab replaces the MBSE tab in all Specification boxes.
- Properties specified on the MBSE tab are copied to the PathMATE tab.
- Identifier attributes are indicated by setting the Identifier stereotype rather than setting the Identifier property on the MBSE tab to True. The PathMATE tab does not have an Identifier property.
- The MBSE Event stereotype on Operations is renamed to Event.

NOTE

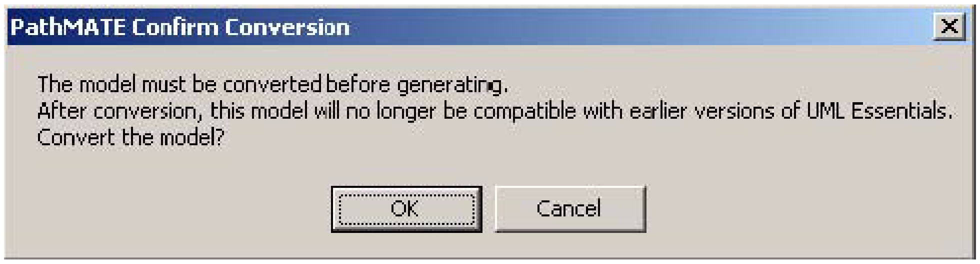
Models converted to PathMATE are not compatible with UML Essentials releases.

Preparing Models for Conversion

Locate the models that require conversion. Make sure that the .mdl file and any .cat files that the model references are writable. If you are using a configuration management system, check out the models that require conversion. If the model file or one of its subunits is not writable, conversion reports an error.

Conversion during Generation

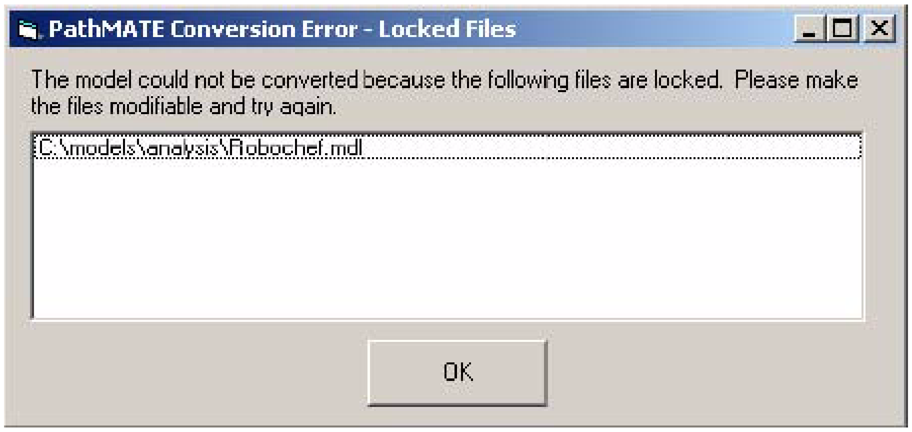
To convert the model by generating, open the model file in Rose. Select any of the **Create > PathMATE Generate** menu items. The PathMATE Confirm Conversion dialog appears:



The model must be converted before generating.
After conversion, this model will no longer be compatible with earlier versions of UML Essentials.
Convert the model?

Click **OK** to start the conversion. If you click **Cancel**, the model will not be converted and generate will fail.

If the model file or any of its subunits are read-only, the PathMATE Conversion Error dialog appears:



The model could not be converted because the following files are locked. Please make the files modifiable and try again.

C:\models\analysis\Robochef.mdl

Click **OK**. Generation will fail. Using the Windows File Explorer or your configuration management tool, unlock the listed files. For your convenience, a full list of the locked files may be found in the conversion log file. The log file is located in the same directory as the model and is called *<model_name>_convert.txt*.

If the conversion succeeds, the model generation process will begin. The conversion utility will automatically save the model.

Converting with rose_convert Program

If you have a number of models to convert, use the rose_convert program. From the DOS prompt or from a batch file use the following command line:

```
start /wait rose_convert <model path>.mdl
```

The */u* option to rose_convert will automatically clear the read-only attribute on the .mdl file and any subunit that it references. If you use a configuration management system that requires you to check out files, check out the model file and any subunits that it references.

After running the rose_convert program, check the log files for errors. Errors in the command line options and PathMATE licensing will be reported in the *PathMATERoseConvert.log* file located in the directory from which rose_convert was run. Errors detected while converting the model will be reported to a log file called *<model_name>_convert.txt*, located in the same directory as the .mdl file.

Model Creation

Rose stores all diagram, data dictionary, and description information for a model in a single file with a .mdl extension. Domains and subsystems may be stored separately from the .mdl file in a .cat file. See the section on the domain repository for more information about how to use .cat files.

All the tools you need for analysis entry in Rational Rose become available when you install the PathMATE for Rose add-in. You must create a model and set it up properly before analysis entry can begin.

Files and Subdirectories

The PathMATE product works with files in addition to the Rose model file. The directory containing the Rose model file is considered the *target directory.* Relative to the *target directory*, Pathfinder add-in menu items will create and use the following files and subdirectories:

- `.\diagrams` – transforming a Rose model creates a .wmf file for each Analysis diagram for later use with formatted document generation, and places it in this subdirectory.
- `.\operations` – **PathMATE Open Action** creates and opens class operation action language files in this subdirectory.
- `.\state_actions` – **PathMATE Open Action** creates and opens state action language files in this subdirectory.
- `.\types` – **PathMATE Open Types** creates and opens user defined type declaration files in this subdirectory.
- `..\project\cpp\gencpp.bat` – **PathMATE Generate > C++** creates gencpp.bat the first time you invoke the command. **PathMATE Generate > C++** also runs gencpp.bat to generate compilable C++ code.
- `..\project\java\genjava.bat` – **PathMATE Generate > Java** creates genjava.bat the first time you invoke the command. **PathMATE Generate > Java** also runs genjava.bat to generate compilable Java code.
- `..\reports\genreports.bat` – **PathMATE Generate > Report** creates genreports.bat the first time you invoke the command. **PathMATE Generate > Report** also runs genreports.bat to generate model-specific documentation.
- `<system name>.xml` – This file is XMI data for your model, and is created/updated whenever you run a **PathMATE Generate** sub-menu item. It is the Analysis import vehicle for the PathMATE Engine. This file also contains a copy of each action file, so even if the only change is in an action file, you must still run a **PathMATE Generate** sub-menu item.

Once users are comfortable with the Rose environment, they are encouraged to integrate their own menu items to generate other reports, code, HTML, or anything else to suit their needs and preferences.

Diagram Documentation Blocks

We recommend that all diagrams be annotated with a text note that contains basic information, such as author and version information.

Analysis Element Names

Use legal action language identifiers when naming analysis elements such as classes, association roles, operations, attributes, and states. A legal action language identifier is a letter followed by a letter, a number, or a period.

3. Modeling

Section 3 on modeling covers the following topics:

- Domain modeling
- Scenario modeling
- Class modeling
- State modeling
- Action modeling
- User defined types
- Constants

Domain Modeling

To start a new domain chart:

1. In the Rose browser on a new model, select the package to be used as the root package for the system. Logical View is the suggested package to be used as the root package.
2. If the root package is to be a package other than Logical View, right-click on the package and select **Open Specification**. In the stereotype field enter *system*.
3. In the Rose browser, expand the root system package.
4. Rename Main with the same name as the system. This diagram is now the Domain Chart.
5. Add domains by adding UML packages. All UML packages added to the system package will be regarded as a domain and must follow the rules defined for a domain.

