



---

## **PathMATE in Rhapsody: Modeling Guide**

Version 1.5  
July 29, 2009

---

### *PathMATE Technical Notes*

Pathfinder Solutions LLC  
33 Commercial Street, Suite 2  
Foxboro, MA 02035 USA  
[www.pathfindermda.com](http://www.pathfindermda.com)  
888-662-7284

# Table Of Contents

<b>1. Introduction .....</b>	<b>1</b>
Model Organization.....	1
Installation .....	2
Using a Single Site License (Floating License) .....	2
Supported Versions of Rhapsody .....	3
<b>2. Setting up the Model .....</b>	<b>4</b>
Create a New Blank Project .....	4
Import the PathMATE Profile and SoftwareMechanisms Domain.....	4
Set up the Project Hierarchy .....	5
<i>Apply System Stereotype to the Project .....</i>	<i>5</i>
<i>Set up Domains .....</i>	<i>5</i>
Splitting the Rhapsody Project into Multiple Files.....	6
<b>3. Modeling Topics .....</b>	<b>7</b>
Types .....	7
<i>Using Groups as Types.....</i>	<i>9</i>
Setting PathMATE Properties.....	9
Identifiers .....	9
System and Domain Initialization .....	9
<i>System Level .....</i>	<i>9</i>
<i>Domain Level.....</i>	<i>10</i>
Domain Interfaces.....	11
Subsystems.....	11
Transformation .....	11
Setting Diagram Properties for Report Generation .....	12
<i>Specifying Main Diagram .....</i>	<i>12</i>
<i>Handling Oversized Diagrams .....</i>	<i>12</i>
<b>4. Example Systems .....</b>	<b>14</b>
Available Samples .....	14
<b>5. Current Limitations .....</b>	<b>15</b>
<b>Appendix A – PathMATE Properties .....</b>	<b>16</b>
<b>Appendix B – PathMATE Types Mapped to Rhapsody Types.....</b>	<b>18</b>
<b>Appendix C – Error Messages .....</b>	<b>19</b>

1. Extraction Fails Immediately .....	19
<i>Task 1. Ensure that rhapsody.jar and rhapsody.dll exist. ....</i>	<i>19</i>
2. Prompted to modify the system path to include the parent directory containing rhapsody.jar. ....	19

## Table Of Figures

Figure 2-1 Rhapsody License Limit Error .....	2
Figure 2-1: Importing PathMATE Profile and SoftwareMechanisms Domain .....	5
Figure 2-2: Setting System Stereotype .....	5
Figure 2-3: Setting Domain Stereotype .....	6
Figure 3-1 Setting an Attribute Type.....	7
Figure 3-2 Selecting a Type from PathMATE Profile.....	8
Figure 3-3 System Initialization Tag .....	10
Figure 3-4 Domain Initialization Property .....	11
Figure 3-5 Rhapsody Package Properties .....	12
Figure 3-6 Rhapsody Diagram Properties.....	13
Figure C-0-1 PathMATE Console Error .....	19
Figure C-0-2 Example PathMATE Engine Error Prompt.....	19
Figure C-0-3 Example of Share/JavaAPI Folder .....	20
Figure C-0-4 System Properties .....	21
Figure C-0-5 Set Environment Variable .....	22

# 1. Introduction

This guide provides an overview of how to install and use Pathfinder Solutions *PathMATE* with Telelogic *Rhapsody* version 7.1+ (excluding 7.1.1). It is expected the reader is at least familiar with the Rhapsody environment and has experience creating MDA models using PathMATE.

While there are a wide range of ways to use Rhapsody, this document guides the reader in how to use Rhapsody in the creation of platform independent models that can be transformed, executed, debugged, tested and deployed with the PathMATE environment. The methodology supported is taught in the Pathfinder Solutions *Effective MDA* training.

The user may wish to complete the PathMATE for Rhapsody Quick Start Guide first to gain a feel for creating a simple platform independent model.

## Model Organization

PathMATE is a transformation environment that works on a System – a complete set of model information. A System is made up of

- A top level System package containing all other elements
- System-level user defined data types – visible to all components
- Use Cases and their constituent elements
- Diagrams
  - The System Domain Diagram (aka Domain Chart)
  - Any other diagram, to be used as a system supplemental diagram
- A System Initialization action
- Domains and their contents

Domains are packages that serve as logical components of the system. Domains contain:

- A domain class model
- Any other diagram, to be used as a system supplemental diagram
- An interface, containing domain services
- Classes
- Associations and generalization relationships
- Domain-level user defined data types – visible only within this component

- Subsystems – packages used to organize the contents of a complex domain. These contain the same elements a domain can.

## Installation

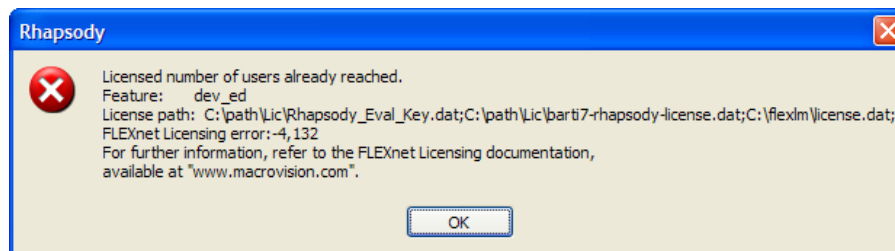
Install PathMATE for Rose, RSM, and Rhapsody 7.3+. Installation instruction can be found in the *PathMATE Quick Start Guide for Rhapsody*, under "Downloading and Installing PathMATE".

## Using a Single Site License (Floating License)

If you are using a the Rhapsody/Rational license server with a Single Site License, Rhapsody limits your computer to running one instance of the Rhapsody.exe process at a time, in comparison to a Node-Locked license which allows a single computer to run as many instances of Rhapsody as resources allow. The PathMATE engine does not connect to a current running instance of Rhapsody, and force the load of the model to be transformed, instead if it finds that the model is not currently loaded, the model is opened in a new instance of Rhapsody. As a result, a developer must either close all instances of Rhapsody before transforming or ensure that the model is already loaded in Rhapsody (recommended).

