1. Introduction

Script Engine in MagicDraw allows you to create your own script by using BeanShell, Groovy, JRuby, JavaScript, or Jython. With Script Engine, you can control everything that is allowed in Open API, for example, transforming and manipulating models. You can find MagicDraw Open API in MagicDraw OpenAPI User-Guide.pdf in the manual directory and sample scripts in <MagicDraw>/samples/product features/script engine.


2. Working with Script Engine

2.1 Selecting a Default Scripting Language

Use the Environment Options dialog to select a default scripting language.

To select a default scripting language:

1. Click Options > Environment on the MagicDraw main menu (Figure 1). The Environment Options dialog will open (Figure 2).

![Figure 1 -- Environment Options Dialog Menu](image)

2. Select the Scripts node (Figure 2).
3. Click the box next to the **Default Script Language** box to see a list of supported programming languages (Figure 3).
2.2 Adding a Script, Entering, and Editing Script Information

To execute a script, you need to enter additional information to MagicDraw by following the steps described in sections 2.2.1 Opening the Script Information Dialog and 2.2.2 Adding a Script and Its Information.

2.2.1 Opening the Script Information Dialog

You can add or modify script information such as script name and description, in the Script Information dialog. To open the Script Information dialog, you need to open the Scripts dialog first.

4. Select Jython, BeanShell, Groovy, JRuby, or JavaScript.
5. Click OK to save the selected language as the default scripting language.

Figure 3 -- Selecting a Scripting Language

NOTE
- JavaScript is the default scripting language.
- Script Engine 16.8 currently supports BeanShell, Groovy, JRuby, JavaScript, and Jython only.
To open the **Scripts** dialog:

1. Click **Tools > Scripts...** on the MagicDraw main menu (Figure 4). The **Scripts** dialog will open (Figure 5).

![Figure 4 -- Scripts Dialog Menu](image)

![Figure 5 -- Scripts Dialog](image)

To open the **Script Information** dialog:

1. Click **Tools > Scripts...** on the MagicDraw main menu to open the **Scripts** dialog.
2. Click **Add** (Figure 5). The **Script Information** dialog will open.
2.2.2 Adding a Script and Its Information

Use the Add or Edit button in the Scripts dialog to add or edit a script and its information in the Script Information dialog. You can also press the mnemonic keys to add or edit a script (see 2.3.3 Scripts Dialog Mnemonic Keys).

To add a script and enter script information in the Script Information dialog:

1. Click Tools > Scripts... on the MagicDraw main menu to open the Scripts dialog.
2. Click Add. The Script Information dialog will open (Figure 6).

**NOTE**
The Edit, Delete, and Run buttons in the Scripts dialog will be disabled if there is no script in the table or if you do not select any script from the table.

3. Type the script name.
4. The default scripting language you have previously selected (see 2.1 Selecting a Default Scripting Language) will appear in the Script Language box (Figure 6).
5. Click the ... button to locate the script file. The Open file dialog will open (Figure 7).
6. Select the file and its type (there are 5 types of file filter: *.bsh, *.groovy, *.rb, *.js, or *.py) (Figure 8). If you do not select the Use path variables check box (Figure 7), the full pathname will be saved.
7. Click **Open** (Figure 8). The selected pathname will appear in the **File** box in the **Script Information** dialog (Figure 9).

---

**NOTE**

If you have specified the file or network path in the **Environment Options** dialog by clicking **Options > Environment > Path Variables** and selected the **Use path variables** check box in **Open** dialog, the `<Path Variable name>` will show in front of the file pathname. This field is the [Required] field, for example, `<mypath>script.js`.

---

**Figure 7 -- Use Path Variables Check Box**

**Figure 8 -- Types of File Filter**
8. Type the script description in the **Description** box (Figure 9).

**NOTE**
A script name must be unique and cannot be duplicated.

9. Type a keyboard shortcut that will be used to run the script in the **Press new shortcut key** box and click **Assign**. The newly assigned keyboard shortcut will appear in the **Current keys** box.

10. Click **OK**. The **Scripts** dialog will open, showing the newly added script name, description, and keyboard shortcut.

11. Click **OK** to close the **Scripts** dialog.

**NOTE**
- The script name, filename, and language are required.
- The script description and keyboard shortcut are optional.
- If any of the required fields are not filled in, a message will open: **Please input all the required fields**.
- If a duplicate script name is entered, an error message will open: **The script name has already been used, please input another name**.

### 2.2.3 Editing Script Information

You can see script information such as name and description, as well as the keyboard shortcut for the script in the **Scripts** dialog.

To edit script information:

1. Click **Tools** > **Scripts**... The **Scripts** dialog will open (Figure 10).
2. Select a script from the table and either click **Edit** or press **Alt + E**. The **Script Information** dialog will open.
3. Edit the script.
4. Click **OK** to save the information.
5. You will see the modifications in the **Scripts** dialog. Click **OK**.

