



# INTEGRATIONS

with Eclipse, CVS, AndroMDA, and oAW

18.1

user guide

No Magic, Inc.  
2015

All material contained herein is considered proprietary information owned by No Magic, Inc. and is not to be shared, copied, or reproduced by any means. All information copyright 1998-2015 by No Magic, Inc. All Rights Reserved.

# CONTENTS

---

## INTEGRATION WITH ECLIPSE/WSAD/RAD 5

### Installation instructions 5

Automatic MagicDraw installation into Eclipse 5

Manual MagicDraw installation into Eclipse 7

### MagicDraw and Eclipse integration functionality 8

Opening MagicDraw project from Eclipse 8

Integration options 9

Creating not integrated MagicDraw project from Eclipse 11

Creating MagicDraw project from template 12

Opening MagicDraw project 14

Integrating MagicDraw projects with Eclipse projects 15

Unintegrate MagicDraw and Eclipse projects 15

Defining MagicDraw activity in Eclipse 15

Customizing MagicDraw environment 16

Working with integrated Eclipse and MagicDraw project 19

Menu system for UML modeling in Eclipse 23

MagicDraw shortcut keys in Eclipse 24

Eclipse integration and Teamwork 25

Libraries visualization 26

### Known Eclipse problems 26

## INTEGRATION WITH CVS 27

Getting Started 27

CVS properties 27

Checkout module 28

Add a Project to CVS 29

Commit project to CVS 30

Update project 31

## INTEGRATION WITH ANDROMDA 32

### Installation instructions 32

System Requirements 32

Automatic MagicDraw installation into AndroMDA 32

### Working with profiles 34

Global modules paths 34

UML\_Standard\_Profile-3.2.xml 35

### EMF export 35

How and when EMF export is used 35

Synchronization options 35

EMF Export Location 36

Command line converter 36

Maven 2 plugin 37

### External tools 37

### Customization of AndroMDA profiles/cartridges 38

# CONTENTS

---

AndroMDA Diagram	38
Predefined semantics	39
Converting Class Diagrams to AndroMDA Diagrams	39

## INTEGRATION WITH OAW 40

Introduction 40

Installation 40

System Requirements 41

Installation Process 41

Uninstall 42

Features 43

oAW Metamodeling Diagram 43

Workflow Component for MagicDraw EMF Export 43

# INTEGRATION WITH ECLIPSE/WSAD/RAD



As of MagicDraw version 16.5, MagicDraw Professional, Architect, and Enterprise editions can be integrated with Eclipse Workbench. As of version 16.8, MagicDraw Reader edition can be integrated with Eclipse Workbench as well.

Integrating with Eclipse Java IDE (JDT) is available only in Standard, Professional Java, and Enterprise editions as it was before.

As of version 18.0, MagicDraw UI and Eclipse UI integration is no longer supported on Mac OS.

MagicDraw is integrated into the Eclipse environment as a module<sup>1</sup> which provides UML modeling using a current Eclipse project. Common MagicDraw functionality is added as well. This integration enables automatic synchronization of your model in MagicDraw with your code in Eclipse in the same environment. MagicDraw uses the same windowing style as eclipse. The integration provides all of the MagicDraw functionality along with UML data update according to source code.

## Installation instructions

### Automatic MagicDraw installation into Eclipse

Choose one of the following ways to start MagicDraw UML Integration Tool:

- [From the MagicDraw Startup dialog, when starting MagicDraw for the first time](#)
- [From the main menu](#)
- [By running a specific file](#)

To start the integration tool when starting MagicDraw for the first time

---

1. Click the **Integrate** button in the **MagicDraw Startup** dialog. The **Integration** dialog opens (see the following figure).
2. Select Eclipse and click the **Integrate/Unintegrate** button.

To start the integration tool from the main menu

---

1. On the main menu, click **Tools > Integrations**. The **Integration** dialog opens (see the following figure).

---

1. Starting from version 18.1, referred as "used project" in MagicDraw UI. This user guide mentions the old keyword, which will be replaced in the documentation of the next MagicDraw version.

2. Select Eclipse and click the **Integrate/Unintegrate** button.

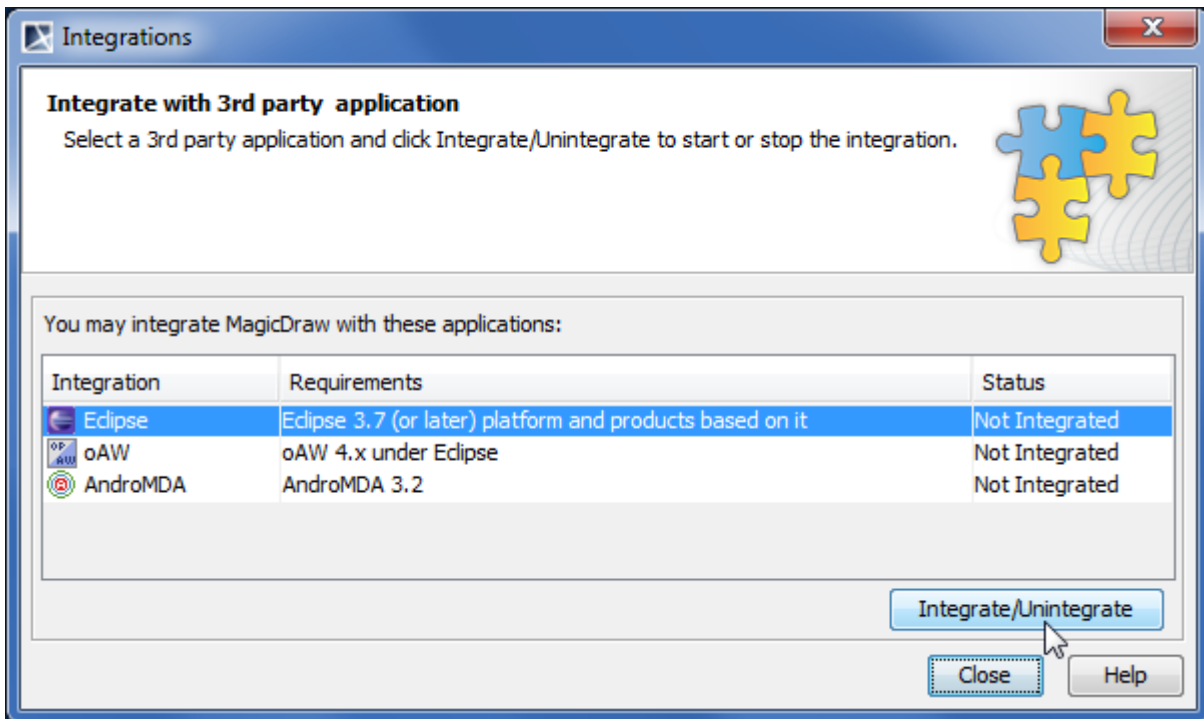


Figure 1 -- Integrations dialog

To start the integration tool by running a specific file

1. Go to <MagicDraw installation directory>/integrations/eclipse.
2. Double-click *install.exe* for Windows OS or *install* for Unix-like OS.

In all three cases a dialog for MagicDraw integration with Eclipse opens.

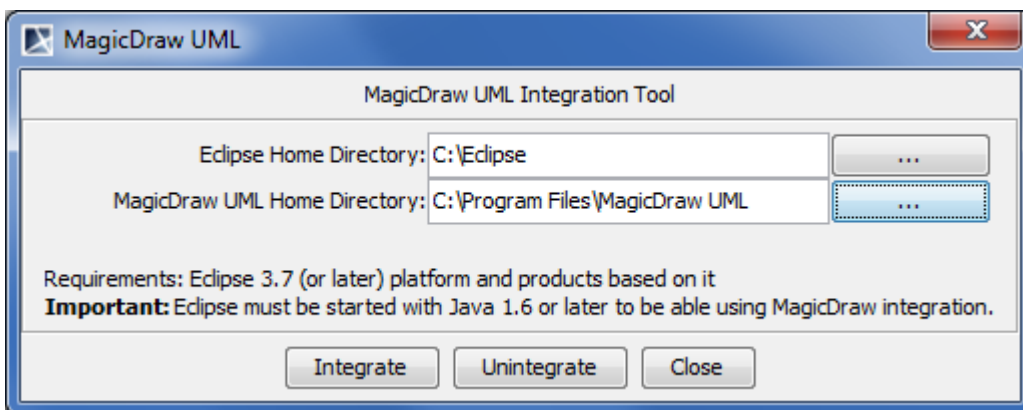


Figure 2 -- MagicDraw UML Integration Tool

If you do not need to change either directory, click the **Integrate** button in the dialog.

To change the Eclipse or MagicDraw home directories

1. Click the ... button in the dialog for MagicDraw integration with Eclipse. The **Eclipse Home Directory** or **MagicDraw Home Directory** dialog appears.
2. Select the installation directories and click **Open**.
3. Click the **Integrate** button in the dialog for MagicDraw integration with Eclipse.

- The Message box appears. If the integration process was successful only the **Exit** button is active. Click **Exit**.



- If integration was successful, the MagicDraw main menu appears in the Eclipse environment with integrated MagicDraw menu commands.
- If automatic installation fails please use the manual installation. Instructions for doing this can be found in "[Manual MagicDraw installation into Eclipse](#)" on page 7.

To unintegrate MagicDraw from Eclipse

---

- Click the **Unintegrate** button in the dialog for MagicDraw integration with Eclipse dialog.

## Manual MagicDraw installation into Eclipse

To install MagicDraw into Eclipse manually

---



Be sure that no earlier MagicDraw integration is installed on your computer.

- Go to `<Eclipse installation directory>\links`.
- Create the file `com.nomagic.magicdraw.integration.link` in the folder.
- Open the file for editing and add `<MagicDraw installation directory>\plugins\com.nomagic.magicdraw.eclipseintegrator` as the path property value



```
path=C:\\Program Files\\MagicDraw
UML\\plugins\\com.nomagic.magicdraw.eclipseintegrator
or
path=C:/Program Files/MagicDraw UML/plugins/
com.comagic.magicdraw.eclipseintegrator
```

- Create the file `com.nomagic.magicdraw.plugins.integration.link` in the same folder.
- Open the file for editing and add `<MagicDraw installation directory>\plugins` as the path property value.



```
path=C:\\Program Files\\MagicDraw UML\\plugins
or
path=C:/Program Files/MagicDraw UML/plugins
```



If integration was successful, the MagicDraw main menu will appear in the Eclipse environment with integrated MagicDraw menu commands.

## Manual installation into Eclipse for IBM Rational Application developer

To install MagicDraw into Eclipse for IBM Rational Application manually

---



Be sure that no earlier MagicDraw integration is installed on your computer.

- Copy from `<MagicDraw installation directory>\plugins\com.nomagic.magicdraw.eclipseintegrator` to `<RAD installation directory>\plugins` the following folders:
  - `com.nomagic.magicdraw.rcp`
  - `com.nomagic.magicdraw.jdt`

2. Copy all jar files from <MagicDraw installation directory>\lib into <RAD installation directory>\plugins\com.nomagic.magicdraw.rcp.
3. Copy all subdirectories from <MagicDraw installation directory>\plugins\eclipse\plugins to <RAD installation directory>\plugins.
4. Go to <RAD installation directory>\plugins\com.nomagic.magicdraw.rcp.
5. Open the file *md.properties* for editing.
6. Add MagicDraw installation folder as the *install.root* property value.



```
install.root=C:\\Program Files\\MagicDraw UML
or
install.root=C:/Program Files/MagicDraw UML
```

7. Open *plugin.xml* for edit and specify relative path to the jars.



```
<library name="graphics/freehep-base.jar">
<export name="*" />
</library>
```

## MagicDraw and Eclipse integration functionality

Eclipse contains the UML modeling workspace.

### Opening MagicDraw project from Eclipse

For the newly created Eclipse project, MagicDraw local or Teamwork project can be created:

1. From the Eclipse project shortcut menu, choose **Open MagicDraw Project**.
2. MagicDraw is started. The **New Project Wizard** dialog box opens.

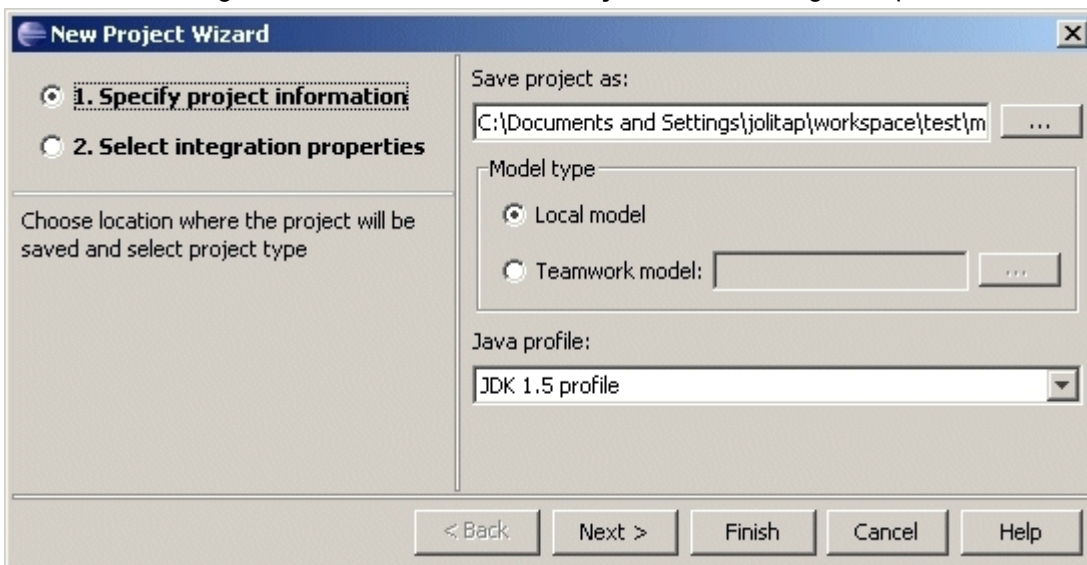


Figure 3 -- New Project Wizard

3. Specify project name and the location of the project.
4. Choose project type: local one or Teamwork. For the Teamwork mode you may select a project from already created models in the Teamwork server.
5. Choose which profile you want to import - JDK 1.5 profile, full JDK 1.4 profile, or only JDK 1.4 java and javax packages.



- Go to the next step, to define integration properties. The detailed description of integration properties you may find in "[Integration options](#)" on page 9.

## Integration options

Specify the MagicDraw and Eclipse integration properties in the **Integration Options** dialog box.

To open the **Integration Options** dialog box

- In the Eclipse perspective, select project and from the shortcut menu, choose **Properties**. Select **MagicDraw** in the opened properties tree and click the **Integration Options** button.
- In the Eclipse perspective, from the **Project** main menu, choose **Properties**. Select **MagicDraw** in the opened properties tree and click the **Integration Options** button.