Please note, that as of Rhapsody version 7.3, Rhapsody can be integrated with Eclipse. If your instance of Eclipse is running with the Rhapsody integration, you need to be aware that the *Rhapsody Perspective* (if open), launches an instance of Rhapsody in the background that will count towards your license count. You, must either (1) ensure that the model you will be transforming is designated as the active model in the *Rhapsody Model Explorer* in the *Rhapsody Perspective* or (2) close the *Rhapsody Perspective* by right clicking on the perspective tab in Eclipse and selecting **close**.

Please note, that if you receive an error message (similar to the one below) stating that you cannot checkout a license, but don't believe that Rhapsody is running launch Task Manger and close any *Rhapsody.exe* processes found in the *Process Tab*, under certain situations Rhapsody does not properly close, leaving these processes running in the background.



**Figure 2-1 Rhapsody License Limit Error**

## **Supported Versions of Rhapsody**

- Rhapsody 7.3.x
- Rhapsody 7.4.x
- Rhapsody 7.5.0

## 2. Setting up the Model

Open Rhapsody and follow these procedures to create a proper container for your model.

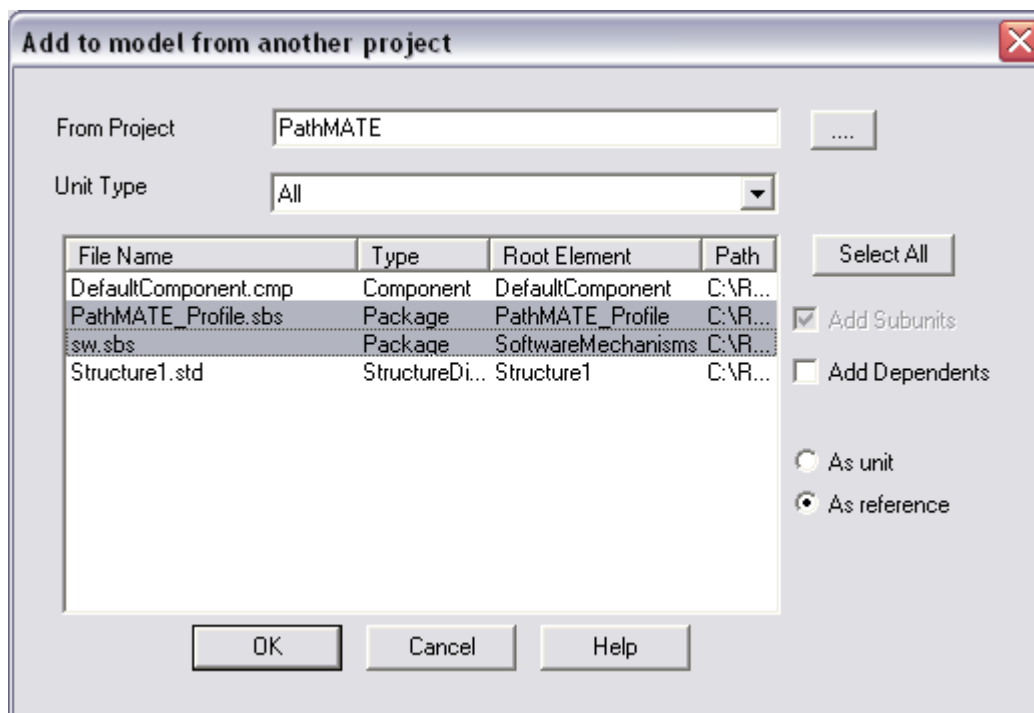
### Create a New Blank Project

1. File → New
2. Specify project name and path
3. Specify "Type" as "Default"

### Import the PathMATE Profile and SoftwareMechanisms Domain

The PathMATE Profile includes important type definitions and stereotypes which allow the user to configure PathMATE transformation properties of model elements. The SoftwareMechanisms domain provides many useful operations, which can be directly called in action language.

1. File → Add to Model...
2. Select the PathMATE Rhapsody project file located at:
3. C:\pathmate\config\Rhapsody\PathMATE\PathMATE.rpy
4. Select "PathMATE\_Profile.sbs" and "sw.sbs"
5. Select "As reference"
6. Click "OK"



**Figure 2-1: Importing PathMATE Profile and SoftwareMechanisms Domain**

## Set up the Project Hierarchy

### *Apply System Stereotype to the Project*

The System Stereotype allows the PathMATE Transformation Engine to detect a properly configured, PathMATE transformable model.

1. Double click on the project container (the highest level folder in the project explorer) to bring up the features window
2. In the "Stereotype" field, select "System in PathMATE\_Profile"
3. Click "OK"



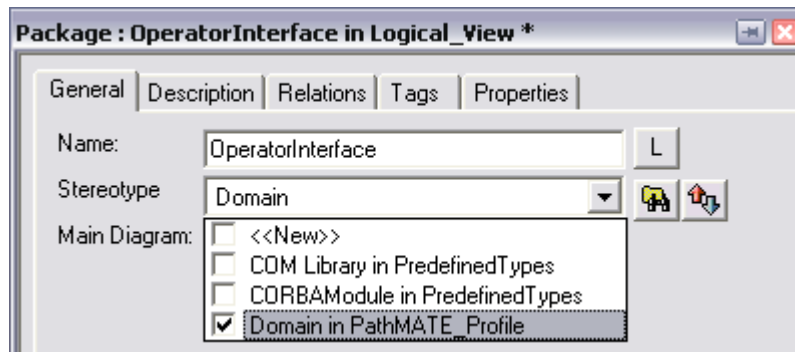
**Figure 2-2: Setting System Stereotype**

### *Set up Domains*

1. Create packages within the project



2. Apply the "Domain in PathMATE\_Profile" stereotype to create domains.



**Figure 2-3: Setting Domain Stereotype**

## Splitting the Rhapsody Project into Multiple Files

To dedicate a separate file to a model element, simply right click on a model element in the project explorer and select "Create Unit". The model elements will still be contained by the project, but will exist in a separate file on disk.

