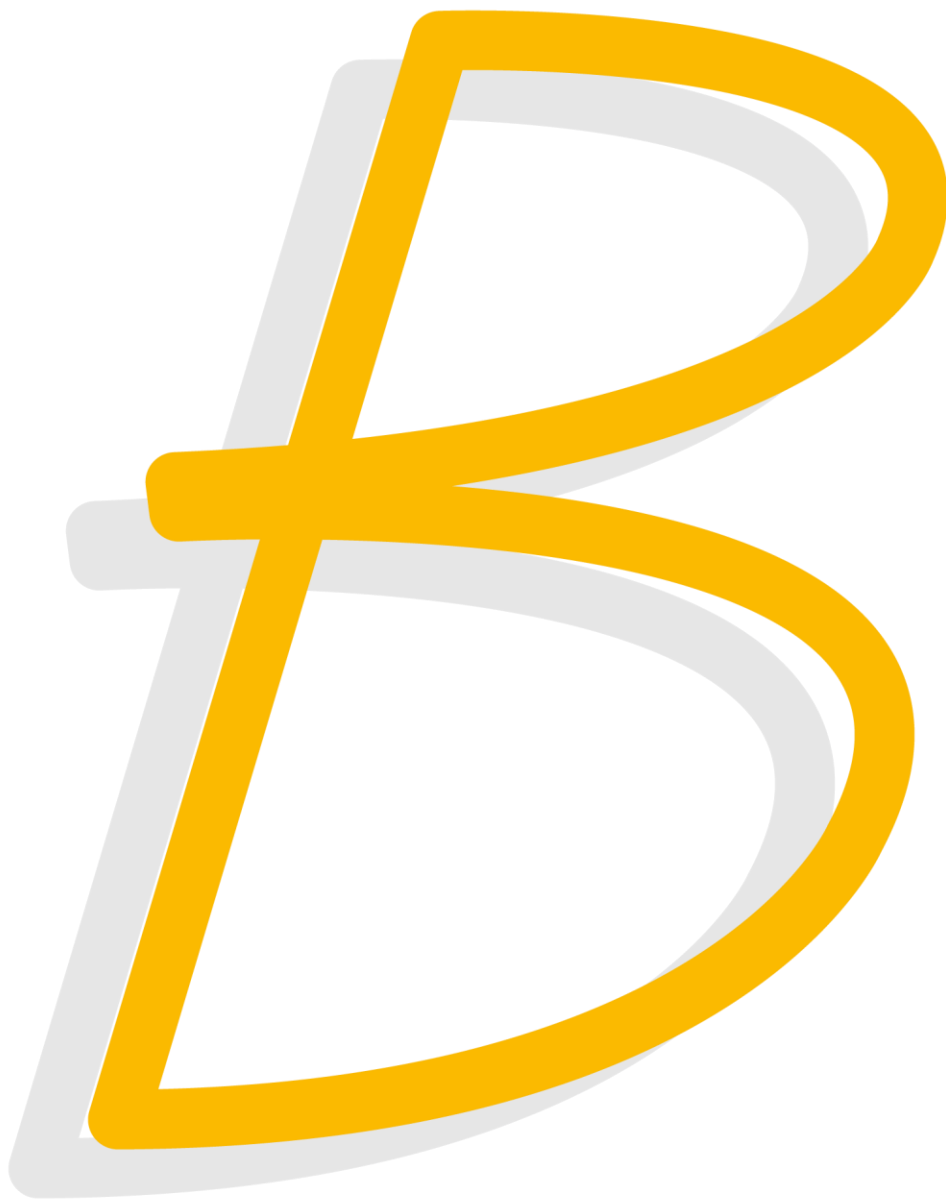
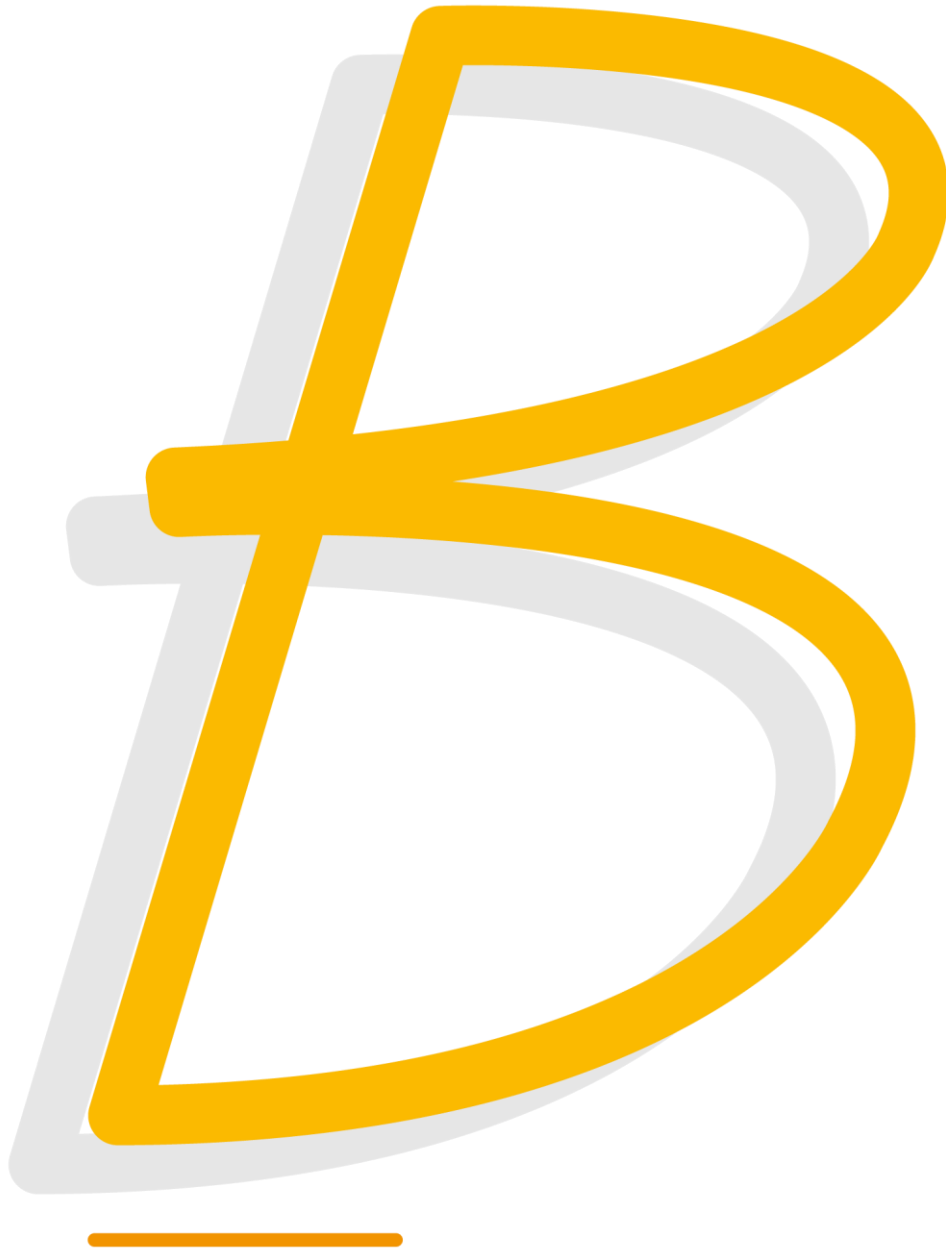