To enter domain information:

- *Open:* For each new domain, select its symbol, right-click, and select Open Specification in the pop-up menu.
- *Description:* Be sure to enter a domain description in the Documentation field on the General tab.
- *Prefix:* Enter a Prefix on the PathMATE tab – select the Prefix line, then left click in the Value column on this line. Enter the domain prefix in the provided sub-window, and click Edit Set. If you do not set a prefix, the domain prefix will be the same as the domain name.
- *Analyzed:* Set the Analyzed property on the PathMATE tab to False for all domains that are realized. When the Analyzed property is set to false, all classes in this domain except the Services class representing the domain interface will be ignored when extracting to XMI. In addition all action language is ignored when Analyzed is false.
- *Other:* You may ignore the Detail and Files tabs.

Domain Symbols and Requirements Flow

Place the most abstract domains at the top of the domain chart. Use the dependency arrow to show the flow of requirements from higher-level domains to lower-level domains. Select the Dependency arrow symbol from the palette, left-click in the upper domain, then drag the dependency into the lower domain.

NOTE

It may appear on the diagram that the dependency arrow is not connected to the lower level or server domain. This is only a diagram presentation issue and will not cause any code or report generation problems.

Class Diagram

Double-click a domain's package symbol in the domain chart to open its class diagram – this is initially called Main – rename it to the domain name.

Domain Services

To define the domain services available for a domain:

1. Open the Class diagram for the domain.
2. Create a new class in this class diagram.
3. Rename it in the browser with the name *Services* to avoid dragging in a services class from another domain.
4. Specify the prefix *interface* for this class.

5. Add domain services as operations of this class.
6. To edit the action language file for an operation, select the operation in the Rose browser, right-click, and select **PathMATE Open Action**.

   See *Creation of Action Language Files* on page 20 for more information about how to create PAL files.

Add Subsystems

A domain may be partitioned into a number of subsystems. To add a subsystem, select the domain package in the browser. Right-click and select the **New > Package** menu item. A new package inside the domain will be added. When adding a class diagram for the subsystem, name the class diagram the same as the subsystem name.

When adding other supporting diagrams such as Use Case diagrams, Scenarios, and other Class Diagrams, name the support diagram with the prefix of the top level domain followed by a period, followed by any other text.

When using the Engine, the support diagrams located in the domain package will be in the list of support diagrams associated with the domain. Support diagrams located in the subsystem package will be in the list of support diagrams associated with the subsystem. Thus to get the entire list of support diagrams for the domain, any templates must traverse the list of child subsystems as well as the domain support diagrams.

A subsystem may be stored separately from the model in a .cat file for easier version control. To extract a .cat file, right-click on the subsystem in the browser and select **Units > Control** in the pop-up menu. By default, the **PathMATE Generate** and **PathMATE Open** menu items will look for the PAL files for state actions, transition actions, operations, and domain services in the support file directory for the domain. To locate the PAL files relative to the subsystem .cat file, open the specification for the subsystem and set the PALLocation property on the PathMATE tab to SubsystemRelative. The PALLocation property applies to the subsystem and all uncontrolled child subsystems.

Domain Repository

The analysis for each analyzed domain may be located entirely in the system-level model (.mdl file), or it may be separated by domain. To separate a domain, export the domain to a Rose category file (.cat). The system model file then has a link to the <DomainName>.cat file.

To extract a .cat file, right-click on the domain in the browser and select **Units > Control**. PathMATE generate and open menu items will look for the .pal and .typ files in a directory relative to the .cat file. If the .cat and .mdl files are in different directories, move the .pal and .typ files for the controlled domain from the directories relative to the model to the directories relative to the .cat file. To locate all the PAL files and type files for the controlled domain, search for <domain_prefix>*.pal and <domain_prefix>*.typ using Windows Explorer. Alternately, open the package specification and set the PALLocation property on the PathMATE tab to SystemRelative.

Scenario Modeling

Scenario Modeling is a powerful technique, either at the system level where you create the interactions among domains, or at the domain level where you create the interactions among the classes in the domain.

Scenario models are represented in UML by either sequence or collaboration diagrams. The instructions that follow are for creating sequence diagrams. To create a corresponding collaboration diagram in Rose, press the F5 key.

Create a Sequence Diagram

1. Select the system package (Logical View) or a specific domain in the browser.
2. Right-click and select **New > Sequence Diagram**.
3. Name the diagram either:

   • <system name>.<scenario name> for system-level scenarios (under system package), or

   • <domain prefix>.<scenario name> for domain-level scenarios where <domain prefix> is the prefix of the parent domain package under the system package.
4. Open the scenario diagram.

Alternatively, system scenario diagrams may be added to packages which are not in the system package.

- Select the package containing the diagrams. The diagrams may be nested in a package in the selected package.
- Name the diagram <system name>.<scenario name>
- Right-click on the package and select Open Specification. On the PathMATE tab, set the SupplementalDiagrams property to True.

Describe the Scenario

Add a Note to capture the detailed Preconditions for the scenario, another Note for expected Postconditions, and a third note for a general Description.

Add Domain Entity

1. Place an Object on the chart.
2. Select the Object, right-click and select Open Specification.
3. Enter the domain's prefix as the Name**.**
4. Select the domain's Services class from the Class pick list.

NOTE

This action requires having created a services class on the domain's class diagram.

Add Class Entity

1. Place an Object on the chart.
2. Select the Object, right-click and select Open Specification.
3. Enter a unique instance Name **–** for example, if this class has an identifying attribute, provide the value of this attribute.
4. Select the appropriate class from the Class pick list.

Add Inter-Domain Interaction (System-Level Scenario)

1. Select Object Message from the palette.
2. Left**-**click in the originating lifeline.
3. Drag and Release in the receiving lifeline.
4. Select the newly placed interaction.
5. Right**-**click and select from the list of defined services or select <new operation> to create a new service.

Add Inter-Class Interaction (Domain-Level Scenario)

1. Select Object Message from the palette.
2. Left**-**click in the originating lifeline.
3. Drag and Release in the receiving lifeline.
4. Select the newly placed interaction.
5. Right**-**click and select from the list of defined operations, or select **Open Specification…** and in the Name field enter an event name, *Create*, *Delete*, or *call <service handle name>*.

Class Modeling

Follow these procedures to construct a class diagram.

Class Modeling for a Domain

Double-click on a domain's package symbol on the domain chart to open its class diagram. The class diagram is initially called Main. Or, create/open from the browser under its domain package.

Class Properties

When completing the definition for each class, specify:

1. Type a brief description of the class in the *Documentation* field of the General tab.
2. Specify a prefix in the PathMATE tab.
3. Ignore the Detail, Relations, Components, Nested, and Files tabs.

Specify Attributes

To specify attributes on the class properties Attributes tab:

1. Right-click in free space in the list window.
2. Select **Insert**.
3. Type the attribute name.
4. Right-click the new attribute
5. Select **Specification…**.
6. Specify: *Name*, *Type*, *Initial Value*, and *Documentation* in the General tab.
7. If necessary, select the Identifier stereotype on the General tab to specify that the attribute is an identifier.

   If an attribute is an identifier, some designs display the attribute value in the list of classes in the Spotlight debugger. The default setting is that the attribute is not an identifier.
8. Ignore the Detail tab.

Specify Class Operations

The procedure used to specify class operations on the class properties Operations tab is similar to the procedure used to specify attributes on the Attributes tab.

Add Association

1. Select the Unidirectional Association tool from the palette.
2. Left-click in one participant of the association and drag the association line to the other participant.
3. Right-select **Open Specification** or double-click the association line to open the Specification box.
4. Name the association using *A<number>*, where *<number>* is unique within the domain.
5. Provide role names as appropriate in Role A and Role B on the General tab.
6. Describe the association in the *Documentation* field on the General tab.
7. Specify participant multiplicity on the Role A Detail and Role B Detail tabs.
8. Ignore the Role A General and Role B tabs.