It is also possible to bring in units from other projects by using the "reference" command.

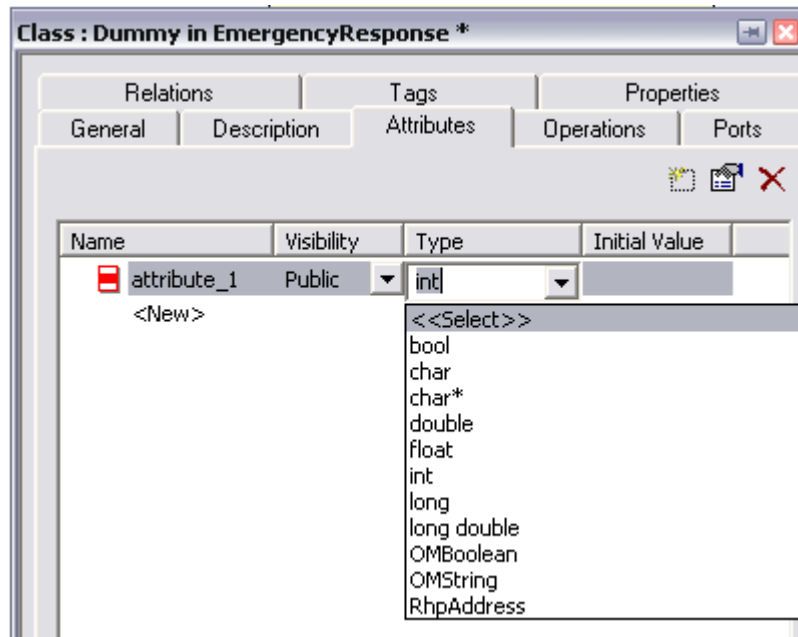
See the Rhapsody help files for more information on these features.

## 3. Modeling Topics

### Types

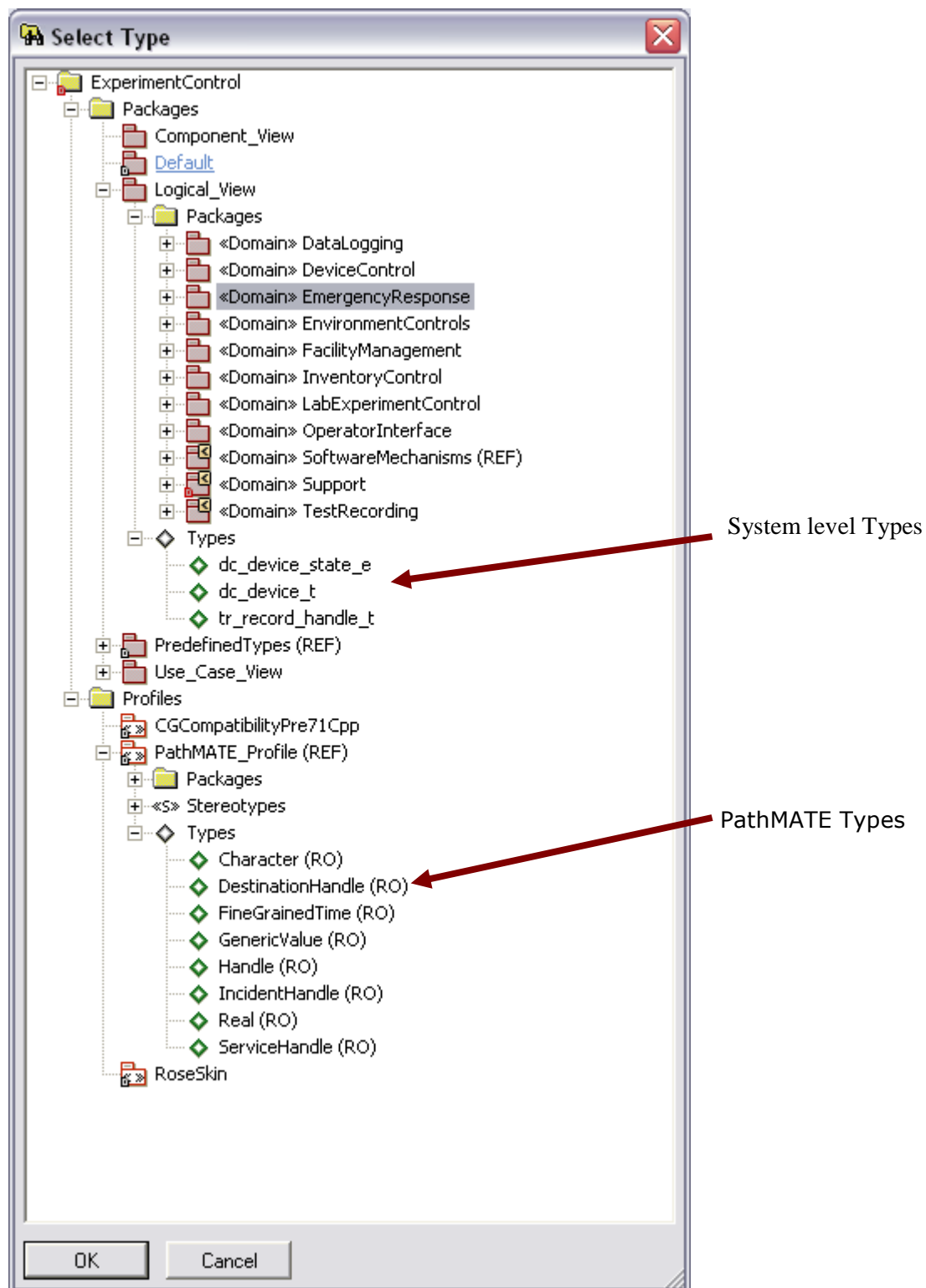
In order to make the transition to PathMATE easier for those already familiar with PathMATE or Rhapsody, the PathMATE Rhapsody integration supports both the Rhapsody built-in types and PathMATE's own types by detecting equivalent Rhapsody types and converting them to PathMATE types. See *Appendix B* for more information about which types are mapped.