Atelier B User Manual

Atelier B
Extention Reference
Version 4.4

- Software development
- System modelling





Document designed by CLEARSY.
All trademarks are the property of their respective owners.
This document is released with the Creative Commons - Attribution License (CC-By).

This document describes the extensibility features found in Atelier B 4.4. Atelier B 4.4 allows running "external tools" from within the GUI.

For now two kinds of external tools can be integrated:

- project external tools. Those tools are running on a project, and appears in the "Project" menu of the GUI. For instance, they can be used to provide metrics on a project
- component. Those tools are running on a component, and appears in the "Component" menu of the GUI. They are launched on the selected component(s)

TABLE OF CONTENTS

1	Integrating an external tool	6
1.1	A simple examples	7
1.2	A more complex example	8
2	List of recognised tags	9
2.1	externalTool	10
2.2	toolParameter	10
2.3	temporaryFile	11
2.4	command	12
2.5	param	12
2.6	List of predefined variables	12
3	REFERENCES	13
3.1	Glossary	14
3.2	Bibliography	14

01.

1 INTEGRATING AN EXTERNAL TOOL



Integrating an external tool is done by writing a ".ETOOL" xml file. This file describes the type of tool, as well as the command that will be started. This file should be located in the "extensions" subdirectory of the Atelier B installation, and is loaded when the Atelier B GUI start.