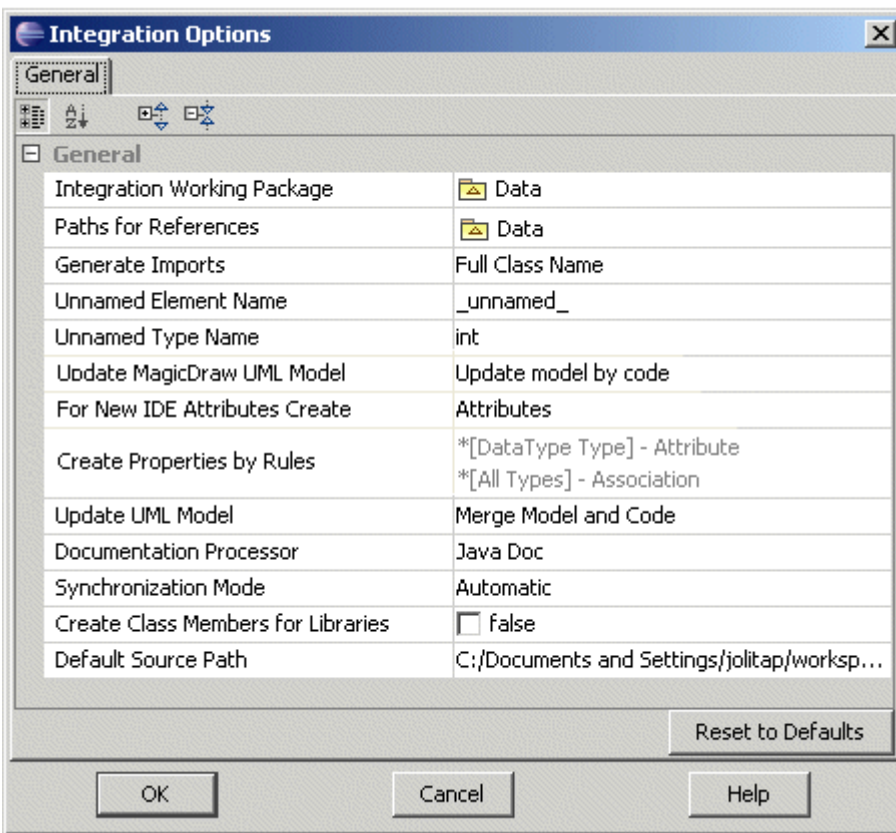


Figure 4 -- Integration Properties dialog box

Command	Option	Function
<b>Integration Working Package</b>		Click the ... button and in the opened dialog box, specify working package where source code will be reversed during update.
<b>Paths for References</b>		To define a path where to search for references (model libraries), click the ... button. The Paths for References dialog box opens.

# INTEGRATION WITH ECLIPSE/WSAD/RAD

## MagicDraw and Eclipse integration functionality

Command	Option	Function
<b>Generate imports</b>	<b>Full Class Name</b>	The import statement with full class name is added during source generation from MagicDraw.
	<b>Package Name</b>	The import statement with package name and * is added during source generation from MagicDraw.
	<b>Do Not Generate</b>	The import statement is not added during source generation from MagicDraw.
<b>Unnamed Element Name</b>	Enter a name of the model element to generate in the source code in instances when a model element within your model does not yet have a name attached to it.	
<b>Unnamed Type Name</b>	Enter a name of the type to generate in the source code in instances when attributes, operations, or parameters do not yet have a type in your model. If an attribute without a type exists in a model, then, after adding the class of this attribute into the Eclipse project, an attribute with <i>int type</i> will be generated in the source code.	
<b>Update MagicDraw UML Model</b>	<b>Update Model by Code</b>	Removes all methods/attributes/associations from the model if they do not exist in the code at the time <b>Update UML Model</b> is executed.
	<b>Merge Model by Code</b>	Leaves all methods/attributes/associations in the model if they do not exist in the code at the time <b>Update UML Model</b> is executed.
<b>For New IDE Attributes Create</b>	<b>Attributes</b>	Creates attributes in MagicDraw for new attributes created in Eclipse.
	<b>Association</b>	Creates associations in MagicDraw for new attributes created in Eclipse.
<b>Create Properties by Rules</b>	Rules defined for the attributes or association creation on reverse. Select this combo box and press the ... button. The Class Field Creation Rules dialog box appears. For more information, see MagicDraw Code&DatabaseEngineering UserGuide.pdf, "Rules of the association or attribute creation on reverse" section.	
<b>Documentation Processor</b>	<b>JavaDoc</b>	When selected, various javadoc tags (@param, @return) are generated in the comments of the code.
	<b>&lt;none&gt;</b>	Documentation from UML model is placed directly into java code with no processing.

Command	Option	Function
<b>Synchronization Mode</b>	<b>Manual</b>	Synchronization between model and code is made manually (by selecting a command).
	<b>Automatic</b> (default)	Every time model is changed, code is updated. Every time code is changed, model is updated (after saving). <b>NOTE:</b> New classes in Eclipse are not added to MagicDraw and vice versa. The user must add them manually.
	<b>Code-to-model automatic</b>	Every time code is changed, model is updated. Changes are not made in code after changing the model. <b>NOTE:</b> New classes in Eclipse are not added to MagicDraw and vice versa. The user must add them manually.
<b>Create class members for libraries</b>	If selected, library classes in model will be created with attributes and methods.	
<b>Default source path</b>	Choose available source paths for all linked projects. You may define source paths in Eclipse.	

## Creating not integrated MagicDraw project from Eclipse

It is possible to create MagicDraw project, which is not bound with Eclipse and can be modeled separately from code.

To create not integrated MagicDraw project

---

1. From the **File** menu or eclipse Java project shortcut menu, choose **New** and then **Other**.

2. In the **New wizard**, expand **MagicDraw** group and select **MagicDraw Project**. Click **Next**.

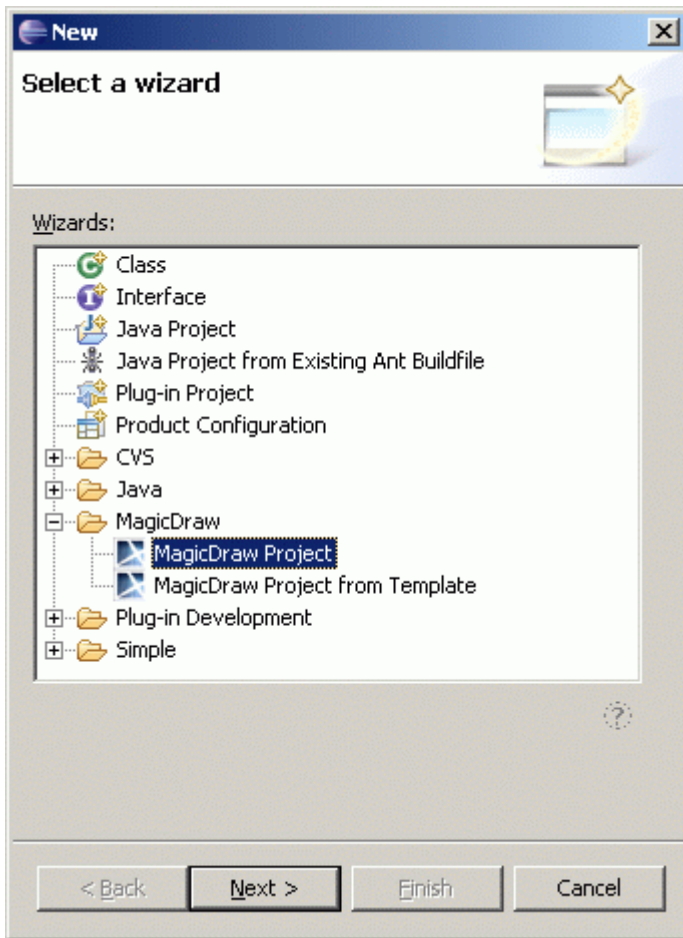


Figure 5 -- New Project wizard, MagicDraw Project

3. Type MagicDraw project name and choose MagicDraw project location. Click **Finish**.



If MagicDraw perspective is set, you may quickly create new MagicDraw project from the **File** main menu by choosing **New** and then **MagicDraw Project**.

## Creating MagicDraw project from template

1. From the **File** menu or Eclipse Java project shortcut menu, choose **New** and then **Other**.

2. In the **New wizard**, expand MagicDraw group and select MagicDraw Project from Template. Click **Next**.

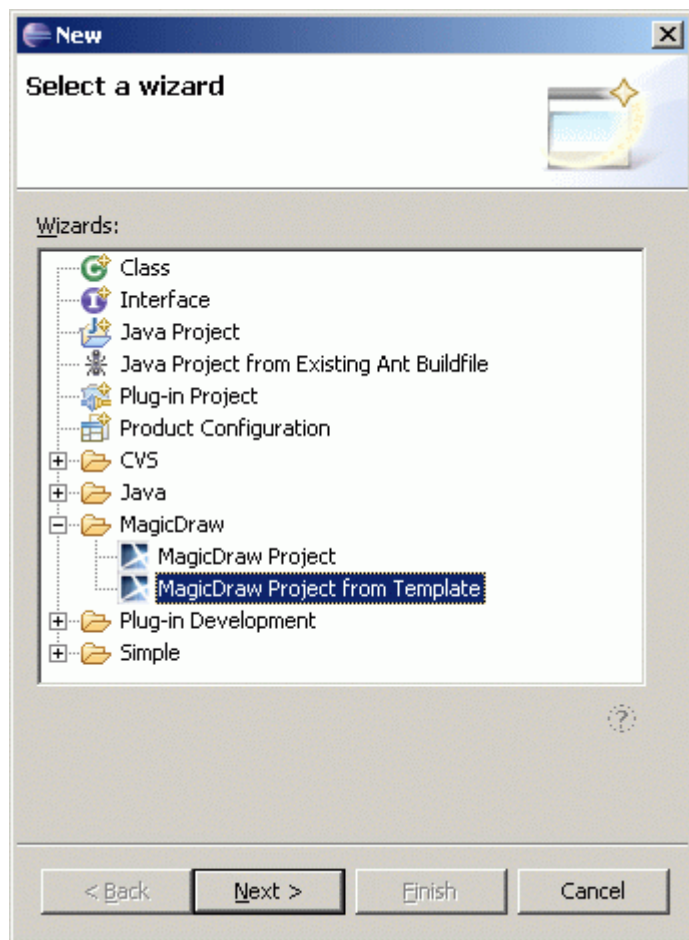


Figure 6 -- New Project wizard, Project from Template

3. In the **New File** wizard, select template.

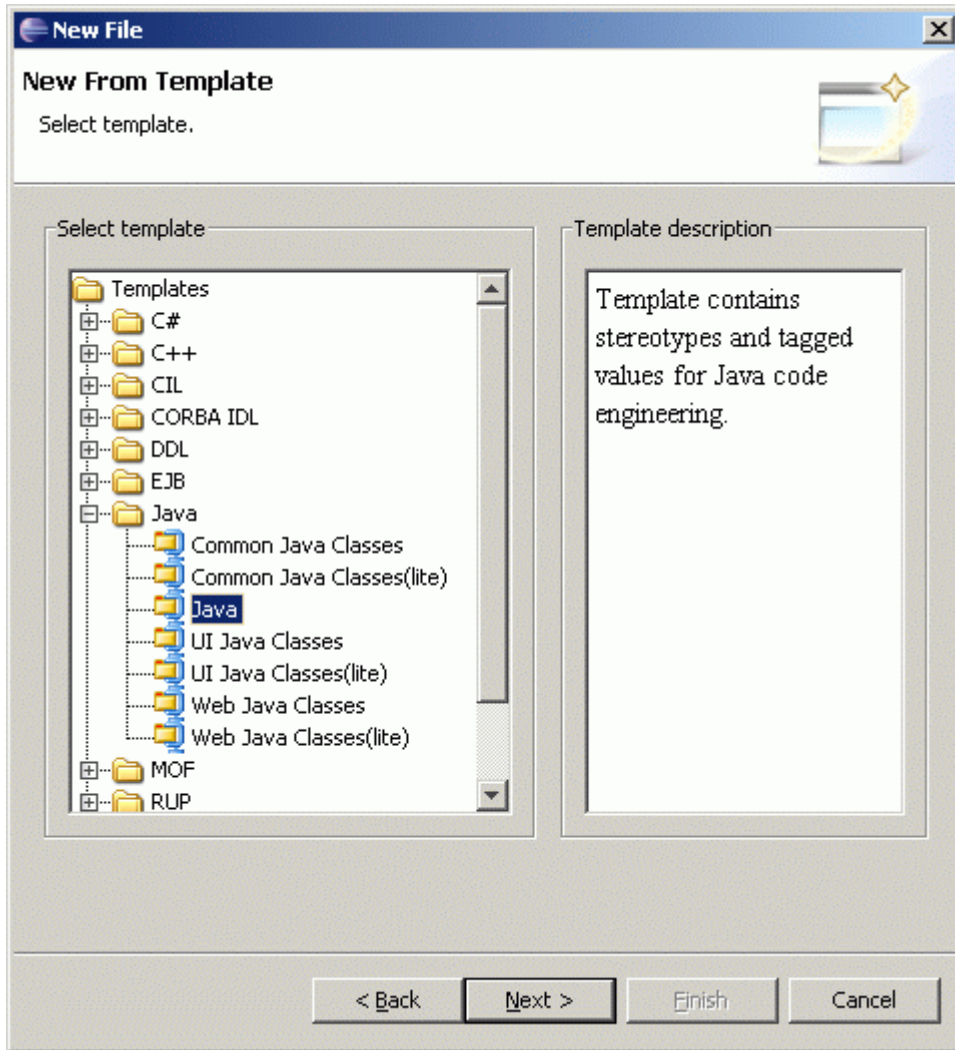


Figure 7 -- New Project wizard, Select Template

4. Click **Next** button to define MagicDraw project title and click **Finish**.

## Opening MagicDraw project

To open already created MagicDraw project, from the **MagicDraw** menu, choose the **Open Project** command.



MagicDraw project is opened as integrated with Eclipse project, if it already was integrated before closing it.

If Eclipse project is already integrated with MagicDraw project, you can quickly open it:

1. Select eclipse Java project in the Package Explorer tree.
2. From the Eclipse Java project shortcut menu, choose the **Open MagicDraw Project** command. Integrated MagicDraw project is opened.



If Eclipse project does not have integrated MagicDraw project, after selecting the **Open MagicDraw Project** command, the **New Project Wizard** is opened.

To close MagicDraw project, from the **MagicDraw** menu, choose the **Close Project** command.

### Integrating MagicDraw projects with Eclipse projects

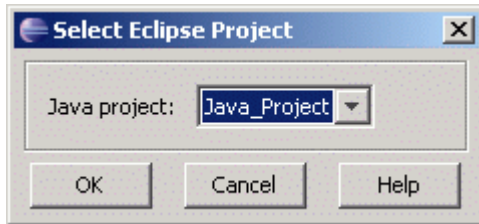


Eclipse projects can have only one integrated MagicDraw project.

To integrate opened MagicDraw project with an Eclipse project

---

1. On the MagicDraw main menu, click **Integrate with Eclipse**. The **Select Eclipse Project** dialog opens.



2. Select an eclipse project and click **OK**. The **New Project Wizard** appears in the **Select Integration Properties** step. Set the integration properties for the current MagicDraw and eclipse projects. Click **Finish**.