When selecting the type for a model element from the pre-defined list it is possible to select one of the Rhapsody built-in types, a local package type, or the "<<Select>>" option. The "<<Select>>" option brings up the "Select Type" dialog, allowing types from anywhere in the project to be selected.



**Figure 3-1 Setting an Attribute Type**

The Select Types dialog allows the selection of types from higher level packages and PathMATE specific types from the PathMATE Profile.

**Figure 3-2 Selecting a Type from PathMATE Profile**

It is also possible to use the “Language Declaration” field to enter types directly as text.

### **Using Groups as Types**

In order to use Groups as parameters and attributes, a primitive type needs to be created in the Model for each Group of a particular type. This is done by:

1. Creating a Rhapsody Primitive type of a *TypeDef*.
2. Naming the type *Group\_Type* (see the following paragraph).
3. Setting the base type to handle.

The required naming convention for is the type name, as utilized in PAL, for example Integer, with a prefix of “Group\_”. An example for a Group of Integers is “Group\_Integer”.

After the type is created Groups can be used as parameters and attributes. Also it is suggested that these primitive types be created at the system level.

## **Setting PathMATE Properties**

Many PathMATE specific properties have been mapped to Rhapsody built-in properties. In order to access other properties a stereotype must be applied in the manner described earlier. A stereotype exists for each type of model element. Access PathMATE properties from the “Tags” tab of the features window on any model element.

Other properties are only available through the PathMATE markings file.

See *Appendix A* for more information about where to access specific PathMATE properties.

## **Identifiers**

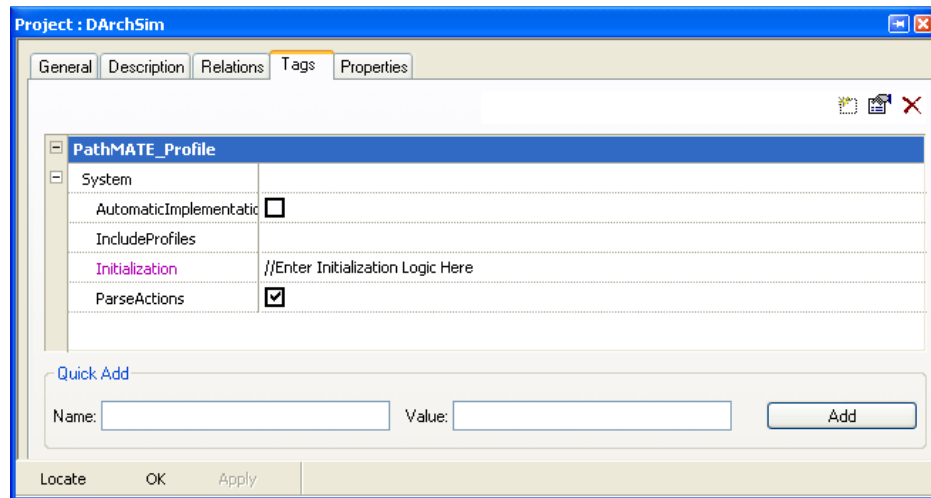
Identifier attributes are denoted with the “Identifier” PathMATE Stereotype.

## **System and Domain Initialization**

### **System Level**

System initialization code is executed during system start-up. The PathMATE stereotype “System”, contains a field, “Initialization” to hold this logic. The following steps describe how to add the System initialization AL to the model.

1. Double click on the system in the Project Explorer to open the features dialog.
2. Select the **Tags** tab.
3. Click in the **Initialization** entry field to activate text entry.



**Figure 3-3 System Initialization Tag**

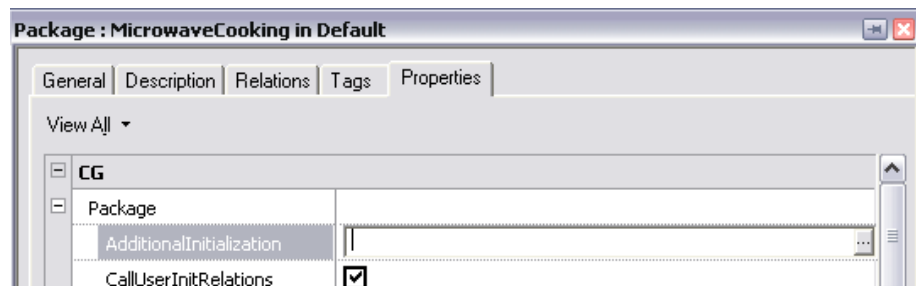
4. Press the "..." button in the *Initialization* field to bring up a bigger "Text Editor" window to work with.
5. Enter the initialization AL here.

### Domain Level

The Domain initialization logic is executed during system start-up following the System level execution. Rhapsody packages, have a predefined property, which is used to store the domain initialization AL, *CG>Package>AdditionalInitialization*. The following steps describe how to add the Domain initialization AL to the model.

Please note, that while this property exist on the System Package, it is the "Initialization" tag that is utilized. This due to the nature of Rhapsody properties, being propagated to contained components. Any initialization actions that would be applied at the system level using the property, would also be propagated to each contained domain..

1. Double click on the domain in the Project Explorer to open the features dialog.
2. Select the **Properties** tab.
3. Select the **View Common** triangle near the top left of the window, then select **View All**.
4. Expand *CG* and *Package* if needed.
5. Click in the **AdditionalInitialization** entry field to activate text entry.



**Figure 3-4 Domain Initialization Property**

6. Press the "... " button in the *AdditionalInitialization* field to bring up a bigger "Text Editor" window to work with.
7. Enter the initialization AL here.

## Domain Interfaces

Domain interfaces define operations which can be accessed from outside the containing domain.

Create an "interface" model element with the domain's implementation name in each domain. The stereotype on this element does not matter. Any operations contained within this interface will function as domain services.

## Subsystems

Each package below a domain is considered a subsystem automatically. See the PathMATE documentation on subsystems for more information.

