# Table of Contents

1. Change Log ........................................................................................................... 5
2. References ............................................................................................................. 6
3. Overview ................................................................................................................ 7
4. Operation ............................................................................................................... 8
   4.1. Getting Started ............................................................................................... 8
   4.2. Wizards ........................................................................................................... 8
   4.3. Rule Editor .................................................................................................... 10
   4.4. Workflow Editor ............................................................................................ 10
   4.5. Advanced Topics .......................................................................................... 11
A. License ................................................................................................................ 12
B. Release Notes ...................................................................................................... 13
1. Change Log
2. References

[1] JAXB
[2] MaduraObjects
[6] EPL 1.0
3. Overview

This Eclipse Plugin supports editing of Madura Rules and Madura Workflow files. The Madura Suite makes use of rules and workflow definitions. Unlike similar products it keeps these definitions in text files rather than maintaining them with a complex UI. But a helpful editor is always welcome and this plugin provides two of them. One for the workflow definitions and one for the rules definitions. There are also creation wizards for each to get you started, syntax checking and content assist while you edit, and an outline to help you navigate. These are just editors, they do not provide a runtime environment.
4. Operation

4.1. Getting Started

Details of setting up a project that uses the Madura Suite[5] are covered in documentation for MaduraObjects[2], MaduraRules[3] and MaduraWorkflow[4]. But the core requirements for using this plugin are:

• Eclipse Luna (4.4.0) through Eclipse Oxygen (4.7.3a) installed
• This plugin installed (from https://www.madurasoftware.com/madura-eclipse)
• A Java project
• An XSD file defining some objects

The XSD file needs to include some Madura-specific tags, but mostly it needs your objects defined in a way that JAXB[1] can generate Java classes from them using a maven plugin. But you don't actually need the classes generated for this Eclipse plugin to operate. This plugin reads the XSD file directly, not the Java classes.

4.2. Wizards

To create a rul or wrk file you can use the relevant wizards. You right-click on a location in the package explorer and pick New>Other, then look under 'Madura Wizards'.

![Figure (1) Wizards](image)

They both operate the same way so if you pick one you will see a dialog like this:
You need to adjust the rule file name to your liking and specify the XSD file the new file will find its data definitions in.

If you want to change the XSD file reference later use the File Properties (right click on the file in Package Explorer, then Properties>Madura).

Figure (2) WizardsRule

Figure (3) File Properties
Once the new file is created it will be opened in the relevant editor.

4.3. Rule Editor

The new file has some sample rules to help you get started but beware, this is just static content, it was not generated from your XSD file so the fields etc will not be correct. That means you may see syntax errors. But seeing the format at this stage is usually helpful.

```
/
*******************************************************************************
* Madura Rules
*******************************************************************************
formula: SampleObject "sample formula"
{
  decisionField = celsius > 30F;
}
rule: SampleObject "sample rule"
{
  if (size == "Small")
  {
    readonly(testing);
    activate(testing);
    amount = 10;
  }
}
constraint: SampleObject "sample constraint"
{
  testDouble > 100D;
}
```

The Outline window allows you to navigate to the rule you want to edit.

The Editor itself does the usual syntax checking you are used to in Eclipse. It will detect mis-spelled variable names and CTRL-SPACE will offer a drop down list of suggestions. The suggestions are always field names and functions.

4.4. Workflow Editor

Like the Rule Editor the Workflow Editor generates some static content to get you started and opens the file:

```
/
*******************************************************************************
* Madura workflow
*******************************************************************************
package nz.co.senanque.process.parser;

queue: name="OrderSupervisor"
  permission="SUPERVISOR"
  read-permission="STOCKSUPERVISOR";

process: Order "AcceptOrder" {
  try {
    message=MyMessage;
  }
  catch (error) {
    compute=orderCompute;
    retry;
  }
  catch (abort) {
```
compute=orderCompute;
retry;
}
}

Like the Rules Editor this one supports outline navigation, syntax checking and drop down suggestions.

4.5. Advanced Topics

If you are using custom functions in your rules (most people don’t) then you need to tell the plugin about those in the project properties (right click on the project in Package Explorer, then Properties>Madura)

![Figure (4) Properties](image)

The Classes path field should hold the directory your compiled java classes are placed into. For maven projects this will default to target/classes/. In the Class names field you just put a comma separated list of the classes that hold your custom functions. This will enable the validation code in the plugin to correctly validate your references to the custom functions without flagging them as errors.

Similarly Madura Workflow allows you to define external messages and compute modules. These are simpler for the parser than Rule Functions because they are (to the parser) just names. Supply a comma separated list of the names you use for messages and computes in the relevant boxes.

Finally the parser used to validate the files needs a buffer size setting, this defaults to 400 which is normally large enough but if you have very long comments in your files you might need to make it larger. Use the preferences page (Window>Preferences>Madura) for this.
A. License

The code specific to MaduraObjects is licensed under EPL 1.0 [6].
The dependent products have compatible licenses as specified in their pom files.
### B. Release Notes

You need Java 1.8 to compile this project.

<table>
<thead>
<tr>
<th>3.1.0</th>
<th>Reworked documentation.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Aligned plugin with Madura versions.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1.0</th>
<th>Initial version</th>
</tr>
</thead>
</table>