Subtype/Supertype (Inheritance)

For inheritance relationships, use the Generalization arrow:

1. Start by clicking in a subtype and dragging to the supertype.
2. Continue drawing generalization arrows by clicking in a subtype and dragging to the first *generalization arrow*.
3. Name all legs the same, using *S<number>*, where *<number>* is unique within the domain.
4. Describe the relationship in the *Documentation* field on the General tab.

Realization Relationship

For realization relationships, use the Realize arrow tool:

1. Start by clicking in the class that realizes the interface and dragging to the interface class.
2. Continue drawing realization arrows by clicking in the classes that realize the interface and dragging to the interface.

NOTE

Rose will not allow you to terminate a realization relationship at the first realization relationship arrow as you can do with Generalization arrows.

3. Name all legs the same, using *S<number>*, where *<number>* is unique within the domain.
4. Describe the relationship in the *Documentation* field on the General tab.

State Modeling

To create a new state model, select the class it will belong to, right-click, and select **Sub Diagrams > New Statechart Diagram**.

New State

When adding a state, place a state symbol on the diagram, and enter its name:

1. Select the symbol.
2. Double-click on the state, to open the Specifications dialog.
3. Describe the state in the Documentation field on the General tab.

Add action summaries on the Action tab:

1. Right-click on the list box in the Action tab.
2. Select **Insert**.
3. A new action will be inserted into the list.
4. Double-click on the new action to display the Action Specification.
5. Select OnEntry from the When combo box.
6. Select Action from the Type combo box.
7. Type a short description of the action in the Name field.
8. Click **OK** to add the new action.

Entry/Exit Actions

Open the action language file for the entry or exit action by:

1. Select the proper state.
2. Right-click in the diagram.
3. Select **PathMATE Open Entry Action** or **PathMATE Open Exit Action**.

   See *Creation of Action Language Files* on page 20 for more information about how to create PAL files.

Transition Actions

Open the action language file for a transition action by:

1. Select the transition line.
2. Right-click in the diagram.
3. Select **PathMATE Open Action**.

   See *Creation of Action Language Files* on page 20 for more information about how to create PAL files.

When you extract the model, the action in the PAL file overrides the action specified on the diagram. If you did not create a PAL file via **PathMATE Open Action**, extract uses the transition action on the diagram. To specify a transition action on the diagram, select the transition, open the specification, select the Detail tab, and set the Action field.

Guard Expressions

To specify a guard expression:

1. Select the transition line.
2. Open the Specification box.
3. Select the Detail tab.
4. Set the *Guard Condition* field.

Update All Actions on a Statechart

To update the entry, exit, and transition actions on a statechart with actions from PAL files:

1. Cancel all selections on the diagram.
2. Right-click in the diagram.
3. Select **PathMATE Update Actions**.

State entry/exit and transition actions updates the appropriate state diagram symbol with the lines in the corresponding PAL file using the following algorithm:

1. If no lines are marked, then all lines in the action will be included in the action summary.
2. Otherwise only marked lines will be included in the action summary.
3. Lines starting with //! will be excluded from the Action Summary.
4. Leading comment characters //# or // will be stripped from explicitly marked lines.
5. If a line is not explicitly marked, the line will appear as it is in the PAL file.

Lines of a PAL file may be marked as Action Summary lines by:

- Starting the line with a //# comment.
- Inclusion between //! STATE ACTION SUMMARY and //! END SUMMARY. If only the beginning comment is found, the marked region starts at the begin mark and extends to the end of the file. If only the end comment is found, the marked region begins at the start of the file and extends to the end marker.

When both forms of marking are used together, all lines marked by either manner are included in the action summary.

NOTE

Due to a Rose limitation, updated transition actions do not appear in the diagram until the model is saved, closed, and reopened.

State symbols will resize if the updated state action is larger than the previous state action. Update the state actions before doing any detailed layout. To lay out the diagram quickly, select **Format > Layout Diagram** in the Rose top menu bar.

Update Selected Actions

To update a selected set of entry, exit, or transition actions, select one or more states or one or more transitions. Right-click and select **PathMATE Update Action**. The instructions for importing all state and transition actions above describe the algorithm used to create the action summary from a PAL file.

NOTE

As above, updated transition actions do not appear in the diagram until you save, close, and reopen the model.

Ignored or Deferred Events

Capture event ignored and event deferred information via the Ignored and Deferred properties on the state PathMATE tab. Use a semi-colon separated list (no newlines).

Superstate

To create a superstate, simply place substates within it.

Define Event

Due to limitations in the Rose state modeler, events need to be defined as operations of the destination class:

1. Name the operation with the event name (leave off the destination class prefix and the colon).
2. Select *Event* as the operation’s Stereotype on the General tab.
3. Define event parameters as input parameters.
4. Use no return data type.

NOTE

See section below on updating event arguments for instructions on how to automatically update the event arguments displayed on the state model.

Use Event

Events are referenced from transition based on naming:

1. Add the proper transition to the state model.
2. Right-click on the transition and select **PathMATE Select Event Spec**.
3. Select the appropriate event from the list of defined events.

Update Event Arguments on State Model

To update the event arguments displayed on the state transition diagram:

1. Open the state model.
2. Right-click in empty space with no symbols selected.
3. Select the **PathMATE Update Event Args** menu option.

Open Event Specification

To open the specification for an event:

1. Select the transition line, not the event name, in the Statechart diagram that contains the event.
2. Right-click in the diagram.
3. Select **PathMATE Open Event Spec** in the pop-up menu. The tool opens the Specification box for the operation that has the same name as the event and the stereotype event. If no event is defined, the tool creates the event and opens the newly created event specification.

Check State Model

To check a state model:

1. Open the state model.
2. Maximize the diagram window.
3. Right-click in empty space with no symbols selected.
4. Select the **PathMATE Check** menu option.

   The tool will check the state model and interactively prompt you to correct errors that it detects.

   When an error is detected, the item with the error is selected. You can adjust the zoom factor by pressing the Zoom In button to make the diagram bigger or the Zoom Out button to make the diagram smaller. Select one of the options to resolve the error or Skip to skip the error. Click **Cancel** to stop the error check. Any errors that are not resolved interactively are captured in the Rose Error Log. To display the error log, select the **Window > Log** menu item.

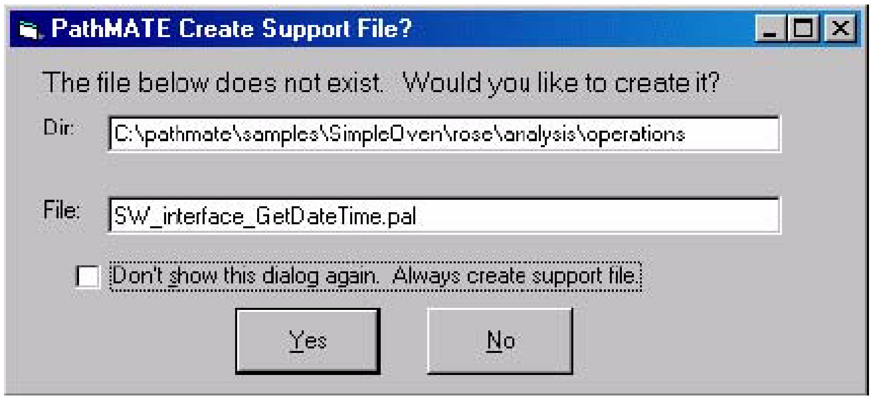
Action Modeling

Action modeling (for states and services) creates text files in subdirectories of the model directory as described in *Model Creation* on page 6. You may choose your favorite editor to associate with the .pal action language text files. If you have not yet done so, please follow the instructions in the installation section for associating your favorite editor with the .pal file extension.