## Transformation

To transform the model and get feedback about errors in the model or action language:

1. Open Eclipse with PathMATE (with Rhapsody integration)
2. Create a new Project
3. Create a new PathMATE Project (File → New → Other..., "PathMATE Project")
4. For the Platform Independent Model Path, select the Rhapsody Project file (\*.rpy). (If the project file is not displayed as selectable when browsing the file system, type \*.\* and then press enter to override the filter and display all files. This is a known issue.)
5. Select (or create) a deployment
6. Select "Transform"

At this time, PathMATE will attempt connect to any running instance of Rhapsody. If the project specified as the PIM is the active project in a running instance of Rhapsody, the connection will be made. If not,

PathMATE will automatically start up a new instance of Rhapsody and open the project.

PathMATE then reads all the model information from Rhapsody into its own semantic repository and performs the transformation requested.

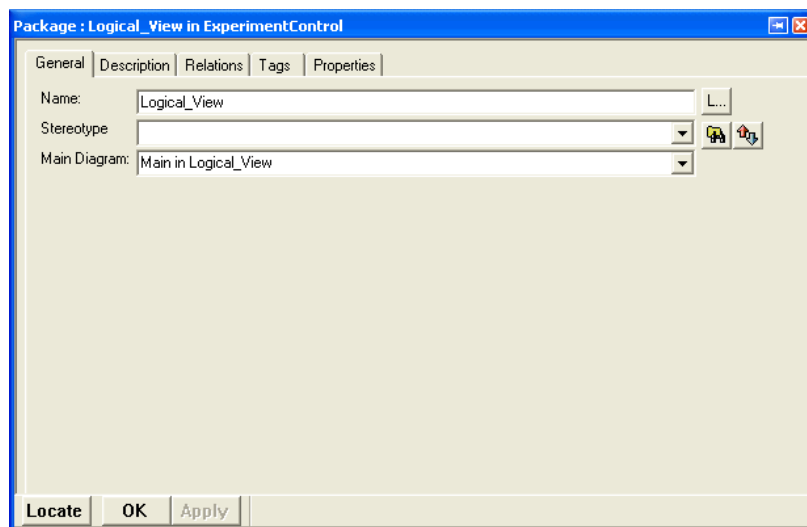
## Setting Diagram Properties for Report Generation

### *Specifying Main Diagram*

The domain chart and class diagrams are considered the “Main Diagram” for the system and each domain, respectively. Each diagram has to be explicitly identified in model properties.

Domain Chart:

1. Open Rhapsody Project
2. Select the top level package containing all the domains i.e. Default (Rhapsody) or LogicalView (Rose).
3. Open the properties for this package and select the “General” tab.



**Figure 3-5 Rhapsody Package Properties**

4. Specify the diagram represent the Domain Charts in the “Main Diagram” pull down menu.

Class Diagrams:

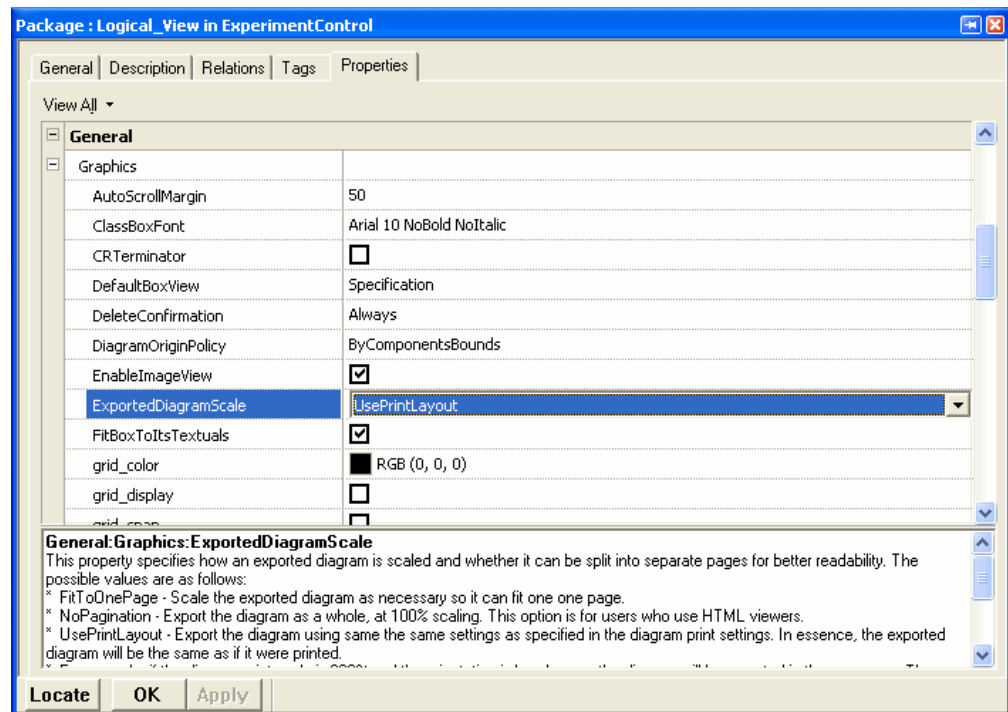
For the top model element (container) of each domain set the “Main Diagram” property for the appropriate Object Model Diagram, following the steps for setting the Domain Chart.

### *Handling Oversized Diagrams*

By default images exported from the Rhapsody modeling environment are not scaled to fit any particular page size. Diagrams that are too

large upon export to fit a page in Word, they need to set their "ExportDiagramScale" to UsePrintLayout.

1. Open Rhapsody Project
2. Select the oversized diagram in the model.
3. Open the properties for this diagram and select the properties tab and select "ExportDiagramScale" under General>Graphics.



**Figure 3-6 Rhapsody Diagram Properties**

4. Change the value in the pull down menu to "UsePrintLayout".

Please note, that this fix should not be applied universally through the project, because it does not only scale the width to 75% of a page but also the height. Images that originally would not have taken a whole page length, will now be extended with white space to encompass this area.