For more information about integration properties, see "[Integration options](#)" on page 9.

### Unintegrate MagicDraw and Eclipse projects

To unintegrate MagicDraw project from Eclipse project

---

1. Open integrated MagicDraw project.
2. From the MagicDraw menu, choose the **Unintegrate from Eclipse** command. Projects are unintegrated.

### Defining MagicDraw activity in Eclipse

Activities are Eclipse mechanism to enable/disable large swaths of functionalities. You can enable/disable activities using the workbench preferences.

MagicDraw defines a separate activity for modeling in Eclipse. All the MagicDraw views, editors, perspective, preference and property pages, menus and toolbars, new project wizards are bound to this activity. If this activity is disabled, all the MagicDraw specific plugin contributions are disabled also.

To enable/disable MagicDraw activity

---

1. From the **Windows** main menu, choose **Preferences**. The **Preferences** dialog box is opened.
2. Expand **General** group tree and select **Capabilities** section.

3. Select/clear the **Modeling** check box to enable/disable all specific MagicDraw functionalities. Click **OK**.

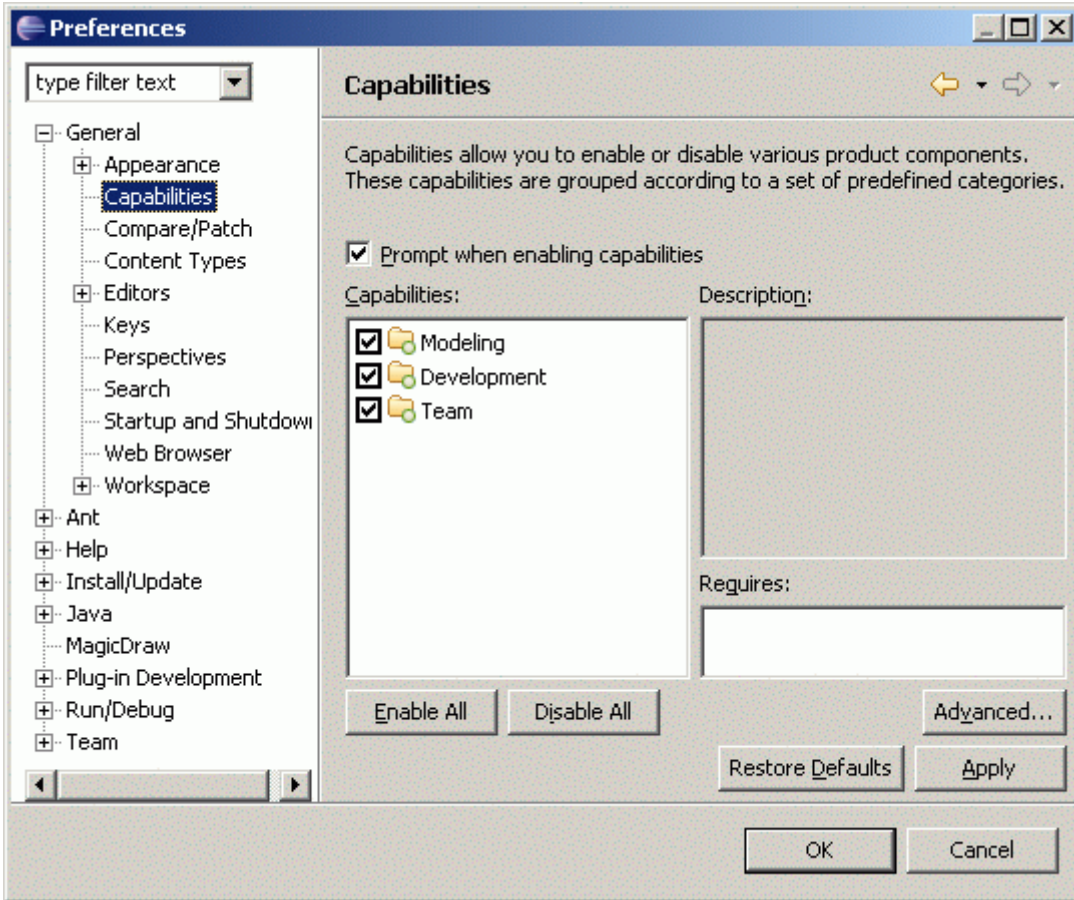



Figure 8 -- Preferences dialog box, Capabilities section

## Customizing MagicDraw environment

There is a possibility to customize separate MagicDraw perspective in Eclipse.

To open MagicDraw perspective

- From the **Window** main menu, choose **Open Perspective** and then **Other**. Select the **MagicDraw** item in the opened **Select Perspective** dialog box.
- Click the **Open Perspective**  button in the right top Eclipse Window near Java perspective icon and then choose **Other**. Select the **MagicDraw** item in the opened **Select Perspective** dialog box. **MagicDraw** perspective icon will be added in toolbar for quick switch between perspectives.



To customize menus for MagicDraw commands

1. From the **Window** main menu, choose **Customize Perspective**.



- Click on the empty space in toolbar and from the shortcut menu, choose **Customize Perspective**. The **Customize Perspective** dialog box opens.
- 2. If the **MagicDraw** check box is selected in the **New** Submenus group, the **MagicDraw Project** and **MagicDraw Project from Template** commands will be added to the **File** main menu, **New** submenu.

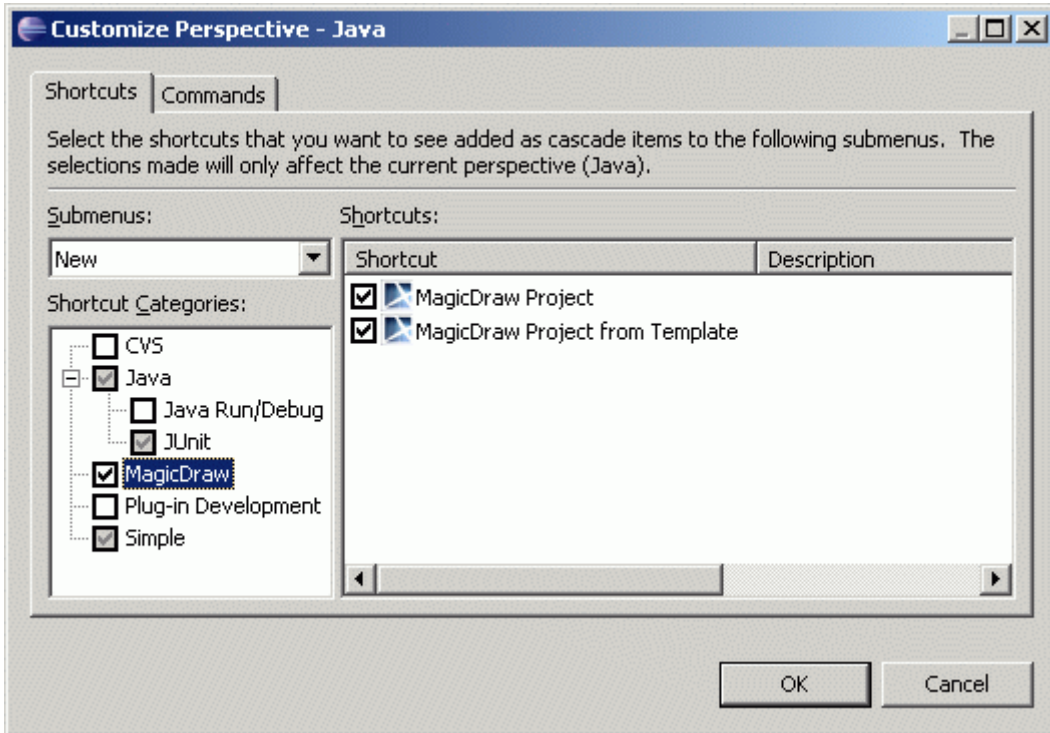


Figure 9 -- Customize Perspective dialog box, New submenu

3. If the **MagicDraw** check box is selected in the **Open Perspective** Submenus group, the **MagicDraw** perspective choice will be added to the **Window** main menu, **Open Perspective** submenu.

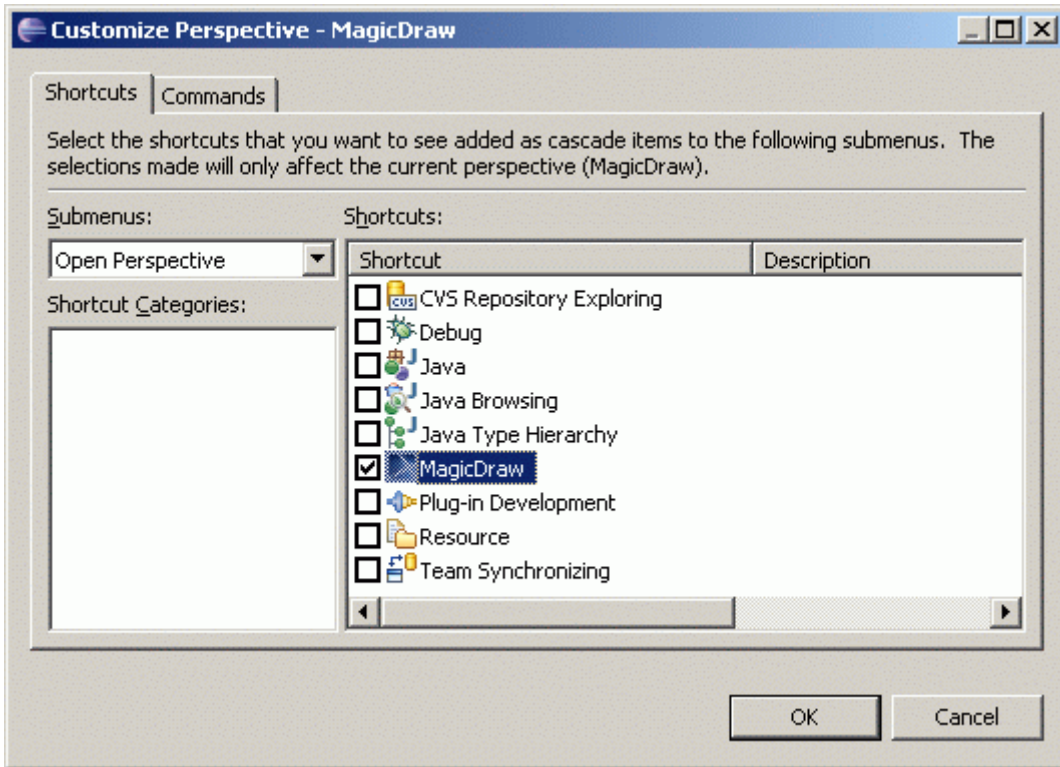


Figure 10 -- Customize Perspective dialog box, Open Perspective submenu

4. If the **MagicDraw UML** check box is selected in the **Show View** Submenus group, listed commands will be added to the **Window** main menu, **Show View** submenu.

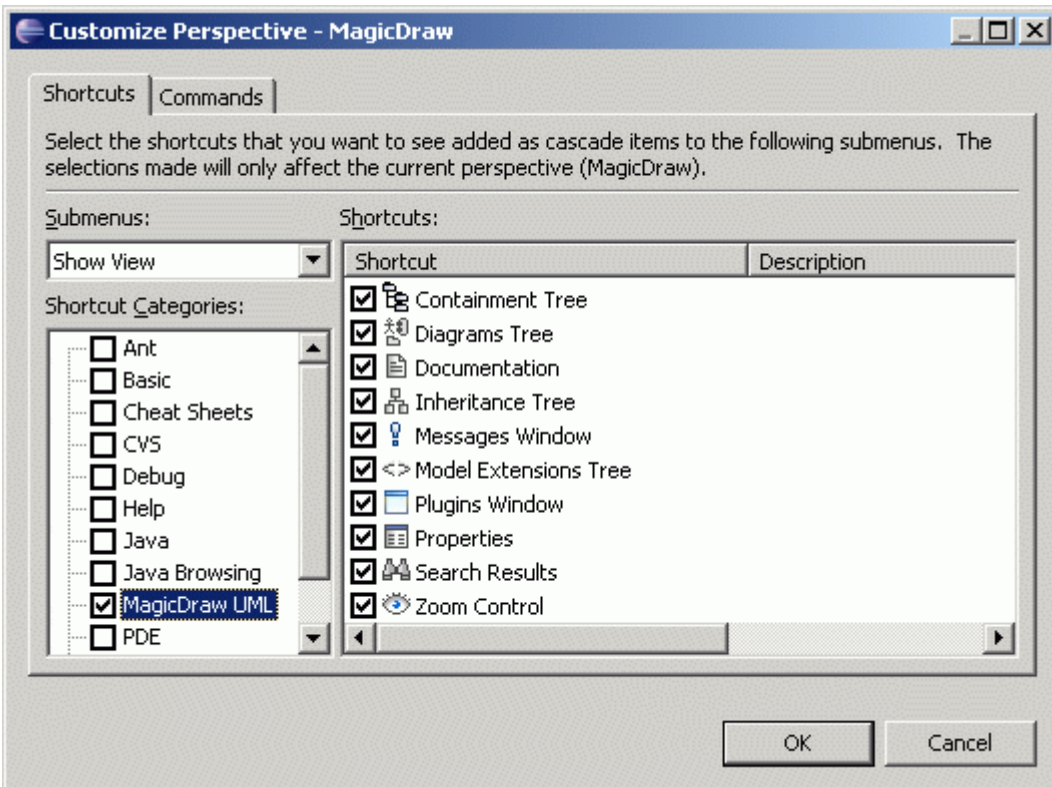


Figure 11 -- Customize Perspective dialog box, Show View submenu

To customize MagicDraw perspective toolbars

1. From the **Window** main menu, choose **Customize Perspective**.
  - Click on the empty space in toolbar and from the shortcut menu, choose **Customize Perspective**. The **Customize Perspective** dialog box opens.
2. Select the **Commands** tab and select the check boxes to add appropriate buttons to toolbar.