- To open a service action body: Go to the Rose browser and select the service you want to edit. Right-click and select **PathMATE Open Action**.
- To open a state action body: Go to the Rose browser and select the state you want to edit. Right-click and select **PathMATE Open Entry Action** to edit the entry or exit action.
- To open a system initialization action: Deselect all symbols on any diagram. Right-click on empty space in any diagram and select **PathMATE Open System Init** to edit the system initialization action.
- To open a domain initialization action: Select a domain package in the browser. Right-click and select PathMATE Open Domain Init to edit the domain initialization action. Alternately, deselect all symbols on a diagram scoped to a domain package. Right-click on empty space and select **PathMATE Open Domain Init**.

Creation of Action Language Files

If you do not have the Auto Create File option enabled, a dialog box asks you to confirm creation of the file (Figure 2). Click **Yes** to create the file specified and open it in the text editor. Click **No** to skip creation of the file.


PathMATE Create Support File?

The file below does not exist. Would you like to create it?

Dir: C:\pathmate\samples\SimpleOven\rose\analysis\operations

File: SW_interface_GetDateTime.pal

Figure 3-1. Create Support File? Dialog Box

NOTE

If you want to create action language files without prompting, check Don’t show this dialog again. Always create support file. *The next time you open an action language file that doesn't exist, the file is created automatically. To change the Auto Create File option, select **Tools > PathMATE Options**.*

When the action file is initially created, a header comment describes the context for the action. You may delete all comments prefaced by //!. If you do not delete the comments, the context will be automatically updated each time you open the action language file and the file is writable. If you do not want the header comments to update automatically, select **Tools > PathMATE Options** and remove the check from the Auto Update Header option.

Storage of Action Language Files

Action files are stored in their corresponding support files subdirectories. File names connect files to their analysis elements.

- *system initialization:* <system support dir>/init/<system name>.pal
- *domain initialization:* <domain support dir>/init/<domain prefix>.pal
- *domain service:* <subsystem support dir>/operations/<domain prefix>_<service class prefix>_<service name>.pal
- *class service:* <subsystem support dir>/operations/<domain prefix>_<class prefix>_<service name>.pal
- *state entry action:* <subsystem support dir>/state_actions/<domain prefix>_<class prefix>_<state name>_entry.pal
- *state exit action:* <subsystem support dir>/state_actions/<domain prefix>_<class prefix>_<state name>_exit.pal
- *state transition actions:* Single-line transition actions may be specified using the built-in Rose mechanisms.

  Select the transition, open the Specification, select the Detail tab, and use the Action field.

  If a more complicated transition action is required, use a PAL file via the transition menu item **PathMATE Open Action**.

  The PAL file is stored in <subsystem support dir>/state_actions/<domain prefix>_<class prefix>_<source state>_<dest state>_<trigger>_<internal Rose transition identifier>_trans.pal.

  If a PAL file is specified, extract uses the action language in the file, not in the action field on the diagram.
- *state transition guard expressions:* Guard Expressions are not stored in PAL files – they are stored using built-in Rose mechanisms. To capture/update these, select the transition, open the Specification, select the Detail tab, and use the Guard Condition field.
- *system support directory* **–** PAL and type files at the system level are stored relative to the system .mdl file.
- *domain support directory* **–** If the domain is controlled, PathMATE stores the domain's .pal and .typ files relative to the .cat file. If the domain is not controlled or the PALLocation property for the domain is set to SystemRelative, PathMATE stores the files relative to the .mdl file.
- *subsystem support directory* **–** PAL files at the subsystem level may be stored relative to the domain or subsystem directories.

If a subsystem is controlled in a .cat file, the PAL files are stored relative to the location of the .cat file if the PALLocation property is set to SubsystemRelative and in the domain location if the PALLocation property is set to DomainRelative or SystemRelative.

If the subsystem is uncontrolled, the PAL files are located using the rule of the closest controlled subsystem or domain.

The following table summarizes key information for the various kinds of PAL files:

Table 3-1. Summary Information for PAL Files

Support Files	Controlled	PALLocation Property	Support Directory
system	N/A	N/A	relative to .mdl file
domain	No	Any	relative to .mdl file
domain	Yes	Domain Relative	relative to domain .cat file
domain	Yes	System Relative	relative to .mdl file
subsystem	No	Any	follow rule of closest controlled parent domain or subsystem
subsystem	Yes	Subsystem Relative	relative to subsystem .cat file
subsystem	Yes	Domain or System Relative	follow rule for domain

User Defined Types

User defined types are specified in a text file stored in the types subdirectory of your model. You may define types that are aliases to an action language type or you may define an enumerated type. You may choose your favorite editor to associate with the .typ text files. If you have not yet done so, please follow the instructions in the installation section for associating your favorite editor with the .typ file extension.

NOTE

The procedure used to create a type file is similar to that used to create a PAL file. See Creation of Action Language Files on page 20 for more information.

System Level Types

Types defined at the system level are accessible to all domains. Domain service parameter types must be defined at the system level.

1. Open any diagram and deselect all symbols.
2. Right-click in the diagram.
3. Select **PathMATE Open System Types**. The types file will open in the editor associated with the .typ extension. Add the new type to the file.

Domain Level Types

Types defined at the domain level are only accessible within the domain.

1. Open any diagram and deselect all symbols.
2. Right-click in the diagram.
3. Select **PathMATE Open Domain Types**. The types file will open in the editor associated with the .typ extension. Add the new type to the file.

Referencing a User Defined Type

User defined types may be used anywhere a type is needed, for example, attribute types and event and operation argument types.

1. Open the specification for the attribute, event argument, or operation argument.
2. Enter the name of the user defined type in the type field.

Constants

Constants may be defined at the domain or system level. Constants are initialized once at startup and never changed. Constants may be read in actions but may not be used on the left hand side of an assignment or passed as an output parameter.

System Level Constants

Constants defined at the system level are accessible to all domains.

1. Open any diagram and deselect all symbols.
2. Right-click in the diagram.
3. Select **PathMATE Open System Init**.

   The system initialization action opens in the editor associated with the .pal extension. Define the constant in the initialization file. Please see the Action Language Overview for an explanation of the global constant definition syntax.

Domain Level Constants

Constants defined at the domain level are only accessible to the actions in the domain where they are defined.

1. Select the domain where the constant will be defined in the browser.
2. Right-click in the diagram.
3. Select **PathMATE Open Domain Init**.

   The domain initialization action opens in the editor associated with the .pal extension. Define the constant in the initialization file. Please see the Action Language Overview for an explanation of the global constant definition syntax.

4. Generate Code and Reports

You generate code interactively, or from a batch file. Likewise, you can generate reports with either method.

Generate Interactively

To generate code or reports interactively:

1. Open the system model in Rose.
2. Select **Tools > PathMATE Generate** tope menu bar. Then select **C++**, **Java**, **C**, or **Reports**.

The tool checks the model errors interactively. If errors are detected, the tool will report the error message along with several options for fixing or skipping the error. If errors are skipped or error checking is cancelled, generation exits and displays an error message to the user. Any unresolved error messages are captured in the error log. View the error log by selecting the **Window > Log** menu item. After correcting the error messages, select the generate menu option again.

When an error is detected, the diagram with the error opens automatically. The Rose extensibility interface is modal, so you cannot scroll through the open diagram. You can, however, adjust the zoom factor when an error is detected. If you maximize an open diagram before you select **PathMATE Generate**, the diagram with the error is maximized as well, and you can see the erroneous diagram more easily.

If the system detects no errors, the PathMATE Extract Progress dialog opens. The tool reads the models and produces a standard XML interchange format file.

If this is the first time you have generated reports or code for this model, the Code Generation Options dialog opens. Select the location of the generated files and template path. Click **OK** to close the dialog. The Engine reads the interchange file and interprets the templates to produce the target files. The next time you select **PathMATE Generate**, the Engine operates with the options you selected the first time.