### 2.2.4 Script Information Dialog Mnemonic Keys

Script Engine provides mnemonic keys to perform some operations, for example, highlight a textbox and click a specific button in the **Script Information** dialog. Table 1 lists the **Scripts Information** dialog mnemonic keys and their function.
2.3 Deleting and Executing Scripts

You can click the **Delete** or **Run** button in the **Scripts** dialog to (2.3.1) delete or (2.3.2) execute a selected script. You can also press the mnemonic keys to delete or run a script (see section 2.3.3 Scripts Dialog Mnemonic Keys below).

**2.3.1 Deleting a Script from the Scripts Dialog**

To delete a script from the **Scripts** dialog:

1. Click **Tools > Scripts...** to open the **Scripts** dialog.
2. Select a script from the table and either click **Delete** or press **Alt + D**. A dialog will open, asking whether or not you want to delete the script (Figure 11).
3. Click **Yes** and the script will be deleted from the **Scripts** dialog.

**2.3.2 Executing a Script from Scripts Dialog**

To execute a script from the **Scripts** dialog:

1. Click **Tools > Scripts...**. The **Scripts** dialog will open (Figure 12).
2. Select a script from the table and click **Run**. After the script has been executed, a message will open: `<script name> has been executed.`

**NOTE**
- You can also execute a script from the MagicDraw main browser by pressing the shortcut key that you have defined in the **Scripts** dialog.
- You can only run one script at a time.
- If there is an error while running a script, for example, syntax error, a message will open: **MagicDraw cannot execute the <script_name> script, please make sure that <path, filename, extension> is correct. <error description>**.
- If MagicDraw cannot find a script file in the location that you have specified in the **Open** dialog, a message will open: `<path, filename, extension>: File not found`.

### 2.3.3 Scripts Dialog Mnemonic Keys

Script Engine also provides mnemonic keys to add, edit, delete, and run a script from the **Scripts** dialog. Table 2 lists the **Scripts** dialog mnemonic keys and their function.

**Table 2 -- Scripts Dialog Mnemonic Keys**

<table>
<thead>
<tr>
<th>Mnemonic keys</th>
<th>Button</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alt + A</td>
<td>Add</td>
<td>To add a script in the <strong>Script Information</strong> dialog.</td>
</tr>
<tr>
<td>Alt + E</td>
<td>Edit</td>
<td>To edit a script in the <strong>Script Information</strong> dialog.</td>
</tr>
<tr>
<td>Alt + D</td>
<td>Delete</td>
<td>To delete a script from the <strong>Scripts</strong> dialog.</td>
</tr>
<tr>
<td>Alt + R</td>
<td>Run</td>
<td>To run a script from the <strong>Scripts</strong> dialog.</td>
</tr>
</tbody>
</table>
2.4 Script Keyboard Shortcuts

Script Engine allows you to assign keyboard shortcuts to the scripts that you have created using either the Environment Options (Figure 13) or Script Information dialog (Figure 14). You can later press the keyboard shortcuts to execute or run the scripts in MagicDraw.

To open the keyboard shortcuts pane in the Environment Options dialog:

1. Click Options > Environment on the MagicDraw main menu. The Environment Options dialog will open.
2. Click Keyboard (Figure 13).

Script information and keyboard shortcuts that are saved either in the Environment Options or Script Information dialog will be stored as a MagicDraw environment. They will not be kept in a specific project file[*.mdzip].

Figure 13 -- Environment Options Dialog
Figure 14 -- Script Information Dialog

NOTE

- The Assign button will be disabled until you type a keyboard shortcut in the Press new shortcut key box.
- The Remove and Remove All buttons will be disabled if there is no keyboard shortcut in the Current keys box.

Table 3 -- Keyboard Shortcut Buttons

<table>
<thead>
<tr>
<th>Button</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assign</td>
<td>To assign a new keyboard shortcut to a script.</td>
</tr>
<tr>
<td>Remove</td>
<td>To remove a keyboard shortcut from a script.</td>
</tr>
<tr>
<td>Remove All</td>
<td>To remove all selected keyboard shortcuts from a script.</td>
</tr>
</tbody>
</table>

Table 4 -- Keyboard Shortcut Text Boxes

<table>
<thead>
<tr>
<th>Field</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current keys</td>
<td>To store a list of keyboard shortcuts currently assigned to a script.</td>
</tr>
<tr>
<td>Press new shortcut key</td>
<td>To type a new keyboard shortcut to be assigned to a script.</td>
</tr>
</tbody>
</table>
2.4.1 Assigning a Keyboard Shortcut to a Script

To add a new keyboard shortcut to a script:

1. Open either the (i) Environment Options or (ii) Script Information dialog.
2. Type a keyboard shortcut in the Press new shortcut key box.
3. Click Assign to assign the keyboard shortcut to a script. If the keyboard shortcut has never been used by any scripts, it will appear in the Current keys box.
4. Click OK.