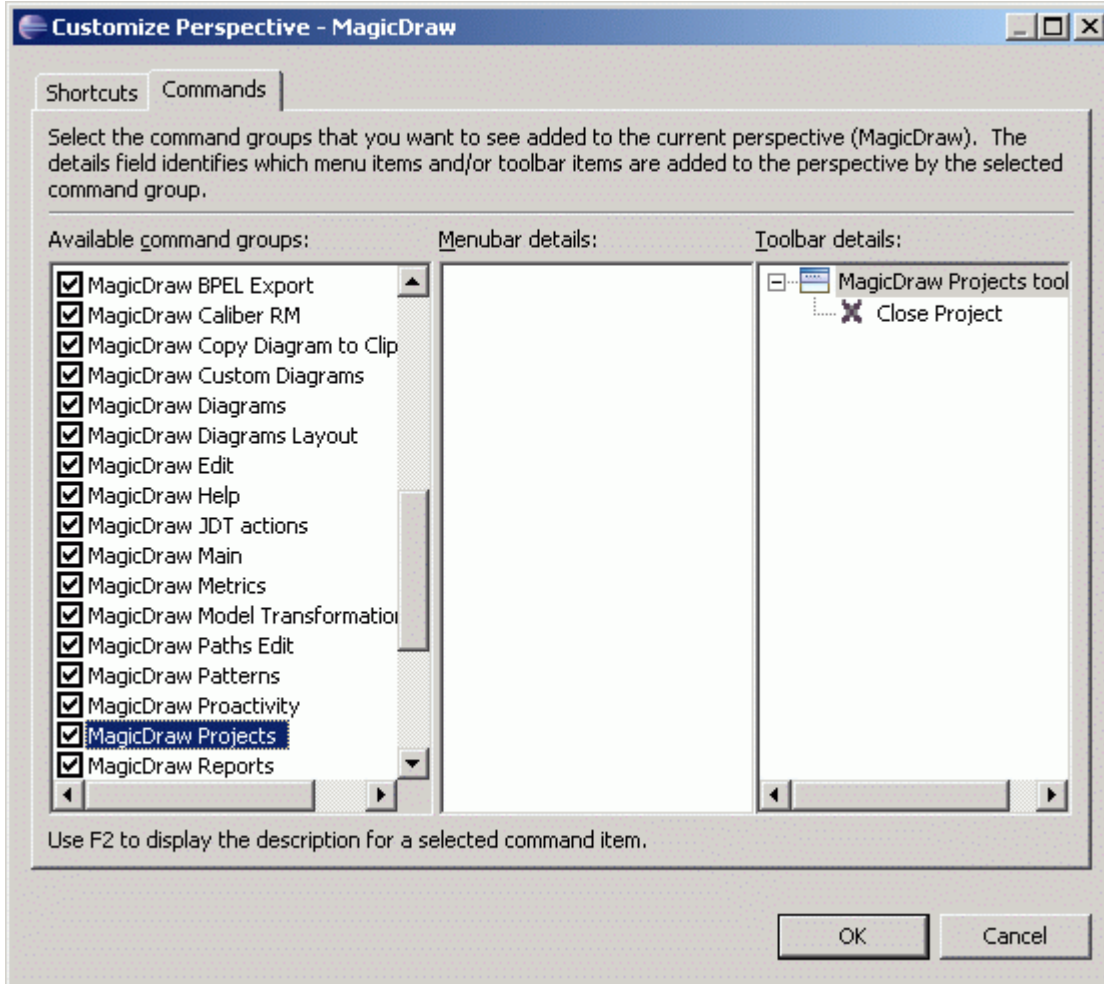


Figure 12 -- Customize Perspective dialog box, Commands tab

To reset perspective to default

From the **Window** main menu, choose **Reset Perspective**. All previously opened tabs or windows will be closed and configuration set to default.

## Working with integrated Eclipse and MagicDraw project

Main actions will be described how to update data between Eclipse and MagicDraw projects.

### Updating the UML model

There are several ways to represent classes created in Eclipse in the MagicDraw project:

- Select a java source file (or several files) in the Eclipse project browser. Open the shortcut menu and select **Update UML Model**:

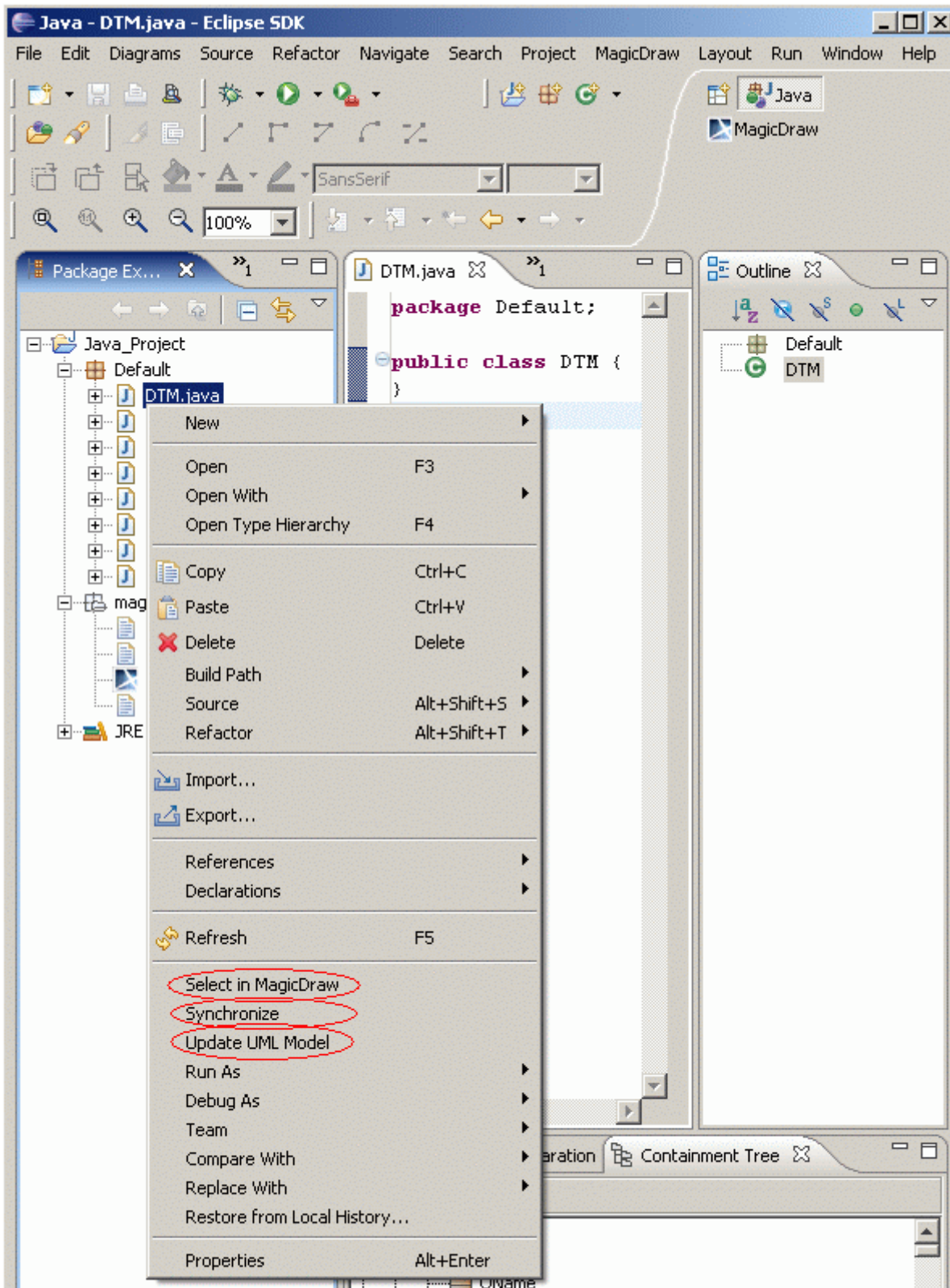


Figure 13 -- Select in MagicDraw and Update UML Model command

- Open the java source in the Eclipse editor window. From the MagicDraw menu, select **Update UML Model**:

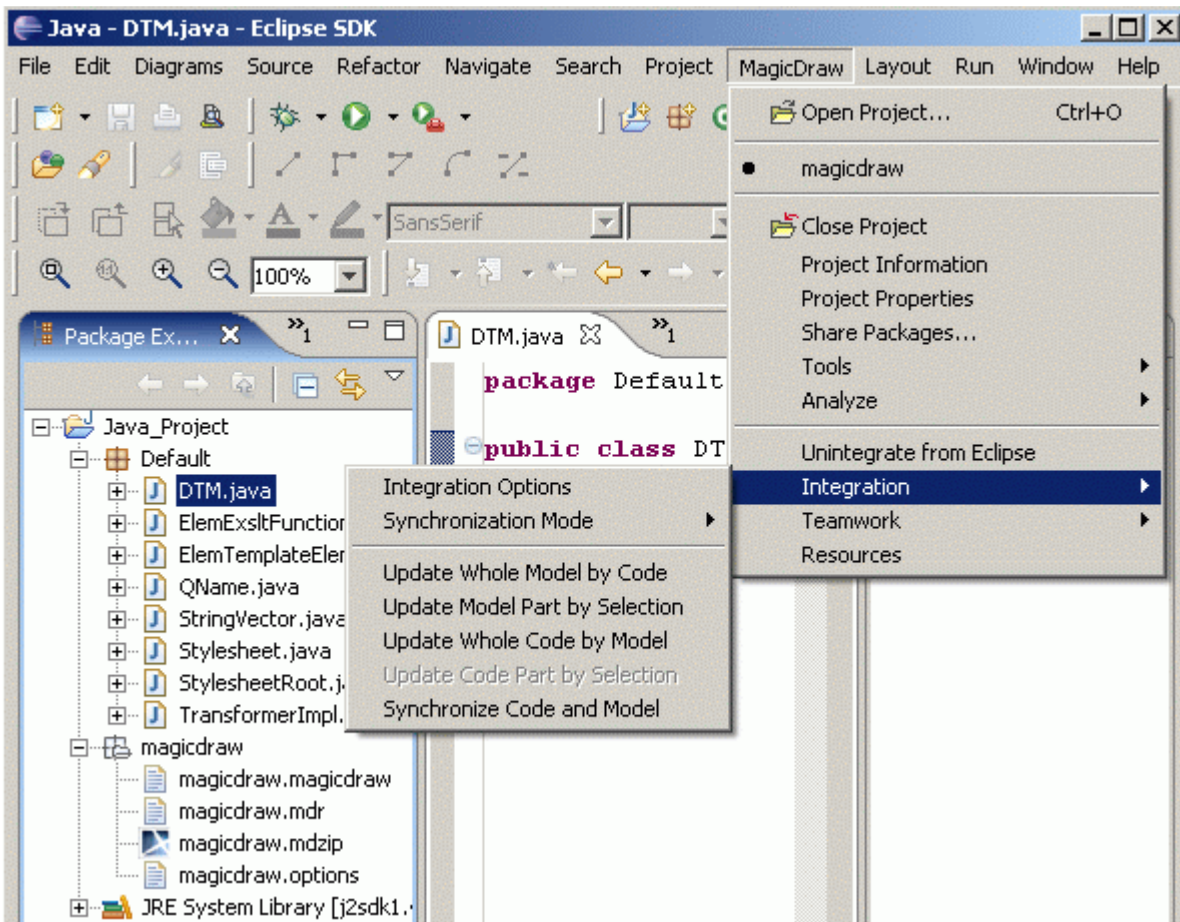


Figure 14 -- Update UML Model command

There are several commands to perform model update from the **Integration** submenu:

- **Update Whole Model by Code** - command will update model by all entire java code, created in this project.
- **Update Model Part by Selection** - command will update only selected code part (for example, class) in model.
- **Synchronize Code and Model** - command will merge all changes - update code according to model and will add model elements, created in code.

Changes made in the java source file are automatically reflected in the MagicDraw project once you have selected Synchronization mode as **Automatic** or **Code to model automatic** in the **Integration Options** dialog box (See "[Integration options](#)" on page 9).

When updating UML model, you may choose one of the following option in the integration **Properties** dialog box:

<b>Update Model by Code</b>	Removes all methods/attributes/associations from the model if they do not exist in the code at the time <b>Update UML Model</b> is executed.
<b>Merge Model and Code</b>	Leaves all methods/attributes/associations in the model if they do not exist in the code at the time <b>Update UML Model</b> is executed.

Beginning with version 9.5, you can choose attributes or associations you want to create in MagicDraw for attributes created in Eclipse. This option you can also define in the **Integration Options** dialog box.

### Adding/updating a class to the Eclipse project

There are several ways to add/update a class to an Eclipse project from integrated MagicDraw project:

- From the class in the MagicDraw browser shortcut menu, select **Update Eclipse**. The class source will be generated and added to the Eclipse project.
- From the package in the MagicDraw browser shortcut menu, select **Update Eclipse**. All classes from this package will be added to the Eclipse project.
- From the selected class or package in any MagicDraw diagram shortcut menu, select **Update Eclipse**.
- Open the **MagicDraw** main menu, **Integration** submenu and select **Update Whole Code by Model**. Code to all created model elements will be generated in Eclipse.
- Select a class in any MagicDraw diagram. Open the **MagicDraw** main menu, **Integration** submenu and select **Update Code Part by Model**. The class source will be generated and added to the Eclipse project.

### Synchronization between MagicDraw and Eclipse

The following synchronization modes are presented:

Synchronization mode	Description
<b>Automatic</b>	Every time model is changed, code is updated. Every time code is changed, model is updated (after saving). <b>NOTE:</b> New classes in Eclipse are not added to MagicDraw and vice versa. The user must add them manually.
<b>Manual</b>	Synchronization between model and code is made manually (by selecting correspondingly <b>Update Eclipse</b> or <b>Update UML Model</b> ).
<b>Code to model automatic</b>	Every time code is changed, model is updated. Changes are not made in code after changing the model. <b>NOTE:</b> New classes in Eclipse are not added to MagicDraw and vice versa. The user must add them manually.

You can define synchronization mode in the **Integration Properties** dialog box or from the **Integration** submenu in the **MagicDraw** menu.

### Selecting a class in the MagicDraw browser

From the Eclipse project browser, choose **Select in MagicDraw**. The class will be selected in the **Containment Tree** tab in Eclipse perspective. If you switch to MagicDraw perspective, the same class will be selected in MagicDraw browser.

### Going to class source from MagicDraw environment

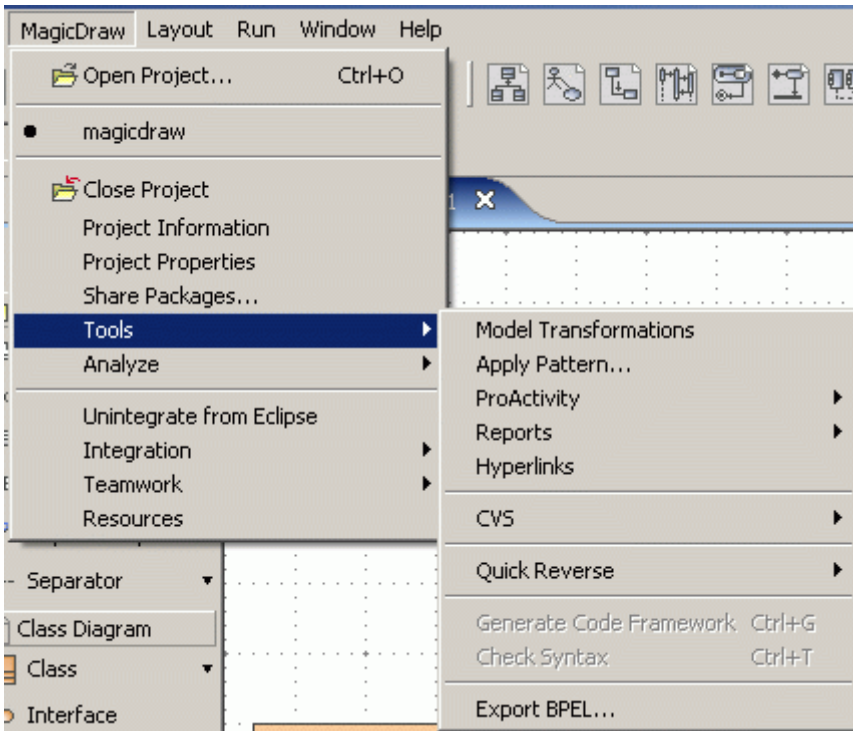
- To open the class/operation/attribute source code in Eclipse, from the **Navigate** menu, **Go to** and then **Open Source** command. This command is enabled just after the class has been added into the Eclipse project.
- From the element in the MagicDraw browser shortcut menu, select **Go to Source in Eclipse**.
- On the MagicDraw diagram pane, select element and from the shortcut menu, choose **Go to Source in Eclipse**.