If you want to change the code generation options you originally selected, delete the batch file called after extracting the models to the standard interchange format. For reports, locate genreports.bat in your reports directory. For code, locate *genc.bat, gencpp.bat, or genjava.bat* in the generated code directory. PathMATE opens the Code Generation Options dialog the next time you generate.

NOTE

Alternately, you can open the batch file in your text editor, and modify the destination directories directly in the batch file. Either method of changing the code generation options works equally well.

If the Engine detects errors, the error log file sprngbrd.err opens. Correct the errors and select **PathMATE Generate** again.

If the Engine does not find any errors, it generates the desired files. By default, generated files are placed in a path relative to your models. The default directory for generated reports is *<model directory>\..\reports*. The default directory for generated code is *<model directory>\..\project\<language>\gc*.

Generate from a Batch File

In the batch file, specify the following command line to extract a Rose model to an XML file:

```
start /wait rose_extract <model_path>.mdl
```

The extracted XML file will be generated in the same directory as the .mdl file. Use the XML file as an input to the springboard command using the –x option. See the *PathMATE Transformation Engine User's Guide* for more information about the Engine command line options.

After rose_extract completes, check the log file for errors. Errors in command line options and PathMATE licensing are reported to the file PathMATERoseExtract.log located in the directory from which rose_extract is run.

NOTE

If Rose is running when you invoke rose_extract, rose_extract will not function properly. Ensure that Rose is closed before you run the rose_extract program.

5. Rename Analysis Elements

PathMATE for Rose augments the information in the Rose standard repository with supplemental information required for complete platform independent models. This supplemental information is stored in PathMATE -specific Rose repository fields, and in text files (.pal and .typ). As you rename model elements in the Rose standard repository, the PathMATE Rename utility performs any required updates of supplemental information.

NOTE

Note: *The International version of Rose 2001 does not support notifications of rename. When using this version, you must use the PathMATE Rename menu options.*

The Rename utility may be run manually or be automatically triggered from a Rose rename operation on one of the following types of model elements:

- Attribute
- Binary Association
- Domain (package)
- Event (<<Event>> operation)
- Class
- Class and Domain Service (operation)

The following model element types require the Rename utility to be run manually:

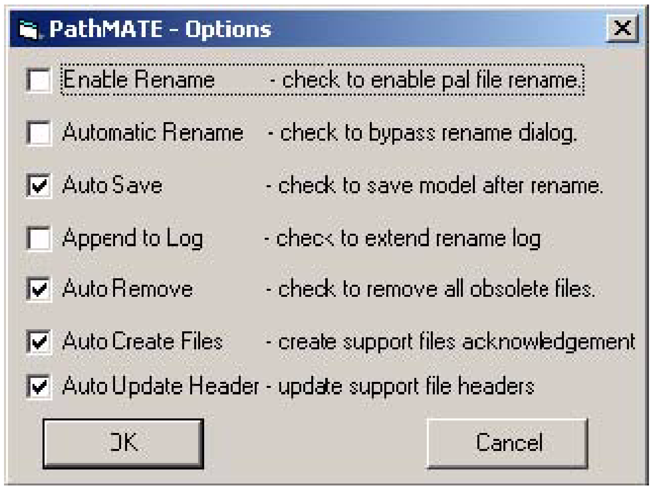
- SubSuper relationships
- Realization relationships
- Parameters
- User defined types
- State

See the sections below for details on how to rename these element types.

If Rename is not automatically activated from appropriate Rose rename activities, check to make sure the *PathMATE for Rose* add-in is enabled by selecting **Add-ins > Add-In Manager** from the Rose menus, and checking the *PathMATE* check box.

Options Dialog

To set PathMATE options, select the **Tools > PathMATE Options** menu item. The Options dialog opens:

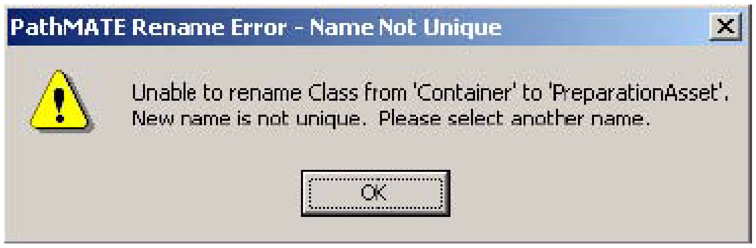


- [ ] Enable Rename - check to enable pal file rename.
- [ ] Automatic Rename - check to bypass rename dialog.
- [x] Auto Save - check to save model after rename.
- [ ] Append to Log - check to extend rename log
- [x] Auto Remove - check to remove all obsolete files.
- [x] Auto Create Files - create support files acknowledgement
- [x] Auto Update Header - update support file headers

- Check *Enable Rename* must be checked to authorize the Rename utility.
- Check *Automatic Rename* to run the Rename utility without displaying the PathMATE Rename dialog.
- Check *Auto Save* to save the Rose model after each rename. This option is recommended.
- Check *Append* to Log to append text to the existing rename log file.
- Check *Auto Remove* to remove obsolete files automatically.
- Check *Auto Create Files* to create action language and type files without prompting.
- Check *Auto Update* Header to update the //! header comments automatically when opening writable .pal and .typ files.

Uniqueness Checks

When you rename an analysis element, PathMATE checks the new name for uniqueness. If the newly selected name is not unique, the following error dialog appears:


PathMATE Rename Error - Name Not Unique

Unable to rename Class from 'Container' to 'PreparationAsset'.
New name is not unique. Please select another name.

The rename will fail. If rename was initiated through the **PathMATE Rename** menu item, select the menu item again and enter a new name. If rename was initiated through the specification and rename was enabled, change the name back to the original name by disabling rename, changing the class name to its original name, and then enabling rename again.

Table 5-1 summarizes the range of PathMATE's view when it checks for the uniqueness of an analysis element's name.

Table 5-1. Uniqueness Checks for the Names of Analysis Elements

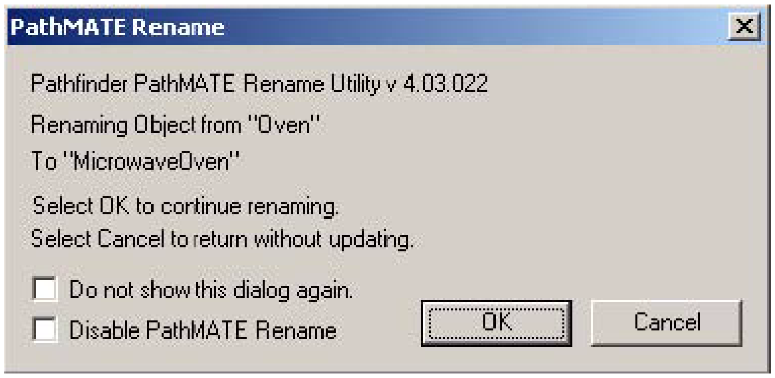
Analysis Element	Make Element's Name Unique Within This Range
Association	All associations contained in domain and its subsystems
Attribute	Attributes in containing class
Class	All classes in the domain and its subsystems. If class does not have a prefix, class name must also be a unique prefix
Class Prefix	All class prefixes within the domain and its subsystems
Domain	All domains within the system. If domain does not have a prefix, domain name must also be a unique prefix
Domain Prefix	All domain prefixes for all domains in the system
Event	All operations in the containing class with <<Event>> stereotype
Operation	All operations in the containing class without <<Event>> stereotype
Parameter	Parameters of the operation
Realization Relationship	All SubSuper and Realization relationships in the domain and its subsystems
Role	Role names on the owning association
State	All states and nested states on the statechart
SubSuper Relationship	All SubSuper and Realization relationships in the domain and its subsystems
User Defined Type	All user defined types within the owning domain or system

Rename Dialogs

The next dialog box is displayed at the start of the Rename utility. Click **OK** to continue renaming. Click **Cancel** if you not want to rename at this time.

