

Documentation

1 Introduction

To make a software available for use in Watchdog workflows, a new module has to be created. Watchdog already provides a helper script for creating the module XSD file and (optionally) a skeleton Bash script that only has to be extended by the program call. Nevertheless, this requires manually listing all parameters for the module. The new *ModuleMaker* GUI automatically extracts parameters and flags from a software help page or usage output to more conveniently create the corresponding module.

It uses a sets of regular expressions matching common help page/usage formats to parse the help page/usage of a software. Currently, 8 pre-defined regular expression sets are provided but users can also define new sets using the GUI and add them to the pre-defined list. When creating a module with the GUI, users may either choose one particular regular expression set explicitly or let *ModuleMaker* rank the regular expression sets based on how well they match the help page/usage. In the later case, the user can then examine the results of the n best-matching regular expression sets (with n user-defined) and choose the result they consider best. Subsequently, the user can correct errors in the automatic detection, add additional flags or parameters and modify or delete detected parameters. In a next step, existence checks for input files or directories can be added and return parameters for the module can be defined.

Once the user is finished, *ModuleMaker* creates the module XSD file and a wrapper Bash script for the software. This wrapper script enforces that required software is installed, parses parameters, verifies that mandatory parameters are set, performs existence checks on required input files and directories, executes the program, performs default error checks after execution and sets values of output parameters. Optionally, a project file (*.wm) can be saved that allows reloading and modifying modules created with the *ModuleMaker* at a later time.

2 Requirements

ModuleMaker is a Java based application and available as Java Archive (*.jar). It requires Java 11 and JavaFX 11 or higher. A Bash script is available for starting the GUI on Unix-based operating systems (e.g. Linux, MacOS). Please refer to this script how the Java call has to be constructed to include the JavaFX libraries.

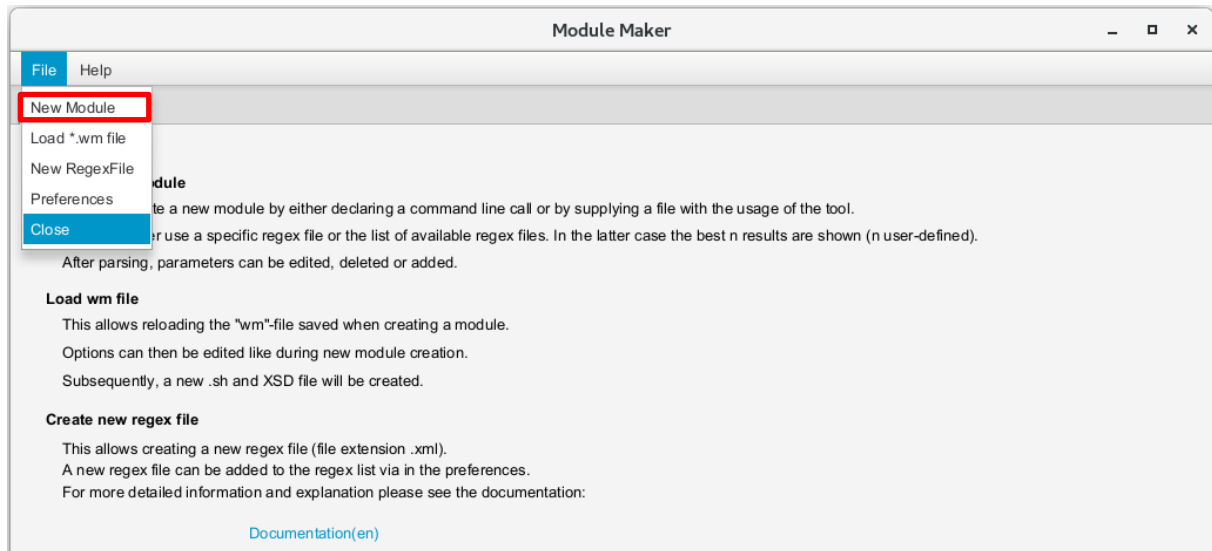
Please note: the following directories have to be contained in the directory *ModuleMaker* is started from:

- the directory containing the module templates ("moduleTemplates/")
- the directory containing the regular expression files ("RegexFiles/")

The jar file, Bash script, required directories (in directory `distribute/`) and source code (in directory `src/`) are available at <https://github.com/watchdog-wms/moduleMaker>.