### Menu system for UML modeling in Eclipse

For a detailed description of the menu system, see the MagicDraw UML User's Manual <MagicDraw installation directory>/Manual folder.

Most MagicDraw menu commands retain the same position as it was in standalone MagicDraw version. If not, you may find them under the **MagicDraw** main menu.

The **Tools**, **Analyze**, **Teamwork** with all submenus, **Shared Packages**, **Resources**, **Project Properties**, **Project Information** commands can be found under **MagicDraw** main menu.



### MagicDraw toolbar

In the MagicDraw perspective, some buttons in the toolbar are merged with actions, performed in Eclipse:



This toolbar contains common actions used while working with MagicDraw.

Button	Function
<b>New</b>	Create new Eclipse or MagicDraw project.
<b>Save</b>	Save Eclipse as well as MagicDraw project on the same action.
<b>Print</b>	Print MagicDraw diagram or Eclipse source code.
<b>Print Preview</b>	Show MagicDraw diagram printing view.

**NEW!** Default MagicDraw perspective toolbar contains the **Create Diagram** button for quick diagrams creation.

To change the toolbar configuration for the MagicDraw perspective, see "[To customize MagicDraw perspective toolbars](#)" on page 19.



For more information about MagicDraw toolbars, see [MagicDraw UserManual.pdf](#), the "Toolbars" section.

### Printing

MagicDraw diagrams printing is merged with Eclipse printing functionality. Active window (java source code or UML diagram) will be printed on the **Print** action from the **File** main menu, clicking on the **Print** button from main toolbar, or pressing **CTRL+P** shortcut key.

From the **File** main menu, the **Print Preview** and **Print Options** dialog boxes have the same functionality as it is in standalone MagicDraw version.



For more information about the **Print Options** dialog box, see *MagicDraw User Manual, Diagram Basics* chapter, *Printing* section.

### Searching

MagicDraw search can be performed only when MagicDraw perspective is switched. Otherwise, Eclipse finding mechanism will be opened.

To open Find dialog box in MagicDraw

---

- From the **Edit** menu, choose the **Find/Replace** command. The **Find** dialog box opens.
- Press **CTRL+F** shortcut key to open the **Find** dialog.



For more information about MagicDraw searching functionality, see *MagicDraw User Manual, Working with Projects* chapter, *Searching* section.

### MagicDraw shortcut keys in Eclipse

Shortcut keys for merged MagicDraw and Eclipse actions are taken from Eclipse. For MagicDraw specific actions, shortcut keys are defined **In MagicDraw** context.

To change shortcut keys

---

1. From the **Windows** main menu, choose **Preferences**. The **Preferences** dialog box is opened.



- Expand **General** group tree and select **Keys** section. Shortcut keys for MagicDraw commands are marked in the **In MagicDraw** context.

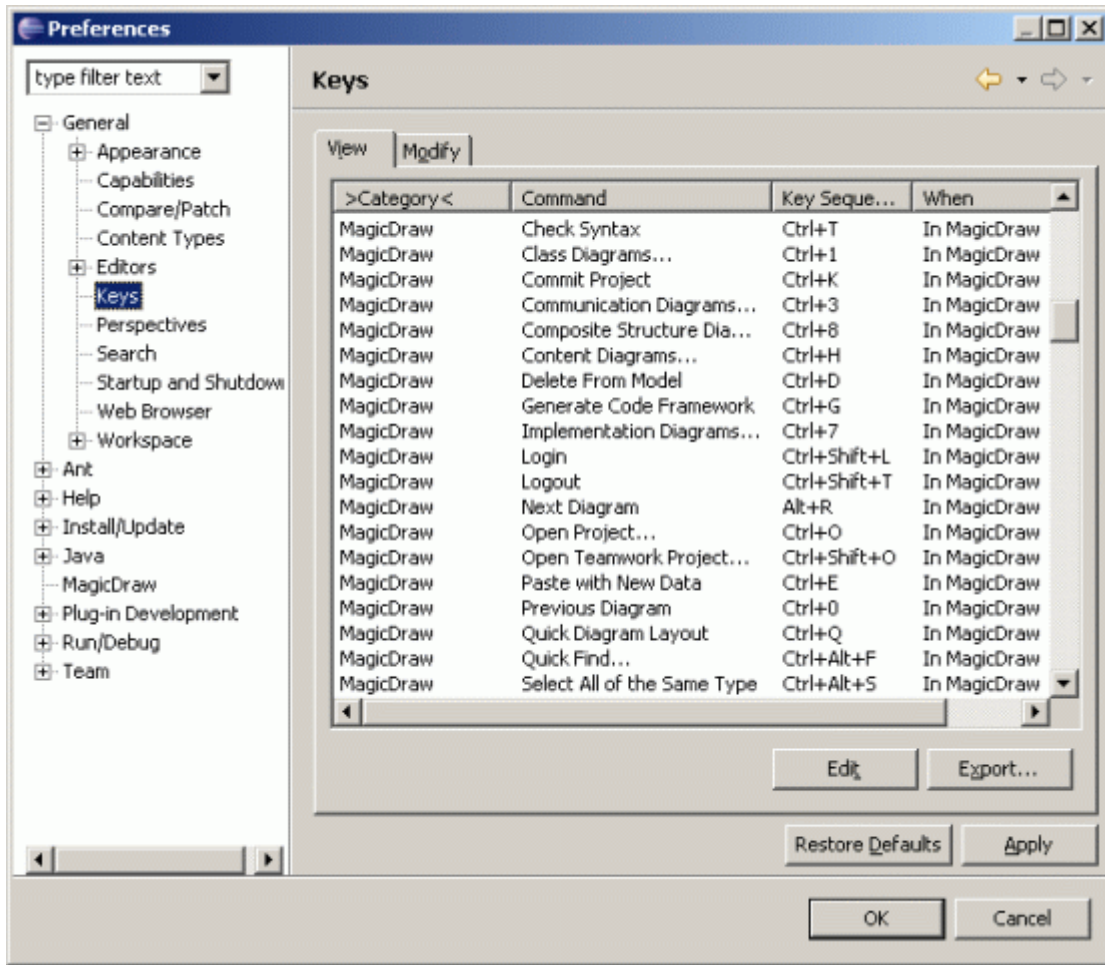


Figure 15 -- Preferences dialog box, Keys section

- Select a category and click **Edit** for changing/adding new shortcut key.

## Eclipse integration and Teamwork

When opening project, created in Eclipse, you may choose to open it in a local MagicDraw project or in a Teamwork project.

To map Eclipse project to a MagicDraw Teamwork project

- Choose a command to open MagicDraw from Eclipse.
- MagicDraw starts up and **New Project Wizard** appears.
- Choose the Teamwork Model option button and then click ... button.
- Log in to the Teamwork server and choose the desired project.
- From Eclipse, update UML model.

To change the mapping from local model to teamwork model

- Connect to the MagicDraw Teamwork server.
- Choose the **Add Project To Teamwork** command.

### Libraries visualization

Beginning with version 9.5, libraries that are used in Eclipse can be referenced in MagicDraw models. On lib, in Eclipse, choose **Update UML Model**. Created model package in MagicDraw will be marked with stereotype <<topLevel>>.

In the Integration **Properties** dialog box, you may choose what you want to create - just classes in model without attributes and methods (default), or together with attributes and methods (option **Create class members for libraries**).

### Known Eclipse problems

- Sometimes change in class source does not update opened editor.
- No way to remove all extended classes or implemented interfaces for some class or interface.
- Class fields defined using ',' are updated incorrectly.
- After unintegration, Eclipse cannot be started with new integrated MagicDraw at once. Old directory is still remembered by Eclipse. As a workaround - unintegrate Eclipse from MagicDraw, then start Eclipse, close it, and only then integrate with new MagicDraw version and start again.
- Shortcut menu in the Browser does not disappear if without closing it any main menu command selection is performed.
- Select All, Copy/Paste and Delete commands are not available in MagicDraw specification dialog boxes.
- Focus in dialog box is lost when going through text boxes using Tab key.
- Sometimes diagrams or Browser tabs are frozen because of modal dialog appearance. The solution is to close and reopen diagram view or Browser tabs in all Eclipse perspectives.

# INTEGRATION WITH CVS



This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

MagicDraw™ integration with CVS is used only for the storage of MagicDraw files in the CVS repository. Since MagicDraw™ has a built-in CVS client, no other application is needed to integrate MagicDraw with CVS.

## Getting Started

1. Define CVS properties in the **CVS** pane of the **Environment Options** dialog box.
2. Make sure that the module you are working with is checked out through MagicDraw or using some other program.



For detailed instructions on how to check out a module in MagicDraw, see "[Checkout module](#)" on page 28.



- You can add, update, or commit projects to CVS only if they are saved in a checked out directory or subdirectory.
- The Command line runs in a local folder, which is specified in the **Environment Options** dialog box. In order for the Command Line to find CVSROOT, the Local Folder must be checked out and must have a CVS subdirectory.

## CVS properties

Define CVS Properties in the **CVS** pane of the **Environment Options** dialog:

1. From the **Options** menu, choose **Environment**. The **Environment Options** dialog appears.
2. Open the **CVS** pane.

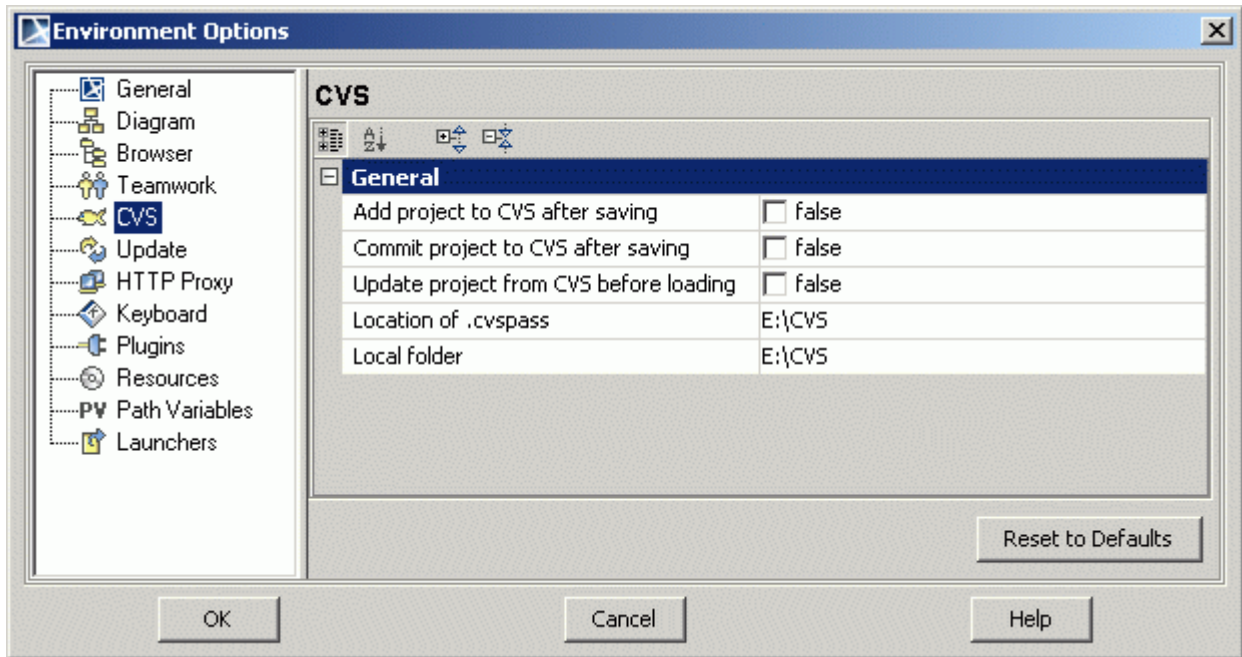


Figure 1 -- Environment Options dialog box. CVS pane

Command	Function
<b>Add Project to CVS After Saving</b>	In every instance where you save, a newly created project is added to CVS in the checked out directory to CVS. The <b>Add Project to CVS</b> dialog box appears.
<b>Commit Project to CVS After Saving</b>	Commits the project to CVS after saving it. The <b>Commit Project to CVS</b> dialog box appears.
<b>Update Project from CVS Before Loading</b>	Updates the project that is added to CVS while loading. The <b>Update Project</b> dialog box appears.
<b>Location of .cvspass</b>	The path where the .cvspass is located. You may enter it here, or choose the path from the <b>Open</b> dialog box.
<b>Local Folder</b>	The path where the module will be saved during the checkout action. You may enter it here, or choose the path from the <b>Open</b> dialog box.

## Checkout module

Use this option to checkout a new module on your disk.

From the **Tools** menu, select **CVS** and then **Checkout Module**. The **Checkout Module** dialog appears.

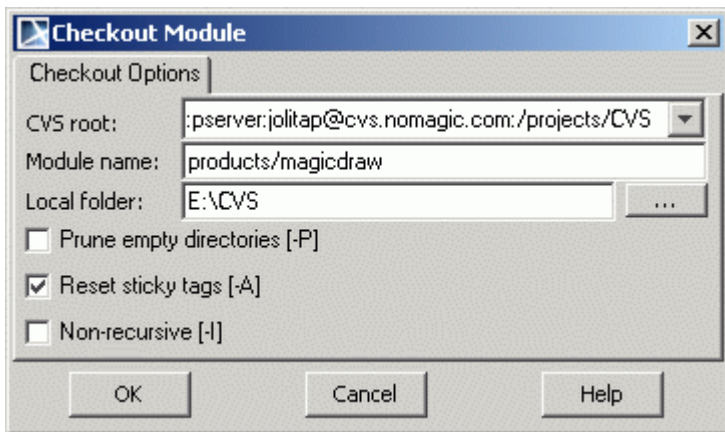


Figure 2 -- Checkout Module dialog box

Command	Function
<b>CVS Root</b>	Shows CVSROOT variable and provides all previously typed CVS Root variables. You must enter the complete path of the module on the server: It should be something like: :pserver:martina@cv.s.nomagic.com:/projects/ CVS
<b>Module name</b>	Enter the module name and path on the server.
<b>Local Folder</b>	The path indicating where the CVS files will be saved locally in the disk. Enter the path, or select any directory by clicking the ... button.
<b>Prune Empty directories [-P]</b>	Automatically removes empty folders when you update a module. <b>NOTE:</b> The sign for the options is listed in the brackets at the end of the check box name.
<b>Reset sticky tags [-A]</b>	Checks out only the last project version.
<b>Non-recursive [-I]</b>	Does not check out subdirectories.