Check *Do not show this dialog again* to bypass this dialog. To re-enable the dialog, uncheck the *Automatic Rename* box in the Rename Options dialog.

Check *Disable PathMATE Rename* to disable rename. The Rename utility will not run until you check *Enable Rename* in the Rename Options dialog.


PathMATE Rename

Pathfinder PathMATE Rename Utility v 4.03.022

Renaming Object from "Oven"

To "MicrowaveOven"

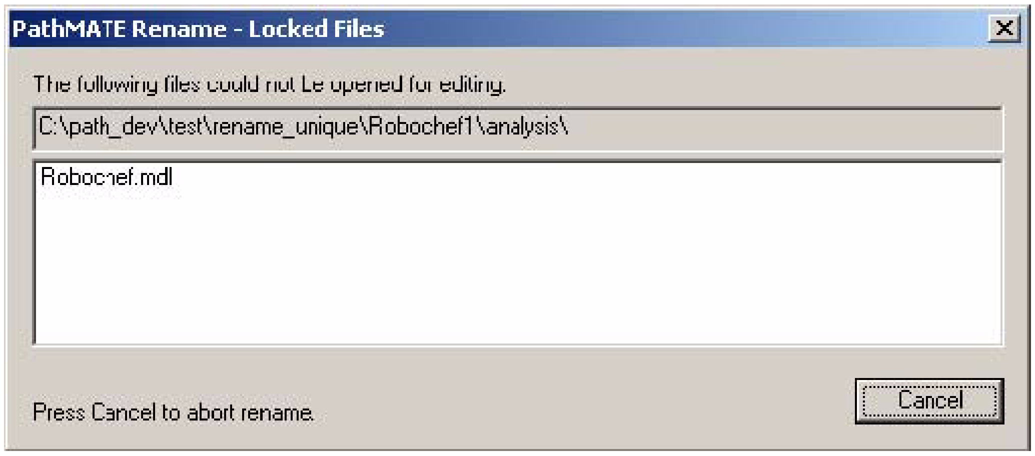
Select OK to continue renaming.
Select Cancel to return without updating.

☐ Do not show this dialog again.

☐ Disable PathMATE Rename

Locked Files

If a model file or supplemental file requiring changes is read-only, the following error dialog is displayed:


PathMATE Rename - Locked Files

The following files could not be opened for editing.

C:\path_dev\test\rename_unique\Robochef1\analysis\

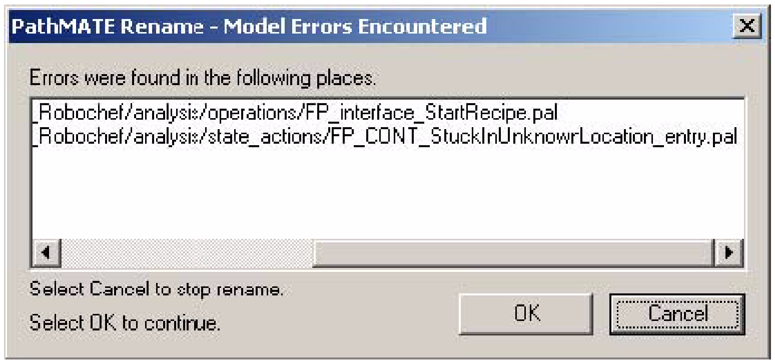
Robochef.mdl

Press Cancel to abort rename.

If you are using a configuration management system, check out the listed files. If you are not using a configuration management system, clear the read-only attribute of the files. A complete list of the files can be found in the rename log. Click **Cancel** to close the box. Attempt the rename again after the files are unlocked.

Model Errors Encountered

The next dialog is displayed only if a parsing error is encountered in one or more of the PAL files or state model strings. Click **OK** to ignore the errors. Click **Cancel** to stop the rename utility.



PathMATE Rename - Model Errors Encountered

Errors were found in the following places.

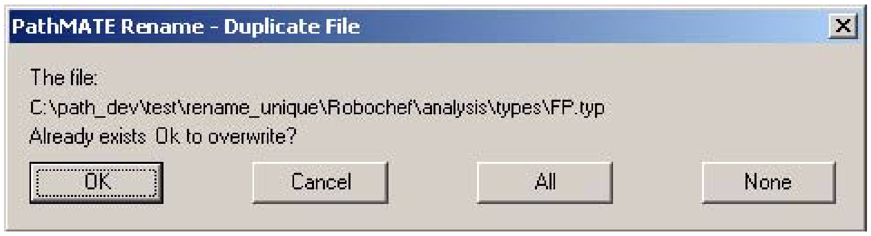
```
Robochef/analysis/operations/FP_interface_StartRecipe.pal
Robochef/analysis/state_actions/FP_CONT_StuckInUnknowrLocation_entry.pal
```

Select Cancel to stop rename.

Select OK to continue.

Duplicate File

Changes to some element types, such as Class, may require that PAL file names be updated to reflect the new element name. In the case where the new filename already exists, the next dialog appears.



PathMATE Rename - Duplicate File

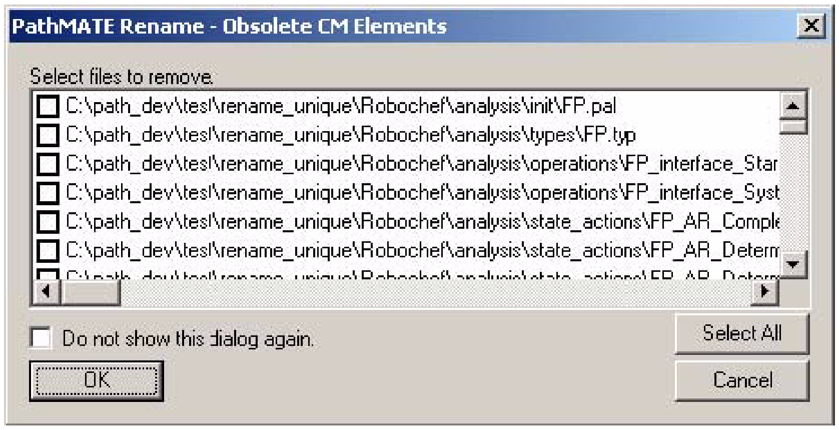
The file:

C:\path_dev\test\rename_unique\Robochef\analysis\types\FP.typ

Already exists. Ok to overwrite?

Obsolete Files

Renaming of files will also cause the old filename to be obsolete. The next dialog shows the files that are no longer in use. Select the files that can be deleted and press OK. To automatically delete all obsolete files, check the Auto Remove box in the Rename Options dialog.


PathMATE Rename - Obsolete CM Elements

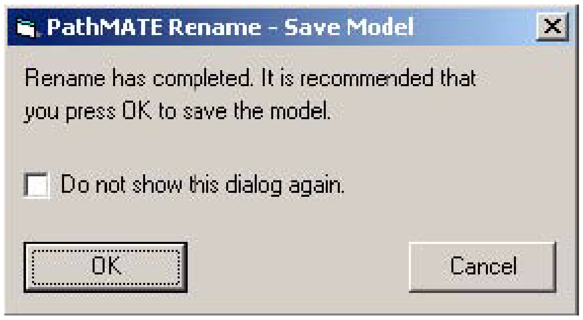
Select files to remove:

- [ ] C:\path_dev\test\rename_unique\Robochef\analysis\init\FP.pal
- [ ] C:\path_dev\test\rename_unique\Robochef\analysis\types\FP.typ
- [ ] C:\path_dev\test\rename_unique\Robochef\analysis\operations\FP_interface_Star
- [ ] C:\path_dev\test\rename_unique\Robochef\analysis\operations\FP_interface_Syst
- [ ] C:\path_dev\test\rename_unique\Robochef\analysis\state_actions\FP_AR_Comple
- [ ] C:\path_dev\test\rename_unique\Robochef\analysis\state_actions\FP_AR_Determ

Save Model

When all changes required for the rename are complete, the next dialog appears to request that the updated Rose model be saved. Click **OK** to save the model. If you click **Cancel** and close the Rose model without saving it, the PAL files must be regenerated the next time you open the model.