**NOTE**
If a keyboard shortcut key entered in the Press new shortcut key box has already been used by another script, a message will appear under the Press new shortcut key box: Currently assigned to <the command name>.

2.4.2 Removing a Keyboard Shortcut from a Script

To remove a keyboard shortcut:

1. Open either the (i) Environment Options or (ii) Script Information dialog.
2. Select a keyboard shortcut from the Current keys box in the Script Information dialog.
3. Click Remove. The selected keyboard shortcut will be removed from the Current keys box.

**NOTE**
You can also remove a keyboard shortcut in the Environment Options dialog by clicking Options > Environment > Keyboard.

To remove all keyboard shortcuts:

1. Open either the (i) Environment Options or (ii) Script Information dialog.
2. Click Remove All. All keyboard shortcuts will be deleted from the Current keys box.

**NOTE**
- The Assign, Remove, and Remove All buttons will be disabled if there is no keyboard shortcut either in the Press new shortcut key or Current keys box.
- The Assign button will be enabled if the new keyboard shortcut entered has not been assigned to any other script.
- You can assign more than one keyboard shortcut to a script.
APPENDIX

Script Engine supports five scripting languages: (i) BeanShell, (ii) JavaScript, (iii) JRuby, (iv) Jython, and (v) Groovy.

(i) BeanShell

BeanShell (JSR 274) is a lightweight scripting language for Java. The shipped version is 2.0b4. The advantage of using BeanShell is that its syntax is compatible with Java. So, you can use the code assistant feature in most Java IDE. The BeanShell syntax documentation is available at http://www.beanshell.org/docs.html.

(ii) JavaScript


(iii) JRuby

JRuby is a 100% pure-Java implementation of the Ruby programming language. Script Engine uses JRuby 1.3.1. The documentation is available at http://kenai.com/projects/jruby/pages/Home.

(iv) Jython


(v) Groovy

Groovy is an agile and dynamic language for the Java Virtual Machine. Script Engine uses Groovy 1.7.0. The advantage of using Groovy is that its Java-like syntax seamlessly integrates with the existing Java source code and libraries. It supports many IDEs that provide code completion and debugging. BeanShell scripts can be easily moved to Groovy, with some modifications, to benefit from code completion. For more information on Groovy, go to http://groovy.codehaus.org.

1. Using Code Completion to Develop BeanShell Scripts

You can use any text editor to develop a scripting language. However, a standard text editor lacks of code assistant features. Most scripting languages are loose type. For example, to define a variable in JavaScript, you need to type:

```java
var a;
```

It is difficult to determine what type of "a" later in the source code. With BeanShell, you can use a variable without declaring it, for example:

```java
a = new File("file.txt");
```

Or you can declare it first:
Java IDE does not officially support code completion for scripting languages. However, there is a workaround if you use BeanShell. First you need a Java IDE. If you do not have one, you can select NetBeans because it has the smallest file size. You can download the latest version of NetBeans at http://www.netbeans.org/downloads/index.html. The NetBeans Java SE package is enough. Second you need to set up a MagicDraw classpath.

To set up a classpath point in the MagicDraw library in NetBeans:

1. Click Tools > Libraries on the main menu. The Library Manager dialog will open.
2. Click the New Library button. The New Library dialog will open.
3. Specify a library name, for example, MD16.6. The Library type must be Class Libraries.
4. Click OK to close the New Library dialog.
5. Select your new library in the Libraries tree.
6. Click the Add JAR/Folder... button and add all the JAR files in <MagicDraw>/lib.
7. Click OK to close the Library Manager dialog.

To develop a BeanShell script in NetBeans:

1. Click File > New Project on the main menu to create a Java application project. The New Project dialog will open.
2. Select Java in the Categories box and Java Application in the Projects box, and then follow the instructions.
3. Expand your project node in the Project window. The Libraries node will appear.
4. Right-click the Libraries node and select Add Library from the shortcut menu.
5. Select the MagicDraw library that you have previously created (see "To set up a classpath point in the MagicDraw library in NetBeans:" ) and click Add Library.
6. Click File > New File on the main menu to add a new Java file.
7. Select Java in the Categories box and Empty Java File in the File Types box. And then follow the instructions until finish.

You need to create a public static method in a Java file, for example, main() method to follow the standard Java programming language. At the end of the file, insert a statement to call the static method. See the example in create_project_elements.bsh in the <MagicDraw>/samples/product features/script engine directory.

NOTE: The official filename extension for BeanShell is .bsh. However, you can add a .java file to the BeanShell scripting language.