## Add a Project to CVS

1. Create a new MagicDraw project.
2. Save it into the currently checked out directory.
3. From the **Tools** menu, select **CVS** and then select **Add**. The **Add Project to CVS** dialog box appears.



If the **Add Project to CVS After Saving** command in the **Environment Options** dialog box, **CVS** pane is selected. The **Add Project to CVS** dialog box appears every time when new to CVS projects are saved.

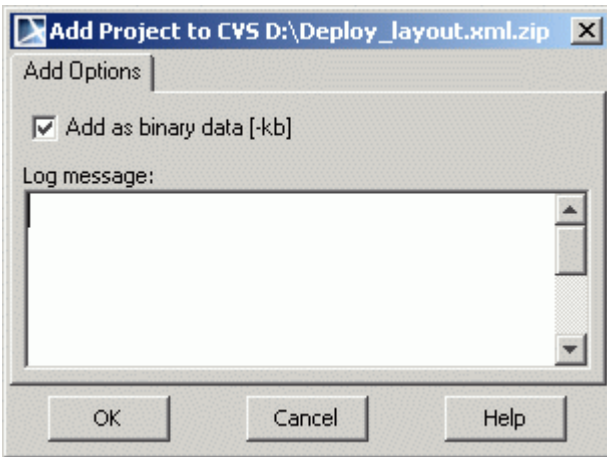


Figure 3 -- Add to CVS Options dialog box

1. Enter a check in the **Add as binary data [-kb]** check box if the project is saved as XML.ZIP format.
2. Type the **Log message** if needed.
3. Click **OK**.

## Commit project to CVS

Commit your project to CVS after making changes to the project.



If **Commit Project to CVS After Saving** is selected (in the **CVS** pane located in the **Environment Options** dialog box), the project is committed to the CVS server every time it is saved.

1. Open your MagicDraw project.
2. From the **Tools** menu, select **CVS**. Then select **Commit Project to CVS**. The **Commit Project to CVS** dialog box appears.

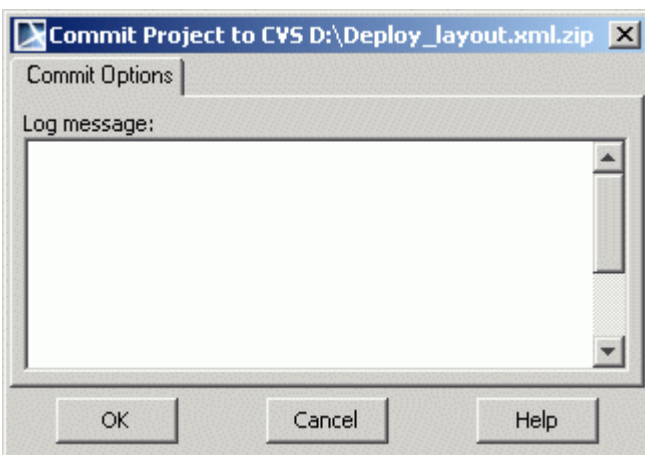


Figure 4 -- Commit Project to CVS dialog box

3. Type the log message if needed.
4. Click **OK**.

## Update project

Update your project when you receive a message indicating a new project version exists on the server.

From the **Tools** menu, choose **CVS** and then **Update CVS Project**. The **Update CVS Project** dialog box appears.

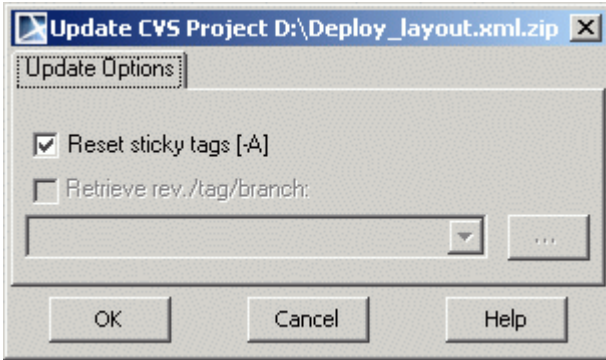


Figure 5 -- Update CVS Project dialog box

Option	Function
<b>Reset sticky tags [-A]</b>	Update to the last version of the project located on the server.
<b>Retrieve rev./tag/branch</b>	Type the tag or version number you want to change. (Enabled, when <b>Reset sticky tags</b> check box is cleared)

You may click the ... button and choose the desired project version from the **Revisions** dialog box.

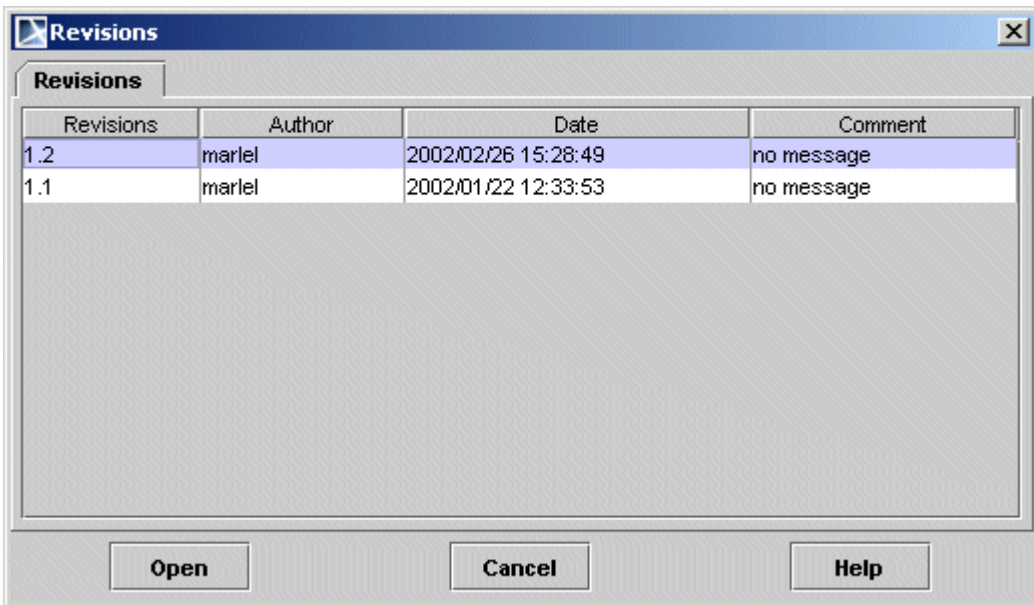


Figure 6 -- Revisions dialog box

# INTEGRATION WITH ANDROMDA



This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

The purpose of this document is to describe MagicDraw and AndroMDA integration features, specific usability improvements and usage scenarios.



Please read <http://galaxy.andromda.org/docs/> for more details about usage of AndroMDA tool itself.

## Installation instructions

### System Requirements

AndroMDA 3.2, Maven.

### Automatic MagicDraw installation into AndroMDA

Choose one of the following ways to start MagicDraw UML Integration Tool:

- [From the MagicDraw Startup dialog, when starting MagicDraw for the first time](#)
- [From the main menu](#)

To start the integration tool when starting MagicDraw for the first time

---

1. Click the **Integrate** button in the **MagicDraw Startup** dialog. The **Integration** dialog opens.
2. Select AndroMDA and click the **Integrate/Unintegrate** button.

To start the integration tool from the main menu

---

1. On the main menu, click **Tools > Integrations**. The **Integration** dialog opens (see the following figure).



2. Select **AndroMDA** and click the **Integrate/Unintegrate** button.

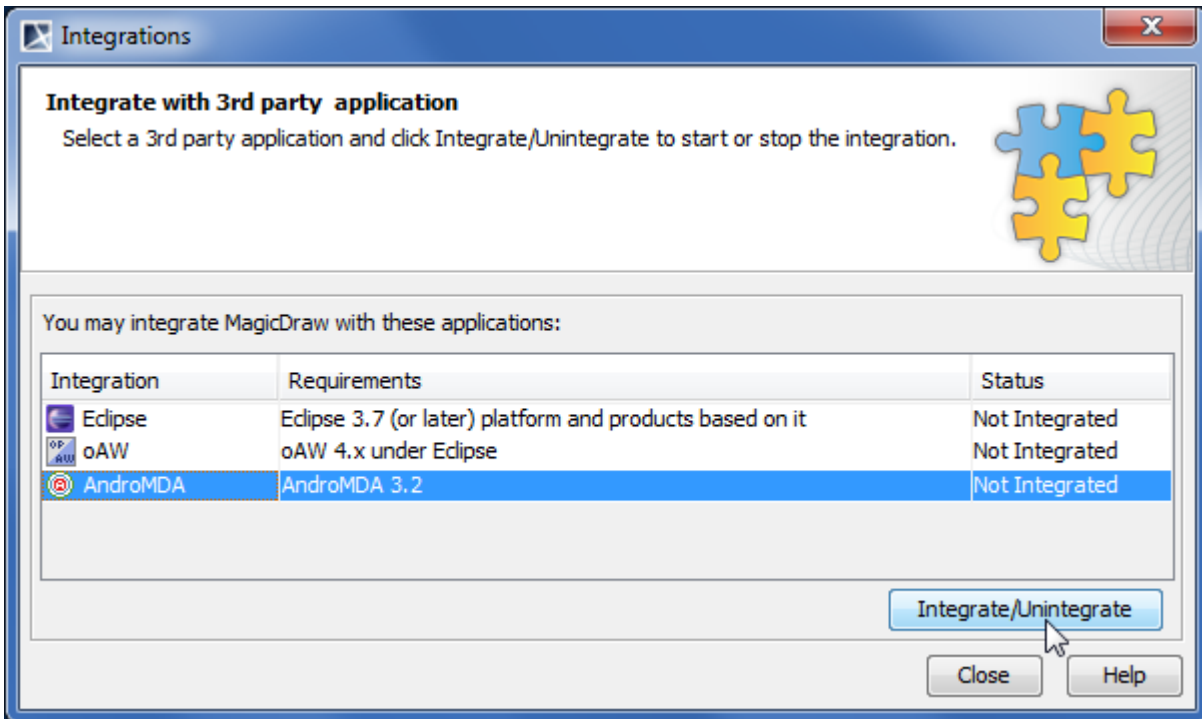


Figure 1 -- Integrations dialog

In both cases a dialog for MagicDraw integration with AndroMDA opens.

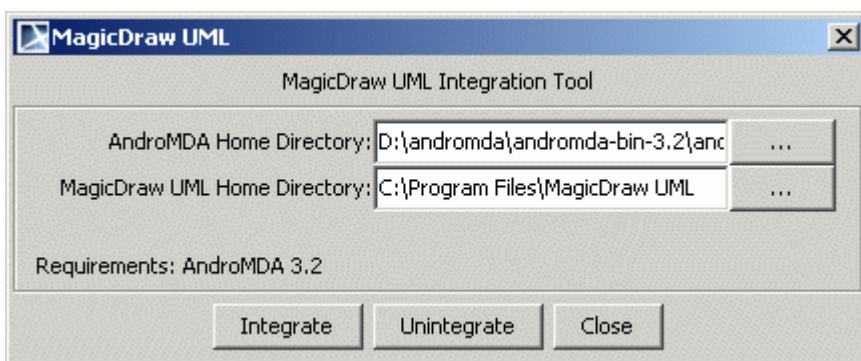


Figure 2 -- MagicDraw UML Integration Tool

If you do not need to change either directory, click the **Integrate** button in the dialog.

To change the AndroMDA or MagicDraw home directories

1. Click the ... button in the dialog for MagicDraw integration with AndroMDA. The **AndroMDA Home Directory** or **MagicDraw Home Directory** dialog appears.



In the **AndroMDA Home Directory** box, specify the directory wherein the AndroMDA home files are located. For example, *D:\andromda\andromda-bin-3.2\andromda\org\andromda*.

2. Select the installation directories and click **OK**.
3. Click **the Integrate button** in the dialog for **MagicDraw Integration** with AndroMDA.
4. The Message box appears. If the integration process was successful only the **Exit** button is active. Click **Exit**.

Automatic integration process performs the following steps:

- Sets `<andromda.home>` variable to MagicDraw **Environment Options**.
- Adds paths to all AndroMDA profiles as "Global Modules Paths" in the **Environment Options** dialog, so all customer projects will be able to find AndroMDA profiles.
- Adds new custom AndroMDA Diagram.
- Adds Maven 2 plugin for MagicDraw project conversion to EMF.
- Adds predefined command lines to start AndroMDA maven tasks as "**External Tools**".

## Working with profiles

AndroMDA models use a lot of profiles that are distributed together with AndroMDA, so customers need sophisticated path variables mechanism to handle multiple tool-related paths.



TIP!

More about path variables, see *MagicDraw User Manual*, *Getting Started* section, *Setting Personal Preferences* subsection, *Environment Options* chapter, *Path Variables Pane*.

## Global modules paths

MagicDraw modules mechanism is improved to fulfill AndroMDA customer needs by adding ability to specify predefined paths for modules into **Environment Options**, not only **Project Options**.

Now paths to common modules/profiles can be specified once and they will be used in all projects automatically. Load process will look at project modules paths at first and if module/profile is not found, it will use global modules paths.

Path variables can be used to specify paths also.

Below are listed AndroMDA related paths to profiles added by automatic integration process:

```
<andromda.home>\profiles
```

```
<andromda.home>\profiles\uml2\andromda-profile\3.2
```

```
<andromda.home>\profiles\uml2\andromda-profile-datatype\3.2
```

```
<andromda.home>\profiles\uml2\andromda-profile-meta\3.2
```

```
<andromda.home>\profiles\uml2\andromda-profile-persistence\3.2
```

```
<andromda.home>\profiles\uml2\andromda-profile-presentation\3.2
```

```
<andromda.home>\profiles\uml2\andromda-profile-process\3.2
```

```
<andromda.home>\profiles\uml2\andromda-profile-service\3.2
```

```
<andromda.home>\profiles\uml2\andromda-profile-webservice\3.2
```

```
<andromda.home>\profiles\uml2\andromda-profile-xml\3.2
```

```
<andromda.home>\profiles\uml2\andromda-profiles-uml2\3.2
```

## UML\_Standard\_Profile-3.2.xml

Some AndroMDA profiles use *UML\_Standard\_Profile-3.2.xml*, but such profile does not exist. Copy *UML\_Standard\_Profile.xml* from MagicDraw profiles folder to AndroMDA profiles folder and rename it to *UML\_Standard\_Profile-3.2.xml* to solve this problem.

## EMF export

MagicDraw UML native file format is UML 2 XMI 2.1.

Eclipse UML 2 (v1.x) XMI, Eclipse UML 2 (v2.x) XMI, and Eclipse UML2 (v3.x) XMI file format support is added as plugins with import/export functions.

EMF export is available with free Community or Personal editions also.