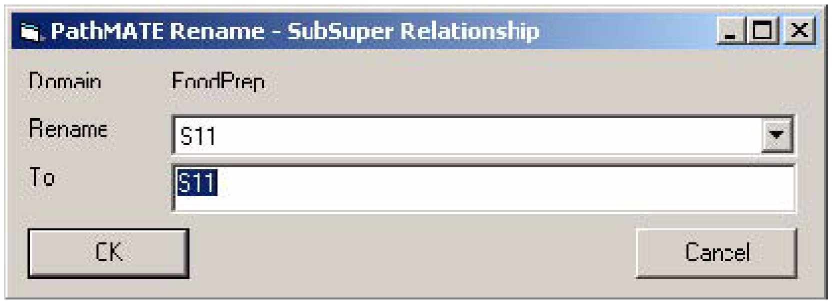
If you want this dialog to prompt you to save the model, uncheck the *Auto Save* checkbox in the Rename Options dialog.


PathMATE Rename - Save Model

Rename has completed. It is recommended that you press OK to save the model.

SubSuper Relationship Rename

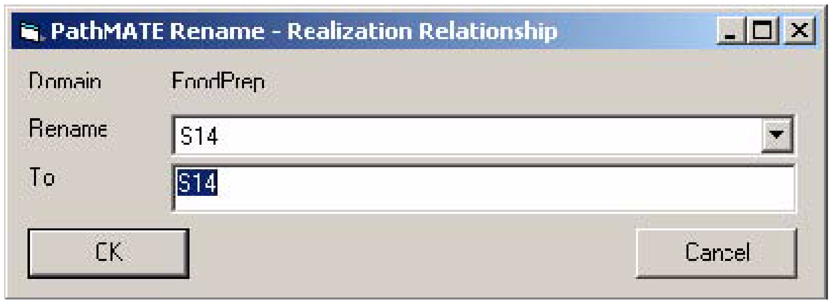
Select the package for the domain. Right-click and select **PathMATE Rename > SubSuper**. The following dialog appears. Select the SubSuper relationship to be renamed from the list and enter the new name below. Alternately, you may right-click in the empty space of any diagram in the package and select **PathMATE Rename > SubSuper**.



Domain FoodPrep
Rename S11
To S11

Realization Relationship Rename

Select the package for the domain. Right-click and select **PathMATE Rename > Realization**. The following dialog opens. Select the Realization relationship to be renamed from the list and enter the new name below. Alternately, you may right-click in the empty space of any diagram in the package and select **PathMATE Rename > Realization**.


PathMATE Rename - Realization Relationship

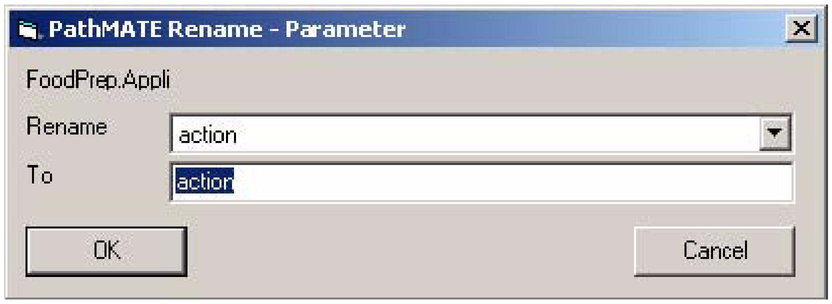
Domain FoodPrep

Rename S14

To S14

Event and Operation Parameter Rename

Select the operation or event. Right-click and select **PathMATE Rename Parameter**. The following dialog appears. Select the parameter to be renamed from the list and enter the new name below.


PathMATE Rename - Parameter

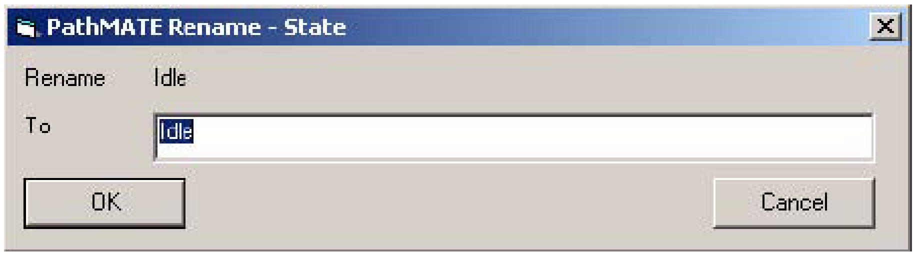
FoodPrep.Appli

Rename action

To action

State Rename

Select a state. Right-click and select **PathMATE Rename**. Enter the new state name in the text box.

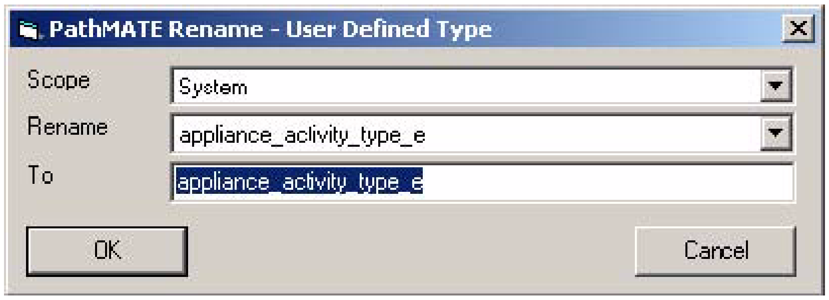

PathMATE Rename - State

Rename Idle

To Idle

User Defined Type Rename

To rename a domain-level type, select the domain (or any of its sub-subsystems), right-click and select **PathMATE Rename > Type**. Select the scope of the type and then select the type name. Enter the new type name. Alternately, you may right-click in the empty space of any diagram in the package and select **PathMATE Rename > Type**.



Scope: System
Rename: appliance_activity_type_e
To: appliance_activity_type_e

Selecting an Event for a Transition

If a transition has an action file associated with it, the action file must be renamed when the event triggering the transition is changed. The name of the action file will be automatically updated when rename is enabled and **PathMATE Select Event Spec** is selected from the transition context menu. If rename is disabled and the transition has an action file, PathMATE warns the user that the transition action file needs to be renamed by hand.

Rename log file

Each time the Rename Utility is used, details about the change are written to the rename log file. The log file named *RenameLog.txt* is written to the current working directory. A new log file is created for each renamed element unless the *Append To Log* box is checked in the Rename Options dialog.

6. User Defined Properties

Create a Rose Add-In

Create a Rose add-in to extend the properties of elements in Rose. The Windows registry holds information about what add-ins are available for use. Add-ins can be turned on or off using the Add-in Manager from the Add-ins menu. Each add-in has a set of properties for the various elements in Rose. A property typically has a default value, and that can be overridden as necessary. Before adding an add-in, close Rose.

Define Properties

User defined design properties are defined in a .pty file.

1. Copy the file *propertyTemplate.pty* in the *<InstallDir>\config\rose* directory to a local work area. Rename the file to *<add-in-name>Properties.pty*.
2. Open the .pty file in any text editor.
3. Replace the text in the .pty file as follows:

   <tabName> – Name of the tab that appears in the Specification box. The name should be similar to your add-in name, and distinct from other add-ins you have installed.
4. Using the template as a guide, specify the properties that you would like to add. Search for the string # *<analysis element type> Properties* to see an example of how to add properties to a specific analysis element type such as a domain or class.
5. Delete parts of the property template that you will not be using.
6. *Check that all parentheses match.* Rose will produce an error in the log for property files that fail to parse properly; this file will indicate where the syntax check failed.