For each "ETOOL" file, a corresponding menu entry is created within the "Project" or "Component" menu of the GUI. When the menu is clicked, the tool is launched, and the corresponding task appears in the "Tasks" window of Atelier B. Double-clicking on the task displays a windows showing the output of the tool.

1.1 A simple examples

Here is a simple example of tool for windows that open the project database file (bdp) in an explorer window:

```
<externalTool name="openbdp"
  category="project"
  label="Open bdp"
  shortcut="Ctrl+Y"
  tooltip="Opens the project database directory in an explorer
window"
  icon="openbdp.png"
  >
  <command>c:\windows\explorer.exe</command>
  <param>${projectBdp}</param>
</externalTool>
```

The file starts with the *externalTool* tag, which contains the following information as attributes:

- *name*: the internal name of the tool (required)
- *category*: the category of the tool. Can be "component" or "project" (required)
- *label*: the text that is displayed in the corresponding menu entry (required)
- *shortcut*: an optional keyboard shortcut for calling the tool
- *tooltip*: information that is displayed about the tool
- *icon*: the icon that will be used in the corresponding menu entry. This must be the name of a .png file located in the same directory as the etool file.

Within the external tool, the following tags are used to describe the command:

- *command*: the path to the executable file that will be started (here *c:\windows\explorer.exe*)
- *param*: describes one parameter of the command. Here the command takes only one parameter, but for more complex commands with multiple parameters, there should be as many *<PARAM>* as parameters.

The variable elements of the command (here the path to the bdp directory) are expressed using the `${variableName}` notation. Here `${projectBdp}` is replaced by the path to the bdp directory of the project.

1.2 A more complex example

The previous example had one major shortcoming: the path to the **"EXPLORER"** executable is hardcoded. A better solution would be to allow the user to configure the path to the **"EXPLORER"** executable. This can be done by using the `toolParameter` tag, that allows defining a new variable for using within the command. In that case, the corresponding **ETOOL** file would be the following:

```
<externalTool name="openbdp"
  category="project"
  label="Open bdp"
  shortcut="Ctrl+Y"
  tooltip="Opens the project database directory in an explorer
window"
  icon="openbdp.png" >
  <toolParameter name="explorer" type="exefile"
default="c:\windows\explorer.exe" />
  <command>${explorer}</command>
  <param>${projectBdp}</param>
</externalTool>
```

Here, the `toolParameter` tag define a new variable that can be set by the user: a new tab is created in the **"Preferences"** dialog, that allows setting the path to the **"explorer"** executable (linux users could also provide a more suitable file manager).

02.

2 LIST OF RECOGNISED TAGS

Here is the list of recognised tags as well as their allowed attributes.

- **externalTool**: The enclosing tag of the etool file. Defines an external tool
- **toolParameter**: Used to define parameters for running the command
- **temporaryFile**: Used to create temporary files before running the command
- **command**: The command that should be run
- **param**: a parameter to the command

The allowed attributes are listed in the next sections.

2.1 externalTool

The **externalTool** tag is the main tag of the etool file. It is required, and can contain the following attributes:

- **name**: the internal name of the tool (required)
- **category**: the category of the tool. Can be "**component**" or "**project**" (required)
- **label**: the text that is displayed in the corresponding menu entry (required)
- **shortcut**: an optional keyboard shortcut for calling the tool
- **tooltip**: information that is displayed about the tool
- **icon**: the icon that will be used in the corresponding menu entry. This must be the name of a .png file located in the same directory as the etool file.

2.2 toolParameter

The **toolParameter** tag is used to define a variable that will be expanded when running the tool. The allowed attributes are the following:

- **name** (required): the name of the parameter. The name correspond to the name of the variable
- **type** (required): the type of the parameter. Can be one of "ressource", "exefile", "file" and "tool"
- **default**: a default value for the parameter if no other value is available
- **optional**: can be "yes" or "no" to indicate that the parameter is optional. If the parameter is not optional, the tool will not be executed in the case where no value is found for the parameter
- **description**: a textual description of the parameter. This field is used when creating a preference page for configuring the tool

Depending on the type of the tool, the textual content of the tag can be required or not, and can have different meanings. The requirement depends on the type of the attribute:

- **ressource**: in that case the text field correspond to an Atelier B ressource, and the variable will be expanded into the value of the ressource. For instance:

```
• <toolParameter name="refinerFile"
  type="ressource">ATB*BART*RefinerFile</toolParameter>
```