Users are able to change EMF export options in the MagicDraw **Options** main menu, **Environment Options**, **Eclipse UML 2 (v1.x) XMI**, **Eclipse UML 2 (v2.x) XMI**, or **Eclipse UML 2 (v3.x) XMI** options groups.

## How and when EMF export is used

There are two major cases of AndroMDA usage:

### 1. Inside MagicDraw

In this case command line exporter or maven plugin should not be used, because it starts new MD application to convert model to EMF.

MD save/export synchronization option should be used or EMF export should be performed before external tool is used (see External Tools chapter)

### 2. Outside MagicDraw

In cases like nightly builds, when EMF conversion of model files is needed outside MagicDraw application, Maven plugin or command line EMF exporter should be used. For better performance new option for file date checking

## Synchronization options

Special options are added into EMF export options, to synchronize MagicDraw native save and EMF export actions:

The **Export Model to Eclipse UML2 XMI on project save** has following options:

- *Do not export* - synchronization is turned off.
- *Ask Before export* - asks to export into EMF on every "Save" action.
- *Always export* - uses silent mode and exports model to EMF on every Save.

The **Ask to overwrite exported files** - ability to turn on/off warning when existing files are overwriting.



EMF exporting on every saving increases time twice, so use this option only if you always need these files synchronized.

If EMF is needed just on launching some build process, External Tools with included EMF export could be used (see "External tools" on page 37).

## EMF Export Location

Every project has predefined the **Eclipse UML2 XMI output location** property. It could be modified in MagicDraw **Options** main menu, the **Project Options** dialog box, **General project options** pane.

Path variables could be used to specify location, so `<andromda.home>` variable or other may be used and multiple users could work on the same project without changing export location.

This location is changed every time when user uses EMF export manually. Last used export location is saved.

## Command line converter

EMF plugin provides ability to use EMF export function using batch mode.

Special executables for Windows and Unix are available in plugin folder:

- `\plugins\com.nomagic.magicdraw.emfuml2xmi_v1\` (for Eclipse v1.x)
- `\plugins\com.nomagic.magicdraw.emfuml2xmi_v2\` (for Eclipse v2.x)
- `\plugins\com.nomagic.magicdraw.emfuml2xmi_v3\` (for Eclipse v3.x)

To start EMF XMI export from command line

Use `exportEMFXMI project_file=<path to project file> destination_dir=<path to destination directory> load_all_modules=<true | false>`

Parameters:

- “**project\_file**” – absolute file name. If relative file name is used, it should be in the same folder as converter launcher.
- “**destination\_dir**” – folder where converted EMF files should be saved. It can be multiple EMF files after conversion of one MD XMI file, because every Profile element is separate file in EMF.
- “**project\_descriptor**” – special URI for project identification. It can be used for both local or teamwork projects. Teamwork project descriptor includes encrypted login information, so it is enough information to log into remote Teamwork Server and get newest version of some project. For more information, see Project Descriptor on page 45.

MagicDraw project descriptor is visible and selectable in the MagicDraw **Project Properties** dialog box, **General** tab (**File->Project Properties**).

Descriptor for the local file contains just absolute file name, so there is no difference which parameter is used – “project\_file” or “project\_descriptor”.

Project descriptor OR project file name could be used, but not both at the same time.

- “**load\_all\_modules=<true | false>**” - if this option is “true”, converter loads all used modules even if they are in “auto load” or “manual load” mode.
- “**check\_time=<true | false>**” - if this option is “true”, converter shall be started only if EMF file is older than native XMI file or EMF file doesn't exist. “false” by default - means that file is always converted if this property is not specified.

All parameters from environment options are reused during conversion, so, for example, leave “Preserve exported IDs” turned on in MagicDraw.



All used modules and profiles will be converted every time, so it could take a while to convert even a small project if all andromda profiles are used. It is recommended to use command line converter for nightly builds on server where time has not the highest priority.

### Maven 2 plugin

Batch mode EMF exporter is presented in form of Maven2 plugin.

AndroMDA users are able to include MagicDraw project file to EMF conversion task into other Maven scripts.

Integration process set "**md\_home**" variable and moves Maven plugin to Maven repository, but only if it is in user home directory.

In other case user should move this plugin manually from AndroMDA integrator plugin folder (*plugins\com.nomagic.magicdraw.andromdaintegrator\maven\_plugin*) to Maven repository.

There is one sample in this folder that illustrates how converter should be used and what parameters it requires. Look to sample *pom.xml* file for details.

Maven plugin uses the same parameters as command line EMF exporter and few additional:

- *md\_home* - MagicDraw install folder. Normally integration process should set this variable and customer does not need to specify it manually and should use it only when path is changed or not found.
- *properties\_file\_path* - path to EMF exporter property file. Ability to use different EMF exporters.
  - com.nomagic.magicdraw.emfuml2xmi\_v1\exportEMFXMI.properties (used by default)
  - com.nomagic.magicdraw.emfuml2xmi\_v2\exportEMFXMI.properties
  - com.nomagic.magicdraw.emfuml2xmi\_v3\exportEMFXMI.properties



*<md\_home>* parameter should contain absolute path to MagicDraw install folder if this sample is running from Maven related folder.

\$MD\_HOME\$ and \$ANDRO\_MDA\_HOME\$ path variables could be used everywhere in paths.

"project\_file" or "destination\_dir" are relative to MD home if relative file names are used.

## External tools

Any command line could be started directly from MagicDraw.

In the MagicDraw **Options** main menu, go to **Environment Options**, select **External tools** group and set any command lines (e.g. to open external XML viewer, commit to CVS, compile generated Java source code or any other).

Path variables could be used in these commands (syntax the same as in modules paths), also new additional variable - "\$f" to indicate absolute file name of current project file.

There is ability to use automatic Save/Export function before command line is launched, so AndroMDA users can use EMF export only before Maven is started.

Multiple command lines to start, build, install or deploy project (commands to launch Maven with different parameters) could be added and used directly from MagicDraw main toolbar as in all major IDE interfaces.

External Tools toolbar is not visible by default, so user should right click on main toolbar and select **External Tools** to be visible (Switch to other Perspective if such toolbar doesn't exist).

More about external tools, see *MagicDraw User Manual*, *Getting Started* section, *Setting Personal Preferences* subsection, *Environment Options* chapter, *External Tools Pane*.



In the command-line interface type:

```
cmd /c start <install.root>/readme.html
```

The command starts the default Windows browser and opens *readme.html* from the MagicDraw installation folder.

In order to successfully build your project using Maven, you will need to know how to invoke the build process for the existing modules. Here is a list of examples:

Command	Option
mvn install	simply builds all modules
mvn -f web/pom.xml -Ddeploy	collects all artifacts and builds a deployable .war which is then deployed
mvn clean	cleans all generated files from each target directory
mvn install -Ddeploy	rebuilds the entire application and deploys

## Customization of AndroMDA profiles/cartridges

MagicDraw provides special set of customizations that helps to use AndroMDA cartridges more comfortable and saves a lot of time on modeling.



See more information about usage of concrete cartridges:

<http://team.andromda.org/docs/andromda-cartridges/index.html>

Customizations include:

- Custom AndroMDA Diagram with predefined stereotypes in toolbar;
- Predefined style of most often used stereotypes;
- Customized specification dialogs - all tags appear as regular properties;
- Predefined basic semantics.

## AndroMDA Diagram

AndroMDA Diagram uses all AndroMDA profiles and suggests most often used stereotypes to create from toolbar.

Diagram Toolbar includes:

- Entity
- Service
- Web Service
- Enumeration
- Embedded Value
- Value Object
- Exception
- Application Exception

- Unexpected Exception

### Predefined semantics

- `<<FrontEndUseCase>>` could be applied to any UseCase and is available in shortcut menu for regular UseCases.
- `<<FrontEndView>>` could be applied to any State and is available in shortcut menu for regular States as checkbox.
- "Unique" checkbox will be available in shortcut menu of regular Property.
- Identifier, FinderMethod and CreateMethod are suggested as "New Elements" for Entity.
- ServiceElement is suggested to create for Service.

Some stereotypes are used only as "holders" of tags, so user will be able to specify these tags even if stereotype is not applied. Stereotype will be applied automatically.

Such behavior is added for these stereotypes:

- `<<PersistentAssociation>>`
- `<<PersistentAssociationEnd>>`
- `<<PersistentAttribute>>`
- `<<PersistentClass>>`
- `<<PersistentElement>>`
- `<<PersistenceProperty>>`

### Converting Class Diagrams to AndroMDA Diagrams

Legacy projects may contain multiple Class diagrams that would be nice to migrate to AndroMDA diagrams.

Follow the steps below:

1. Open project and make sure you are using all required andromda profiles (if not, simply create new empty AndroMDA diagram).
2. Save project into plain XML format.
3. Open it in XML Editor.
4. Replace all occurrences of "`<type>Class Diagram</type>`" into "`<type>AndroMDA Diagram</type>`".
5. Save XML.
6. Open project in MagicDraw - all Class Diagrams become AndroMDA diagrams.

# INTEGRATION WITH OAW

## Introduction



- This functionality is available in Standard, Professional, Architect, and Enterprise editions only.
- In oAW 5, the uml2ecore features is deprecated. If you want to create model in MagicDraw and generate ecore file. Please use Export to EMF Ecore File feature or oAW 4.x instead.

OpenArchitectureWare (henceforth oAW) is a popular model driven development & architecture framework based on Eclipse. This framework consists of various components which allow user to manipulate models in various ways.

oAW supports model checking against a defined set of validation rules:

- model-to-model transformations (e.g. deriving new models from existing models, enriching existing model with additional derived information)
- model-to-text/text-to-model (e.g. code generation from models or parsing of structured texts into the model form), and more.

In general, oAW is modeling-tool agnostic. Modeling tools serve as initial source of the models to be manipulated. Native model format, used by oAW, is EMF but other formats (MagicDraw native format among others) are also accessible using **org.openarchitectureware.classic.\*** plugins (read [Use of oAW Classic with MagicDraw](#) documentation chapter on oAW documentation page).

Due to this modeling tool agnosticism, big swaths of oAW functionality can be used without any integration or additional support from MagicDraw side. Hence this MagicDraw integration for oAW only adds two features in the model handling domain. The two added features are:

- "oAW Metamodeling Diagram", on page 43 - to be used together with oAW's uml2ecore transformer - simplifying metamodel preparation.
- "Workflow Component for MagicDraw EMF Export", on page 43 - allowing triggering of MagicDraw's Eclipse UML2 XMI export feature directly from oAW workflow scripts.

## Installation

Integration with oAW is installed in a similar manner as all other integrations.

oAW has two specific features:

- The custom oAW metamodeling diagram, which operates under MagicDraw.
- The oAW workflow component task, which has to be packaged as Eclipse plugin.

There are several cases of installing the integration: you can install either only one feature or both of them.



### System Requirements

oAW version 4.1 or later installed on Eclipse version 3.7, 4.1, or later.

Required Eclipse version may depend on the oAW version used. Read the installation instructions of your oAW version. MagicDraw does not place additional constraints on Eclipse version beyond v3.6 or higher constraint.

Recommended configuration (which received most of the testing) is oAW v4.1.2 on Eclipse v3.7.

To install oAW into Eclipse, follow oAW's documentation:

<http://www.openarchitectureware.org/staticpages/index.php/documentation>

<http://www.eclipse.org/gmt/oaw/doc/>

oAW is installed into Eclipse as a set of plugins. This plugin set can be installed as a whole or some selective subset of plugins can be installed (obeying interdependence between plugins - Eclipse plugin install mechanism automatically tracks these dependencies).

For MagicDraw's workflow component plugin **org.openarchitectureware.workflow** plugin is required (plus all the plugins it depends on).

For metamodeling diagram, **org.openarchitectureware.util.uml2ecore** plugin is relevant (plus all the plugins it depends on).



To use oAW5, you need to download and install 3 eclipse plug-in below:

- (M2T) - <http://download.eclipse.org/modeling/m2t/updates/releases/>
- (EMFT) - <http://download.eclipse.org/modeling/emft/updates/releases/>
- (TMF) - <http://download.eclipse.org/modeling/tmf/updates/releases/>

### Installation Process

Workflow component Eclipse plugin is always installed into Eclipse when MagicDraw is integrated with Eclipse (using Eclipse integrator). A special oAW integrator installs only the metamodeling diagram part and then invokes the Eclipse installer to install the workflow component part.

To install oAW

1. On the main menu, click **Tools > Integrations**. The **Integration** dialog opens (see the following figure).

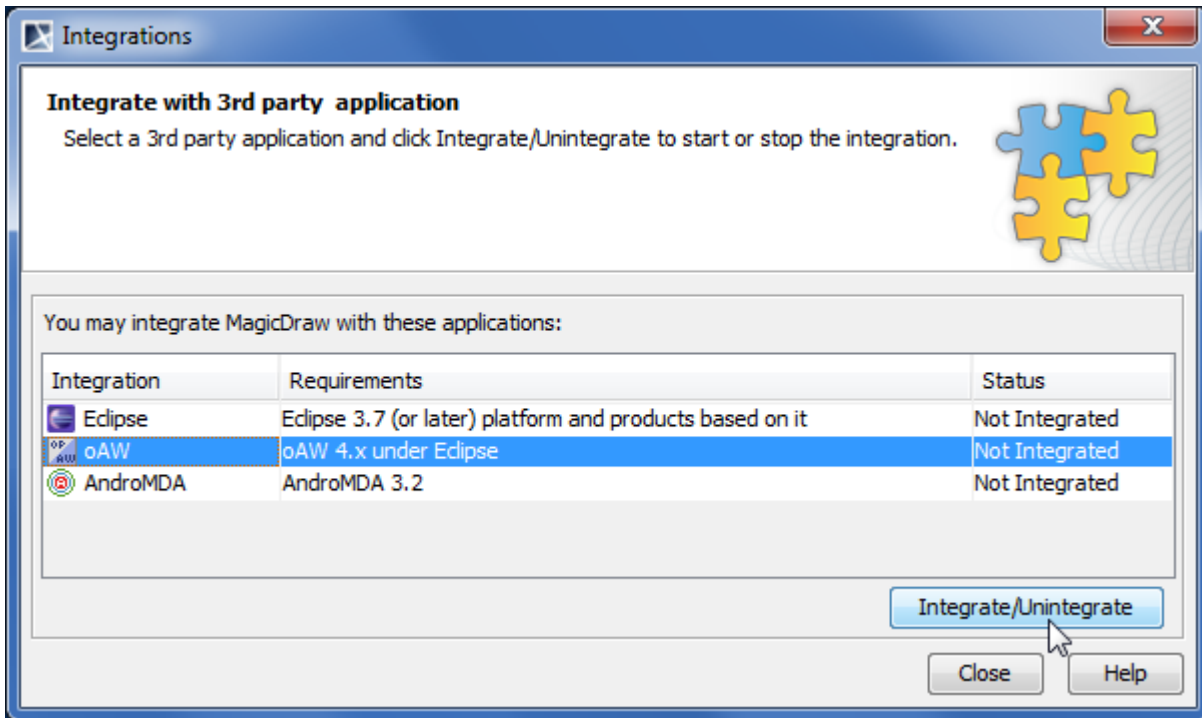


Figure 1 -- Integrations dialog box

2. Select the **oAW** and click the **Integrate/Unintegrate** button. If oAW feature is not integrated (**Status** column shows **Not Integrated**), it will be integrated; otherwise if oAW feature is integrated it will be unintegrated.