## 4. Example Systems

Pathfinder includes several example systems for educational purposes. To use any of these examples, follow these steps:

1. Start Eclipse.
2. Create a new example system project
  1. File → New → Example...
  2. Select the desired sample model under "PathMATE for Rhapsody".
  3. Press "Next".
  4. Specify a project name.
  5. Press "Finish".
3. Create a new PathMATE Project file
  1. File → New → Other...
  2. Select "PathMATE Project" under the "PathMATE" container.
  3. Press "Next".
  4. Select the newly created "rpy" file for the sample system in the example project.
  5. Press "Finish".
4. Save the PathMATE project
5. Transform, compile, and run

### Available Samples

- Simple Oven – This is a very simple example system that covers some of the key elements of a PathMATE system.
- Experiment Control - This is one of the more complex PathMATE samples, showing more domains of greater complexity, realized types, and a range of other constructs. This example also demonstrates different deployment configurations, single process, multiple processes, and multiple processes and tasks.

## 5. Current Limitations

- There are no buttons in Rhapsody to trigger PathMATE transformations. All PathMATE features must be run from an Eclipse installation with PathMATE.
- Spotlight, PathMATE's model debugger, is currently not integrated with Rhapsody.
- Leaving the "Basic Type" field of a typedef blank will cause the transformation process to fail without any errors or explanation.
- State machine diagrams, use case diagrams, and sequence diagrams are all created in the Support Diagram section.
- Ports are not supported
- The activity diagram features "Fork" and "Sync" are not supported.
- Custom PathMATE fields, such as "Profile", "ProfileParameter", and "ImplementationName", are not available for operation parameters. This is a limitation of Rhapsody - there is no way to apply a stereotype to an operation parameter. A simple workaround is to use external markings (via properties.txt).

## Appendix A – PathMATE Properties

<b>Item</b>	<b>Rhapsody Location</b>	<b>RSM Location</b>	<b>Type</b>
<b>Activity</b>			
Action	"Implementation" field	PathMATE Profile	String
<b>AssociationClass</b>			
Name <sup>(1)</sup>	PathMATE Profile	PathMATE Profile	String
<b>Attribute</b>			
Derived	PathMATE Profile	"Is Derived" field	Boolean
ExplicitDelete	PathMATE Profile	PathMATE Profile	Boolean
Exposure	PathMATE Profile	"Is Read Only" field	Boolean
ImplementationName	PathMATE Profile	PathMATE Profile	String
IsComposite	PathMATE Profile	"Is Composite" field	Boolean
Profile	PathMATE Profile	PathMATE Profile	String
SortKey	PathMATE Profile	PathMATE Profile	String
SortOrder	PathMATE Profile	PathMATE Profile	Enum<SortOrder>
<b>Class</b>			
ImplementationName	PathMATE Profile	PathMATE Profile	String
MaxIndex	PathMATE Profile	PathMATE Profile	String
ParseActions	PathMATE Profile	PathMATE Profile	Boolean
Profile	PathMATE Profile	PathMATE Profile	String
SortKey	PathMATE Profile	PathMATE Profile	String
SortOrder	PathMATE Profile	PathMATE Profile	Enum<SortOrder>
<b>Constraint</b>			
Action	"Guard" field	PathMATE Profile	String
<b>Domain</b>			
Analyzed <sup>(2)</sup>	"CG.Package. UseAsExternal" Property	PathMATE Profile	Boolean
ImplementationName	PathMATE Profile	PathMATE Profile	String
Initialization	"CG.Package. AdditionalInitialization " Property		
ParseActions	PathMATE Profile	PathMATE Profile	Boolean
SpotlightEnabled	PathMATE Profile	PathMATE Profile	Boolean
<b>Enumeration</b>			
ImplementationName		PathMATE Profile	String
<b>EnumerationLiteral</b>			
ImplementationName		PathMATE Profile	String
<b>Parameter</b>			
ImplementationName	PathMATE Profile	PathMATE Profile	String
Profile	PathMATE Profile	PathMATE Profile	String
ProfileParameter	PathMATE Profile	PathMATE Profile	Boolean
<b>PrimitiveType</b>			
BaseType	Typedef "Basic type" field	PathMATE Profile	Enum<PrimitiveTypes>

External	"CG.Type. UseAsExternal" field	PathMATE Profile	Boolean
ExternalType		PathMATE Profile	String
ImplementationName		PathMATE Profile	String
<b>Service</b> <sup>(3)</sup>			
Action	"Implementation" field	PathMATE Profile	String
Concurrency	PathMATE Profile	Eclipse UML2	Enum<Concurrency>
Derived	PathMATE Profile	PathMATE Profile	Boolean
ImplementationName	PathMATE Profile	PathMATE Profile	String
Inline	"Inline" field	PathMATE Profile	Boolean
Polymorphism	PathMATE Profile	PathMATE Profile	Enum<Polymorphism>
Profile	PathMATE Profile	PathMATE Profile	String
<b>State</b>			
DeferredEvents <sup>(4)</sup>	PathMATE Profile	PathMATE Profile	String
IgnoredEvents <sup>(4)</sup>	PathMATE Profile	PathMATE Profile	String
ImplementationName		PathMATE Profile	String
<b>System</b> <sup>(6)</sup>			
Automatic Implementation Name	PathMATE Profile	PathMATE Profile	Boolean
IncludeProfiles <sup>(5)</sup>	PathMATE Profile	PathMATE Profile	String
Initialization	PathMATE Profile		
ParseActions	PathMATE Profile	PathMATE Profile	Boolean
<b>Transition</b>			
GuardAction	"Guard" field	PathMATE Profile	String
TransitionAction	"Action" field	PathMATE Profile	String