2. Using NetBeans IDE to Develop Groovy Scripts

You will need NetBeans IDE 6.7.1 or 6.8 with Groovy Plugin to develop Groovy scripts. The Groovy support comes with NetBeans "Java" and "All" package. If your NetBeans does not have Groovy, download the plugin through NetBeans Plugin Manager. For more information on adding a new plugin, click IDE Basics > Plugins > Updating the IDE on the NetBeans Help menu.

Use the following three simple steps to develop and run a Groovy script in NetBeans:

1. Set up a classpath.
2. Develop a Groovy script.
3. Run the Groovy script.

1. To set up a classpath:

   1. Click **Tools > Libraries** on the main menu. The **Library Manager** dialog will open.
   2. Click the **New Library** button. The **New Library** dialog will open.
   3. Specify a library name, for example, MD16.8. The Library Type must be Class Libraries.
   4. Click **OK** to close the **New Library** dialog.
   5. Select your new library in the Libraries tree.
   6. Click the **Add JAR/Folder...** button and add all the JAR files in `<MagicDraw>/lib`.
   7. Repeat steps 2 to 6 to add the Apache BSF library that is in `<MagicDraw>/plugins/com.nomagic.magicdraw.automaton/lib/bsf.jar`.
   8. Repeat steps 2 to 6 to add the Groovy library that is in `<MagicDraw>/plugins/com.nomagic.magicdraw.automaton/engines/groovy-1.7.0/embeddable/groovy-all-1.7.0.jar`.
   9. Click **OK** to close the **Library Manager** dialog.

2. To develop a Groovy script:

   1. Click **File > New Project** on the main menu to create a Java application project. The **New Project** dialog will open.
   2. Select **Java** in the **Categories** box and **Java Application** in the **Projects** box, and then click **Next**.
   3. Choose a project location. Be sure that you do not select **Create Main Class**.
   4. Click **Finish**.
   5. Expand your project node in the **Project** window. The Libraries node will appear.
   6. Right-click the Libraries node and select **Add Library** from the shortcut menu.
   7. Select the MagicDraw, Apache BSF, and Groovy libraries that you have previously created (see “1. To set up a classpath:” above) and click **Add Library**.
   8. Click **File > New File** on the main menu to add a new Groovy file.
   9. Select Groovy in the **Categories** box and Groovy Script in the **File Types** box.
   10. Follow the instructions until finish.

(iii) To run the Groovy script:

   1. Right-click the project node in the **Projects** window and click **Properties**.
   2. Click **Run node**. The **Project Properties** dialog will open.
   3. Enter the following values:
      - Main Class: com.nomagic.magicdraw.Main
      - Working Directory: `<MD installation directory>`
      - VM Options: -Xmx800M -XX:PermSize=40M -XX:MaxPermSize=150M

   **NOTE** These parameters are taken from `<MagicDraw>\bin\mduml.properties`.

   4. Run the project by clicking **Run > Run Main Project**.
   5. Add the script to MagicDraw (see “1. To set up a classpath:” above).

**NOTE** Although the configuration settings presented in the section allows you to debug a script, debugging a Groovy script may cause the MagicDraw application to stop responding.
3. Using Eclipse to Develop Groovy Scripts

Use the following three simple steps to develop and run a Groovy script in NetBeans:

1. Install Groovy-Eclipse Plugin.
2. Develop a Groovy script.
3. Run the Groovy script.

1. To install Groovy-Eclipse Plugin:

   2. Follow the instructions for installing Groovy-Eclipse Plugin.

2. To develop a Groovy Script:

   1. Create a Groovy project.
   2. Right-click the project in Package Explorer and click Properties.
   3. Click Java Build Path in the tree on the left-hand side. Click the Libraries tab on the right pane.
   5. Click OK to close the dialog.
   6. Create a new Groovy class in the project. Remove a class declaration, and then put the script there.

3. To run the Groovy script:

   1. Right-click the created Groovy project and select Run As > Java Application. You will be prompted to select a main class. Choose com.nomagic.magicdraw.Main from the list. The first run will fail, but you will have a run configuration for the script.
   2. Click Run Configuration on the Run menu.
   3. Click the configuration on the left pane, under the Java Application node.
   4. Make sure that the main class is set to com.nomagic.magicdraw.Main.
   5. Click the Arguments tab. And in the VM arguments section:

      • enter: -Xmx800M -XX:PermSize=40M -XX:MaxPermSize=150M
      • set the Working directory to Other
      • Click the File System button to point to your MagicDraw installation directory.

   6. Run the created configuration again.
   7. Add the script to MagicDraw (see “1. To set up a classpath:" above).

**Note:** Although the configuration settings in the section allows you to debug a script, debugging a Groovy script may cause the MagicDraw application to stop responding.