After oAW integrator finishes integration (that is - brings in the custom metamodeling diagram), it checks if MagicDraw is already integrated with Eclipse and if not, you will be prompted if you want to continue and integrate MagicDraw with Eclipse (to install the workflow component part).

After successful install or uninstall you will need to restart MagicDraw for changes to be applied (for custom diagram to appear/disappear).

To install oAW using command line

1. From MagicDraw installation directory, go to **integrations**, and then **oaw** directory.
2. Run the supplied **install** script (depending on your system this will be *install.exe* - for Windows OS, *install.sh* for Unix-like OS, *install.bat/install.sh* for no-install MagicDraw distribution).

## Uninstall

Uninstall is triggered in the same way as install (from **Integrations** dialog or from command line installer). oAW uninstall only removes the oAW metamodeling diagram. It does not uninstall MagicDraw integration with Eclipse - if you need this, you can trigger Eclipse uninstaller separately.

## Features

### oAW Metamodeling Diagram

oAW uses metamodels extensively. They are used for defining model transformation and check rules. But building metamodels with EMF's internal tools is tedious. Using the tree view based editors does not scale. Once you are at more than, say, 30 metaclasses, things become hard to work with.

One way to solve this problem is to use UML modeling tools (e.g. MagicDraw) to draw the metamodel as a simple UML model and then transform the UML model into an Ecore instance. For this purpose, oAW has a special transformation -uml2ecore. The oAW uml2ecore utility transforms a suitably structured Eclipse UML2 model into an Ecore file. Read more about this transformation in the [Transforming UML2 models into Ecore](#) part of oAW documentation.

When integrated with oAW, MagicDraw has a custom **oAW Metamodeling Diagram** for preparation of metamodels. This diagram is a simplified class diagram. All the concepts, irrelevant and not present in Ecore metamodels. This leaves only packages, classes, enumerations, associations and generalizations - hence greatly simplifying the diagram.

To prepare a metamodel, you should:

1. Create a blank MagicDraw project. You can do this in usual MagicDraw IDE or when working in Eclipse IDE with MagicDraw integrated into Eclipse.
2. Create one or more oAW metamodeling diagrams in this model and create your metamodel using these diagrams.
3. Export created metamodel to Eclipse UML2 format. MagicDraw has an utility to save the MagicDraw model files into this format. You can use any of the several ways to trigger this. You can trigger export manually, from the MagicDraw menu (**File->Export, Eclipse UML 2 (v1/2/3.x) XMI File** menu item in standalone MagicDraw IDE or **MagicDraw->Tools-> >EMF UML 2 (v2.x) XMI** menu item in Eclipse IDE with MagicDraw integrated). You can add invocation of EMF export into the oAW workflow (combine this step with the next step) and run the workflow.
4. Create and run the oAW workflow with the properly defined uml2ecore transformation, which takes the export result (Eclipse UML2 file) and transforms it into the real metamodel (EMF Ecore file).



A special note about interface support. A current (v4.1.2) uml2ecore transformation does not transform UML interfaces into Ecore constructs (there is no "interface" concept in Ecore, but UML interfaces could theoretically be transformed into Ecore classes with isInterface() set to true). Because of this, interface element is removed from MagicDraw's oAW Metamodeling Diagram. This could change in future versions of **uml2ecore** transformer.

This feature is obsoleted from oAW5.

## Workflow Component for MagicDraw EMF Export

### Introduction

Native model format, used by oAW, is EMF XMI (EMF of UML2 models, EMF of custom metamodels etc.). Models, created with non EMF-based UML tools, are generally accessible through the **org.openarchitecture-ware.classic.\*** adapter plugins but for full functionality, EMF format should be used (e.g. uml2ecore transformation needs EMF based models).

MagicDraw has functionality to export its models into Eclipse UML2 XMI format (v1.x, v2.x, and v3.x implementation of Eclipse UML2) for some time now (for more information, see *MagicDraw User Manual, Working With Projects* section, *Exporting project as an Eclipse UML2 (v1.x, v2.x) XMI file*).

There is an oAW workflow component for MagicDraw's EMF export directly from the oAW workflow.

oAW workflow is a concept similar to the Ant script - it is an XML-based script, where each item in a script describes some task. Each task can have parameters, configuring task behavior. Tasks can be composite. Users can use tasks, defined in oAW distribution, or create and use their own tasks.

Here is a sample workflow definition, consisting of one task call - MagicDraw's EMF exporter task:

```
<workflow>
<component

class="com.nomagic.magicdraw.oaw.eclipse.plugins.oaw.ExportWorkflowComponent"
  projectFile="source projects/Metamodeling example.mdzip"
  exportDirectory="converted to EMF"
  skipUpToDate="true"
  requiredVersion="2.x"/>

</workflow>
```

Tasks in workflow can be chained into a complex multistep operation, to achieve the necessary goal.

## Component Definition

To call EMF exporter you should simply use `<component>` tag with parameter `class="com.nomagic.magicdraw.oaw.eclipse.plugins.oaw.ExportWorkflowComponent"` in the workflow. Since this is not a composite task, this tag should not have any internal XML tags. It is a simple tag with these attributes, describing task parameters:

- `class`. This tag tells oAW what task to call. For MagicDraw EMF exporter task, this tag should be set to `com.nomagic.magicdraw.oaw.eclipse.plugins.oaw.ExportWorkflowComponent`.

For oAW5, the tag should be set to:

`com.nomagic.magicdraw.oaw.eclipse.plugins.mwe.ExportWorkflowComponent`.

- `projectFile`. This is a mandatory parameter, it specifies a project file for EMF exporter to export - transformation source. Here, a simple path to file (relative or absolute) or an URL (`file://` type URLs) can be used. Also, MagicDraw project on Teamwork server can be specified here (using `teamwork://` URLs). Teamworked project URL can be determined this way: open the necessary project in MagicDraw, then open **Project Properties** dialog (menu **File, Project Properties**).

There is a **Project Descriptor** field in this dialog. Copy contents of this field verbatim into the **projectFile** parameter.

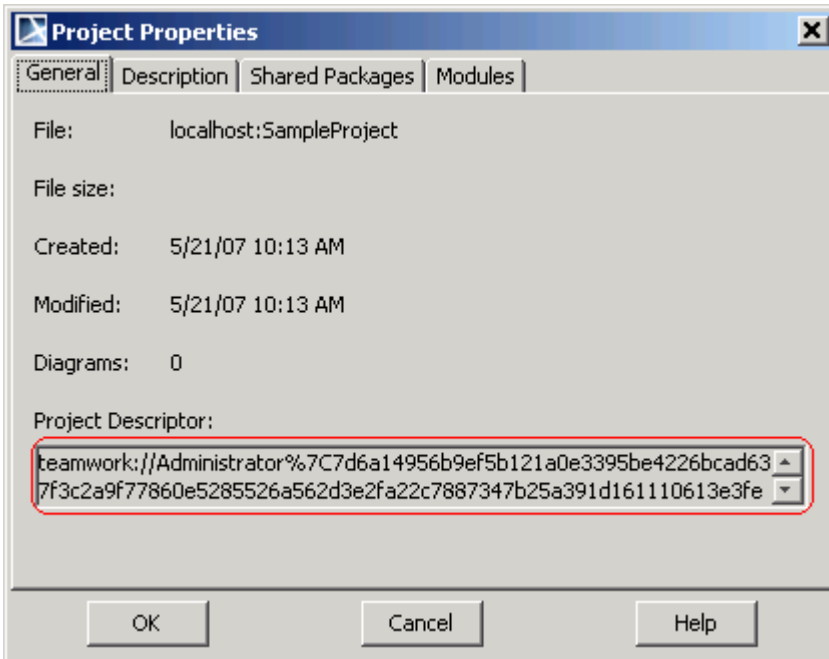


Figure 2 -- Project Descriptor URL

- **exportDirectory**. This is a mandatory parameter, specifying destination directory, where the export results will be written.
- **skipUpToDate**. This is an optional boolean parameter (defaults to **false**), which specifies wherever exporter should check file modification dates on the export target and source. If set to **true**, exporting will be skipped if export target is newer than the source.
- **requiredVersion**. This is an optional field, specifying the necessary Eclipse UML2 XML implementation version (defaults to **2.x**). Possible values are **1**, **1.x**(synonyms), **2**, **2.x**(synonyms). Note that this is not UML1 vs. UML2 export formats; both options give UML2 formats - just different implementation versions. 1.x implementation is based on UML2 (even a bit earlier, pre final UML2 version). 2.x is based on UML2.1 (where several issues of the standard were fixed).
- **magicdrawHome**. This is a text field, specifying directory on disk where MagicDraw is installed. This field is never needed in normal usage. Only when export component is used in nonstandard ways (see the section below about running from command line), this field might be necessary.

## Running

oAW workflow is normally run from Eclipse IDE.

1. You have to create oAW project (choose **File**, then **New, Project, openArchitectureWare, openArchitectureWare Project**) if you haven't already done so.
2. Then create the workflow file (**File->New->Other->openArchitectureWare->Workflow File**).
3. Fill in the workflow file - add the calls to the tasks you want to perform.
4. Run the workflow (**rightclick, Run As, oAW Workflow**).

Depending on what tasks you use in your workflow you will have to add the necessary jars where classes, implementing the tasks, reside to the classpath. This is done leveraging Eclipse plugin mechanism (since oAW project is a kind of plugin project). Open manifest of the plugin project and add the necessary plugins in the dependencies tab. Depending on the called tasks, different plugins need to be added. For workflows, calling

MagicDraw EMF export task, **com.nomagic.magicdraw.oaw** and **com.nomagic.magicdraw.eclipse.rcp** plugins need to be added (see the screenshot below). For workflows, calling e.g. uml2ecore transformation, **org.openarchitectureware.util.stdlib** and **org.openarchitectureware.util.uml2ecore** plugins should be added etc.



In order to run workflow with oAW5, com.nomagic.magicdraw.mwe and com.nomagic.magicdraw.eclipse.rcp are required.

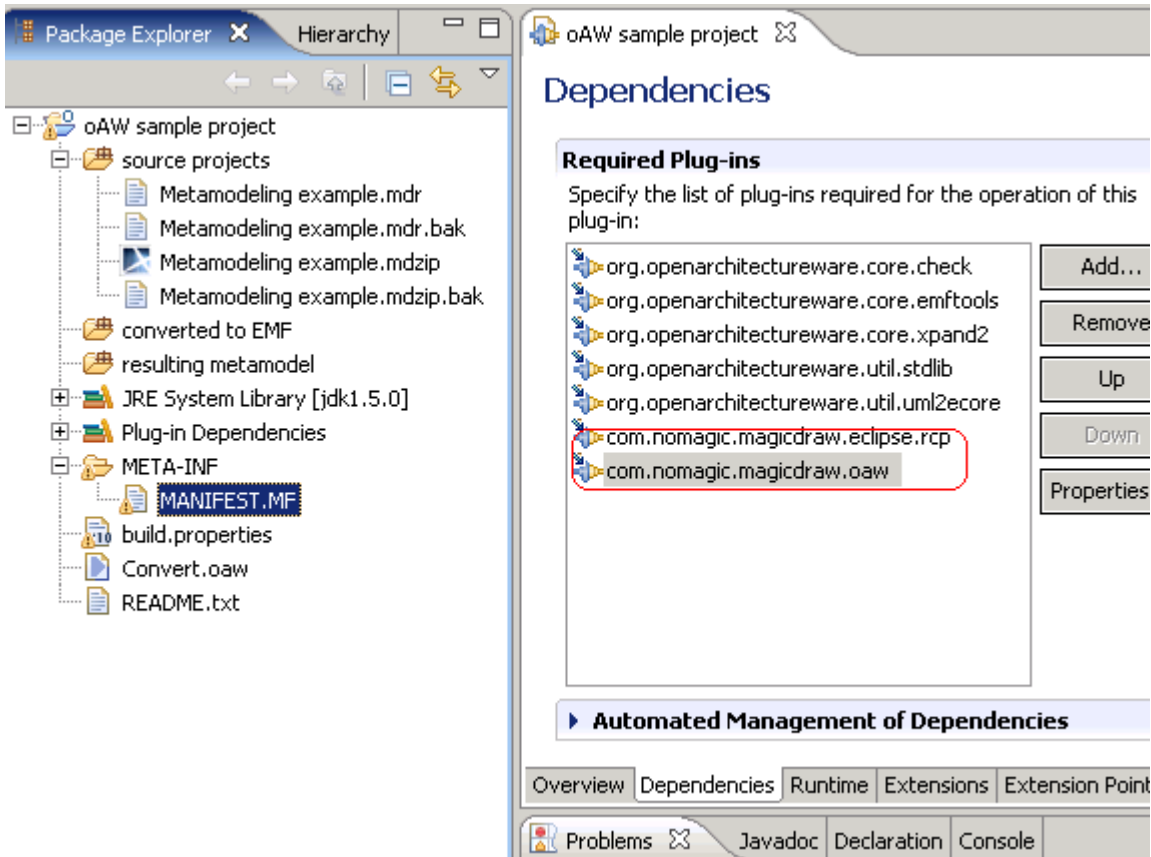


Figure 3 -- Adding necessary plugins for workflow, calling MagicDraw EMF export

oAW workflow can also be started from command line, using **WorkflowRunner** class (see [oAW documentation on workflows](#)). **Note** that in this case you are on your own - you have to build the necessary classpath for the workflow yourself. You will need oAW core jars and jars for all the components, invoked by the workflow.

MagicDraw EMF export component needs the following jars to run correctly:

- <install.root>/lib/launcher.jar
- <install.root>/plugins/eclipse/plugins/com.nomagic.magicdraw.oaw/oawplugin.jar



For oAW5, the two jar files below are required.

- <install.root>/lib/launcher.jar
- <install.root>/plugins/eclipse/plugins/com.nomagic.magicdraw.mwe/mweplugin\_api.jar

All dependencies of oAW need to be set to org.eclipse.emf.mwe.core.\*.

The best way to create the classpath is first to properly configure the workflow in Eclipse and then glance at **Plug-in Dependencies** of Eclipse project to see what jars are needed.

Be advised, that MagicDraw EMF exporter workflow task determines MagicDraw installation directory (necessary for export) on runtime, from the location of launcher.jar. Hence the two above-mentioned jars should nor-

mally be used directly from the MagicDraw installation directory. If you copy them into another place and use them from there, you will have to specify an additional parameter for the workflow task - *magicdraw-Home="path\_to\_the\_magicdraw\_installation\_directory"*.



EMF export task is rather slow, since it always starts a separate GUI-less MagicDraw instance, which performs all the work - even if MagicDraw is already started (inside Eclipse or as standalone IDE). This process may take a minute. Since it is slow, in some usage scenarios it is more convenient to trigger the EMF export by hand, than to integrate it into the workflow.