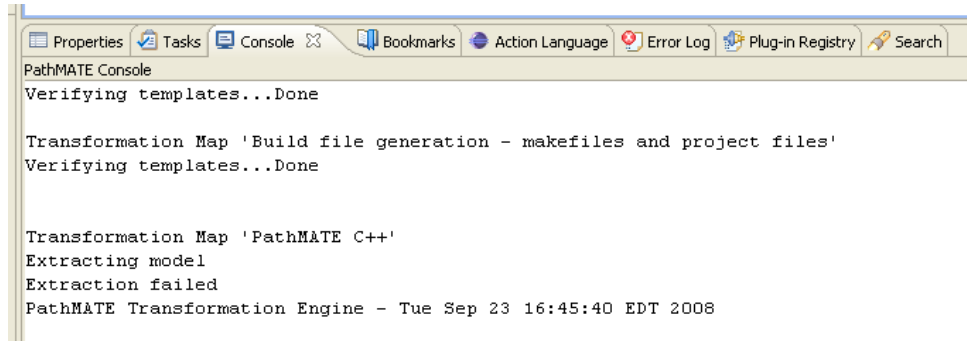
- 1) In Rose, the association class is a separate class from the association. In Rhapsody, the association type is set to "Association Class". In Rose, the Name property of the PathMATE Profile refers to the association number (i.e. A1). In Rhapsody, the Name property refers to the name of the association class to create.
- 2) Set the "UseAsExternal" Rhapsody property to TRUE for Domains that are NOT analyzed (those that are external), FALSE for analyzed domains (those that are modeled).
- 3) Services are called "Operations" in Rhapsody
- 4) These values are ";" delimited lists of Events. The events named here must include the owning class. (i.e. "Class:Event" not "Event")
- 5) A ";" delimited list of user defined profiles to import.
- 6) In Rhapsody, the "System" stereotype is applied to the project, which is the top level model element.

## Appendix B – PathMATE Types Mapped to Rhapsody Types

PathMATE Type Name	Possible Type Selections (case insensitive)	Location
Integer	int	Rhapsody Primitive, Declaration text field
	RhpInteger	Rhapsody Built-in, Declaration text field
	Integer	Declaration text field
Real	RhpReal	Rhapsody Built-in, Declaration text field
	Real	PathMATE Profile, Declaration text field
String	String	Rhapsody Primitive, Declaration text field
	RhpString	Rhapsody Built-in, Declaration text field
Boolean	bool	Rhapsody Primitive, Declaration text field
	RhpBoolean	Rhapsody Built-in, Declaration text field
	OMBoolean	Rhapsody Built-in, Declaration text field
	Boolean	Declaration text field
Void	void	Rhapsody Primitive, Declaration text field

All other PathMATE specific types are available in the PathMATE Profile, or by using the declaration text field. Note that types listed here will be case insensitive when entered in the declaration text field, but types not listed here will be case sensitive.

## Appendix C – Error Messages



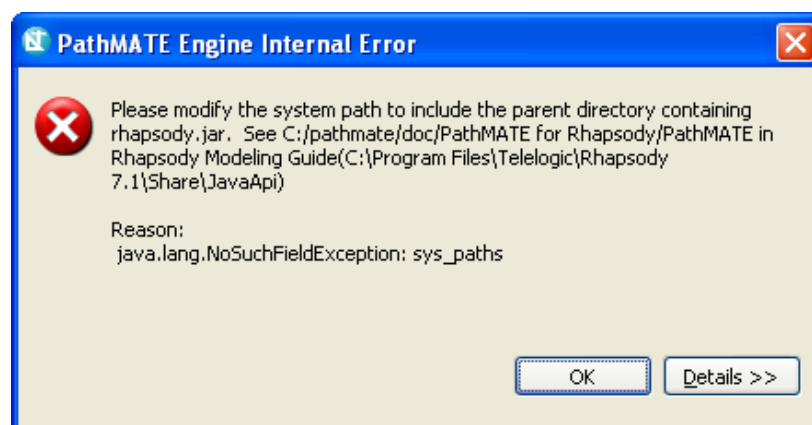
**Figure C-0-1 PathMATE Console Error**

### 1. Extraction Fails Immediately

The reason that for this error message is usually related to problems with the current Rhapsody or PathMATE installation. These errors can result in the improper loading of the Pathfinder plug-ins. Follow these tasks to search for the cause of the failure.

**Task 1. Ensure that *rhapsody.jar* and *rhapsody.dll* exist.**

1. In a windows explorer navigate to you installation of Rhapsody (i.e. C:/Program Files/Telelogic/Rhapsody 7.1).
2. Locate the Share/JavaAPI folder
3. Ensure that both the Rhapsody dll and jar files are present. If not, fix your installation of Rhapsody.