Define Registry Settings

A .reg file creates entries in the registry. The following section contains instructions for adding registry entries necessary for Rose to recognize your add-in.

1. Copy the file *addRoseAddinTemplate.reg* in the *<InstallDir>\config\rose* directory to a local work area. Rename the file to *register<add-in-name>.reg*.
2. Replace the text in the .reg file as follows:

<addinName>	Name of Your Add-In
<drive>:\\<path_with_double_backslashes>	Fully qualified directory where files associated with the add-in will be stored. Use \\ in the file name in place of \.

Install Registry Settings

When you have finished updating the register<add-in-name>.reg file, invoke it by double clicking or right-selecting Open on it. It should return that the registration succeeded. If it does not, double check your .reg file for accuracy.

Once an add-in has been successfully registered, open Rose again, and verify that your add-in is active using the Add-in Manager from the Add-ins menu item.

Make Your Add-In Properties Accessible

To make your new add-in properties accessible to the PathMATE Transformation Engine, open the Specification for the Logical View and Select the PathMATE tab. Ensure that the Set is System, click IncludeTabs in the Name column, and click the blank space in the Value column next to Include Tabs.

Enter in this field the names of other tabs you would like the Engine to have access to, separated by commas. Note that the PathMATE tab is ignored if found in the list.

To access the property from the Engine Template language use the PROPERTY expression. For the name of the property use <tab name>.<property name>. Note that the tab name is omitted for properties defined on the PathMATE tab.

Conventions for User Defined Properties

- Always translate a model in write mode before you check it in.
- Do not change the default property sets in PathMATE, or in other user defined tabs, once the translator has set them.
- If you suspect you are getting the wrong properties, open the model for write and try retranslating.

Additional Information

User defined properties consist of a tab name with one or more property sets. One property set is defined as the default. When an element is created, the default property set is displayed on the user defined property tab.

More than one property set is used when an analysis element can mean more than one thing. For example, all analysis elements and diagrams in the system are contained in the Logical View package. In addition, domains are represented by packages. Thus there are two property sets for a package on the PathMATE tab – one for the system, and the default for the package. You will have more than one property set for user defined properties on operations, since an operation can be stereotyped as an Event or can be just an operation.

When reading the properties tabs for an analysis element, the Visual Basic interface to Rose can read only from the current property set. The current property set is the one you see when you open the specification and select the property tab.

In order to read another property set, you have to change the current property set. Unfortunately, this requires the model to be writable. In UML Essentials for Rose version 4.02.008 and lower, the extract program changed the property set to the required set (System for Logical View package and default for domain packages). This quality ensures that the correct property set is extracted, but it required the model to be writable. Requiring the model to be writable for translation was very inconvenient for users using configuration management systems.

In version 4.02.009 and later, the extract program gets the current property set if the unit, model or .cat file (depending on the analysis element being extracted) is read-only and does a set current property set if the unit is writable.

If all three of these conditions are met, the wrong property for an element could be extracted:

- You create an analysis element that has more than one property set on the tab.
- You do not open the tab to set any of the properties.
- You do not translate the model while it is writable.

The risk that all three things would happen together is small, because if you did not select the property set, you probably did not change any of the property defaults. Thus default values are used when the property is extracted in the templates.

7. Rose Issues

Documentation Diagrams

A Rose defect sometimes causes extra white space to appear in diagrams, or diagrams appear stretched in Rich Text Format Reports. To remedy this problem, perform the following steps in Rose for each diagram that appears incorrectly:

1. Select the diagram.
2. Select **View > Zoom In**.
3. Select **Format > Autosize All**.
4. Select **File > Save**.
5. Regenerate your report.

For more information go to:

http://www.rational.com/technotes/soda_html/SoDA_Word_html/technote_12119.html

Active and Selected Diagrams

Diagram menu items may work in the active diagram, but not diagrams selected in the browser. In the Rose extensibility environment, you cannot determine if the browser or a diagram has focus. When a diagram is selected in the browser and a diagram is active in the window, the diagram context menu will operate on the diagram that is active rather than the diagram selected in the browser.

A. PathMATE Menu Items

The table below describes PathMATE menu item selectability and connectivity with model elements. Menu items that open a PAL file require that the model be saved. PAL files are located in directories relative to the model file. If the model file is not saved, the tool does not know where to open or create the PAL files.

Table A-1. PathMATE Menu Items in Rational Rose

	Model Saved	Diagram Type	Type of iTem Selected
Attribute Menu			
PathMATE Rename	Yes	None	Exactly one attribute assigned to a class in a domain
Default Menu			
PathMATE Open > System > Types	Yes	Any	Any
PathMATE Open > System > Init	Yes	Any	Any
PathMATE Rename Type	Yes	Any	Any
Class Menu			
PathMATE Rename > Class	Yes	Any	Exactly one class assigned to a domain
PathMATE Rename > Prefix	Yes	Any	Exactly one class assigned to a domain
Diagram Menu			
PathMATE Open > System > Types	Yes	Any	None
PathMATE Open > System > Init	Yes	Any	None
PathMATE Open > Domain > Types	Yes	Any diagram assigned to domain	None
PathMATE Open > Domain > Init	Yes	Any diagram assigned to domain	None
PathMATE Rename > Type	Yes	Any	None
PathMATE Rename > SubSuper	Yes	Any diagram assigned to domain	None
PathMATE Update Event Args	Yes or No	Statechart assigned to class and domain	None
PathMATE Update Actions	Yes	Statechart assigned to class and domain	None

PathMATE Check	Yes or No	Statechart assigned to class and domain	None
Role Menu			
PathMATE Rename > Role	Yes	Class diagram in a domain	Exactly one role assigned to a class in a domain
PathMATE Rename > Association	Yes	Class diagram in a domain	Exactly one role assigned to a class in a domain
State Menu			
PathMATE Open > Entry Action	Yes	Statechart assigned to class and domain	Exactly one normal state
PathMATE Open > Exit Action	Yes	Statechart assigned to class and domain	Exactly one normal state
PathMATE Update Actions	Yes	Statechart assigned to class and domain	One or more normal states
PathMATE Rename	Yes	Statechart assigned to class and domain	Exactly one normal state
Transition Menu			
PathMATE Open Event Spec	Yes or No	Statechart assigned to class and domain	Exactly one transition
PathMATE Select Event Spec	Yes or No	Statechart assigned to class and domain	Exactly one transition
PathMATE Open Action	Yes	Statechart assigned to class and domain	Exactly one transition
PathMATE Update Action	Yes	Statechart assigned to class and domain	One or more transitions
Operation Menu			
PathMATE Open Action	Yes	None	Exactly one non-event operation in class assigned to domain
PathMATE Rename > Operation	Yes	None	Exactly one operation in class assigned to domain
PathMATE Rename > Parameter	Yes	None	Exactly one operation in class assigned to domain
Package Menu			
PathMATE Open > Domain Types	Yes	None	Exactly one domain
PathMATE Open > Domain Init	Yes	None	Exactly one domain
PathMATE Rename > Domain	Yes	None	Exactly one domain
PathMATE Rename > Prefix	Yes	None	Exactly one domain
PathMATE Rename > SubSuper	Yes	None	Exactly one domain

PathMATE Rename > Type	Yes	None	Exactly one domain
PathMATE Rename > Realization	Yes	None	Exactly one domain
Tools Menu			
PathMATE Generate > C++	Yes	None	None
PathMATE Generate > C	Yes	None	None
PathMATE Generate > Java	Yes	None	None
PathMATE Generate > Reports	Yes	None	None
PathMATE Options	Conditional	None	None