- **exefile**: in that case, the text field is not used. The variable will be expanded to the path of a file configured by the user in the "Preferences" dialog.

- ```
<toolParameter name="editor" type="exefile" description="The editor" default="/usr/bin/xemacs" />
```

- **file**: the text field is not used, and the variable will be expanded to the path of a file configured by the user in the "Preferences" dialog.

- ```
<toolParameter name="myFile" type="file" description="tool configuration file" />
```

- **tool**: This type correspond to a tool, either provided by Atelier B, either provided by the extension. The text field is optional, and correspond to an Atelier B ressource. The variable is expanded into the full path to the corresponding tool. The tool can be an executable or a logic-solver file (*.kin). Example:

- ```
<toolParameter name="krt" type="tool" description="Logic Solver" default="krt">ATB*ATB*Logic_Solver_Command</toolParameter>
```

The tool is searched as follows: first, the content of the provided ressource (if any) is looked up. If it is empty or if no ressource is provided, the default value is taken. Then, the corresponding file is searched in the following directories

- The os-dependant directory of Atelier B ([bbin/win32](#) on windows, [bbin/linux](#) on linux, etc...)
- The external tool directory. If the tool is named "mytool", this directory will be [extensions/mytool](#)
- The os-dependant subdirectories of the external tool: [extensions/mytool/win32](#) on windows, [extensions/mytool/linux](#) on linux, etc...

## 2.3 temporaryFile

The **temporaryFile** tag is used to write a temporary file before running the command. This can be used in the case where the command takes an input file. The textual content of the **temporaryFile** tag correspond to a template file, where all the variables are replaced by their values. There can be as many **temporaryFile** tags as needed in the etool file.

The following attributes are recognised:

| Attribute | Optionality | Description                                                                                                                                                                           |
|-----------|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| name      | required    | the name of the variable that will hold the path to the created temporary file                                                                                                        |
| directory | optional    | the directory where the temporary file should be created. If no directory is specified, the system temporary directory will be used. Variables can be used to express this directory. |
| template  | optional    | A template for the name of the file. If provided, the name of the temporary file will be based on template                                                                            |

## 2.4 command

The `command` tag specifies the executable that should be started. One and only one `command` tag should be present in the extension file.

List of attributes:

| Attribute        | Optionality | Description                                                                                |
|------------------|-------------|--------------------------------------------------------------------------------------------|
| workingDirectory | optional    | The working directory of the started tool. Variables can be used to specify this directory |

## 2.5 param

The `param` tag specifies one and only one argument to the `<command>` tag. There should be as many `param` tag as there are arguments to the command. This is required to correctly handle spaces in directories.

The current list of recognised attributes is not documented yet, as it is very likely to change.

## 2.6 List of predefined variables

The following table shows all the predefined variables as well as their content:

| Variable          | Tool type | Description                                                                                        |
|-------------------|-----------|----------------------------------------------------------------------------------------------------|
| \${projectName}   | All       | The name of the currently selected project, or the project to which the selected component belongs |
| \${projectBdp}    | All       | The absolute path to the "bdp" directory of the project                                            |
| \${extensionsDir} | All       | The path to the extension directory                                                                |
| \${componentName} | Component | The name of the selected component                                                                 |
| \${componentPath} | Component | The absolute path to the selected component file                                                   |
| \${componentDir}  | Component | The absolute path to the directory where the component is located                                  |

# 07.

---

## 3 REFERENCES

---

## 3.1 Glossary

- **Component**: text-based model that could either be a specification (B abstract machine), a refinement or an implementation.

## 3.2 Bibliography

- **The B-Book: Assigning programs to meanings**, J.R. Abrial, Cambridge University Press (1996)
- **Modeling in Event-B: system and software engineering**, J.R. Abrial, Cambridge University Press (2010)
- **The B-Method: an introduction**, S. Schneider, Palgrave Macmillan (2001)
- **Program Development by Refinement: case-studies using the B-Method**, E. Sekerinski & K. Sere, Springer (1999)
- **Spécification formelle avec B**, H. Habrias, Lavoisier (2001)







Clearsy System Engineering

Parc de la Duranne  
320 Av. Archimède  
Les pléiades III Bat. A  
13857 Aix-en-Provence Cedex 3

Phone /Online

Phone : +33 (0)4 42 37 12 70  
Fax : +33 (0)4 42 37 12 71  
contact@clearsy.fr  
www.atelierb.eu