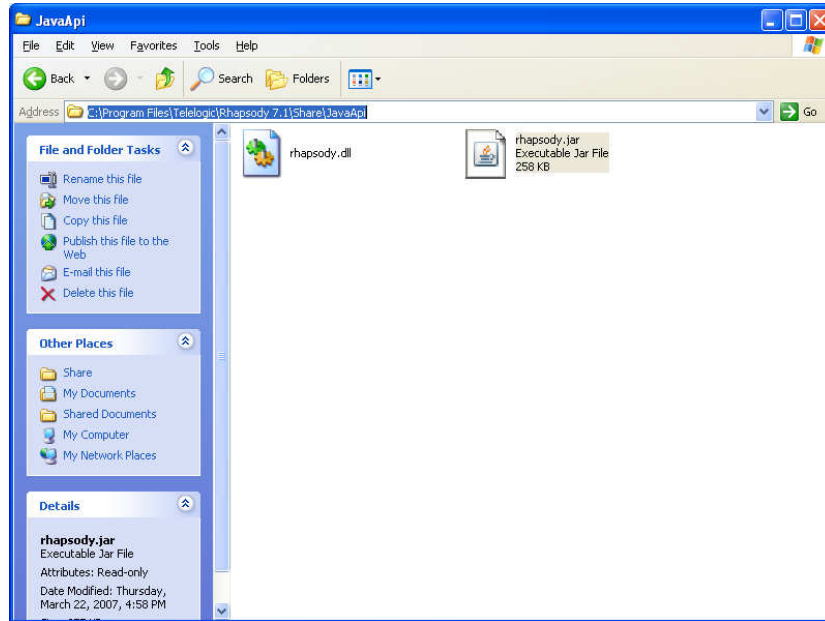


**Figure C-0-2 Example PathMATE Engine Error Prompt**

### 2. Prompted to modify the system path to include the parent directory containing *rhapsody.jar*.

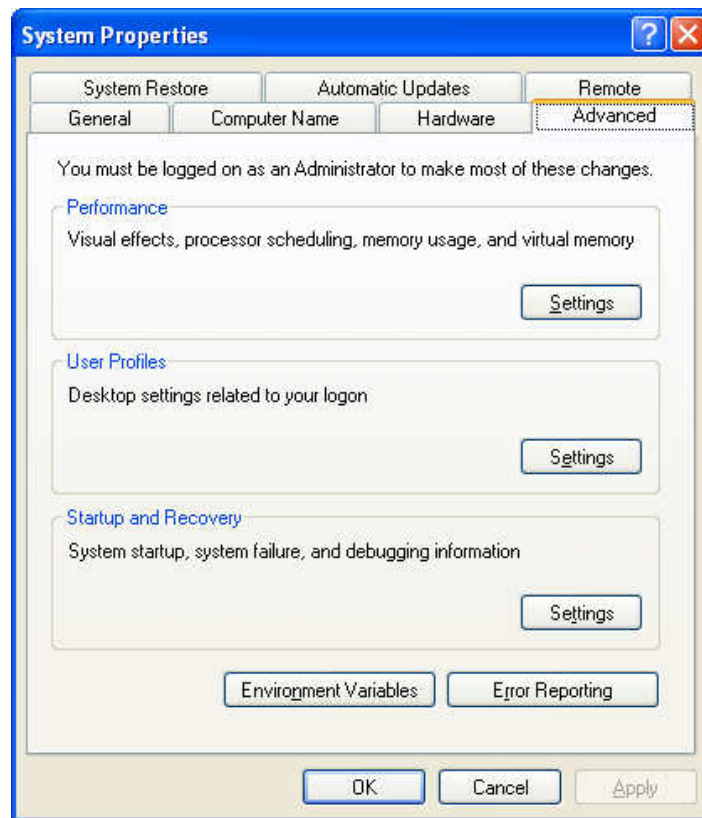
The reason that for this error message is that Rhapsody's Java API directory needs to be added to the System class path, so that Eclipse can properly communicate with Rhapsody. To do so follow these steps:

1. In a windows explorer navigate to you installation of Rhapsody (i.e. C:/Program Files/Telelogic/Rhapsody 7.1).
2. Locate the Share/JavaAPI folder
3. Highlight this folder's path in the address bar and press Ctrl-C



**Figure C-0-3 Example of Share/JavaAPI Folder**

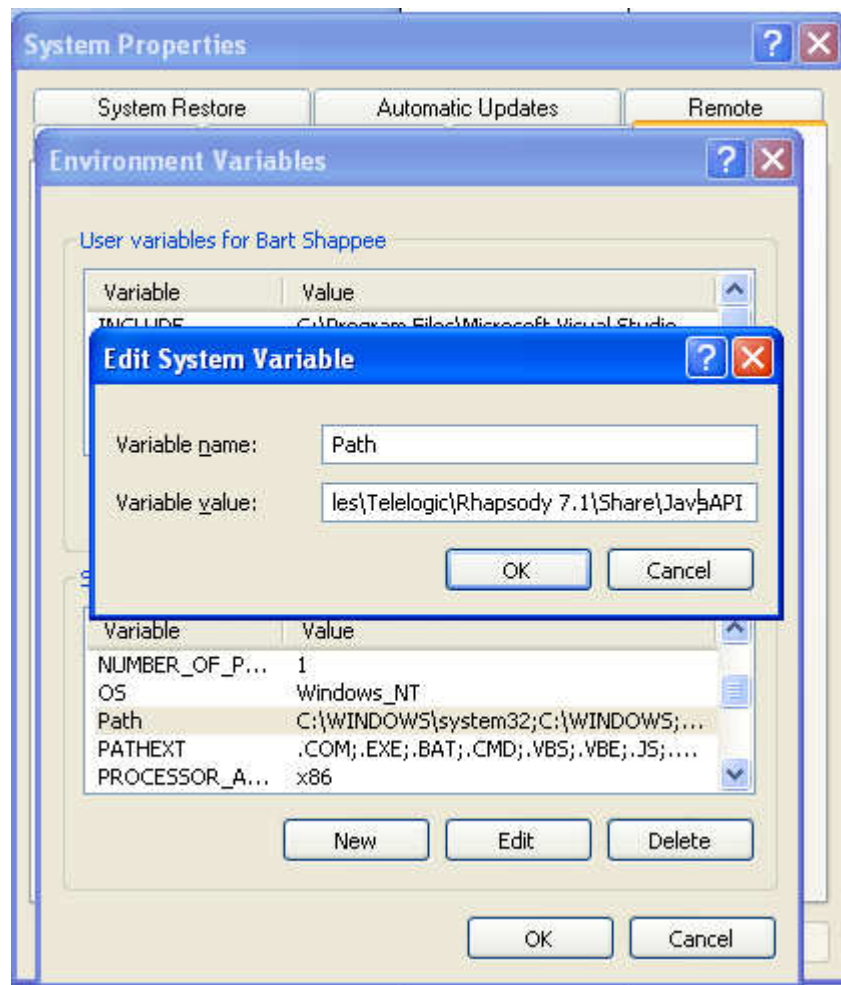
4. Now navigate to "My Computer"
5. Right click in an open area and select properties to bring up the "System Properties" dialog.
6. Click on the advanced Tab



**Figure C-0-4 System Properties**

7. Press the "Environment Variables" button to open the "Environment Variables" dialog.
8. Highlight the "Path" variable in the "System Variables" section.
9. Press the "Edit" to open the "Edit System Variable" dialog.
10. Go to the end of the "Variable Value" field.
11. Add ;
12. Press Ctrl-V to add the Rhapsody Java API directory to your system class path.



**Figure C-0-5 Set Environment Variable**

13. Press OK.
14. Press OK.
15. Press OK.
16. Restart Eclipse and Transform.