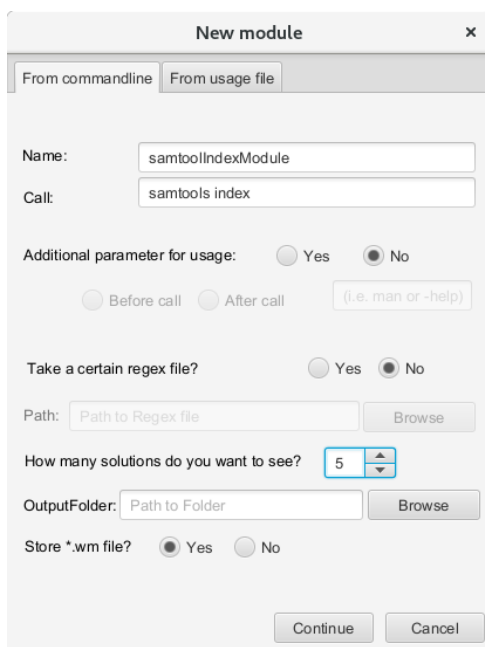
3 Using *ModuleMaker*

3.1 Create a new module



To create a new module, you have to click on "New Module" in the "File" menu. A new window will be opened that offers you two slightly different ways to start module creation. The tab "From commandline" can be used to parse options and flags of the software directly from its command-line call for the help pages (e.g. `gzip --help`). The second tab named "From usage file" allows to use a file as input that contains the usage description of the software.

3.1.1 Create a new module using the command-line call of the software



- **Name:** Specifies how the module is named; no special characters are allowed.

- **Call:** Determines how the software will be called when the module is used. Enter the call exactly as you would start it on the command line (with interpreter). For example, if you want to start a jar, you have to prefix the call with "java -jar". The path to binaries must be contained in the "PATH" environment variable.

- **Additional parameter for usage:** There are tools that need another parameter to show the usage. In many cases it is "-h" or "--help" after the call. You can choose whether that parameter is entered before (e.g. "man") or after the call (e.g. "--help").

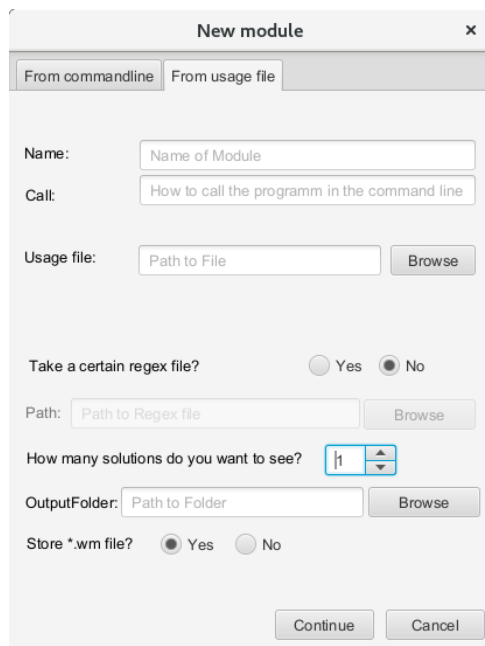
- **Take a certain regex file:** Specifies if a specific regex file should be used. This is a file defining how the help page is parsed (these files can be created with the "New RegexFile" entry in the menu). By default, all available regex files will be

used to parse the input and ranked on how well they match the help page.

- **How many solutions do you want to see:** Specifies how many of the best ranked solutions should be displayed.
- **OutputFolder:** The directory in which the resulting module files will to be saved.
- **Store *.wm file:** Optionally, a project file (*.wm) can be saved that allows reloading and modifying modules created with the *ModuleMaker* at a later time.

By clicking on "Continue" you can then proceed to parsing the help page/usage.

3.1.2 Create a new module using a file as input

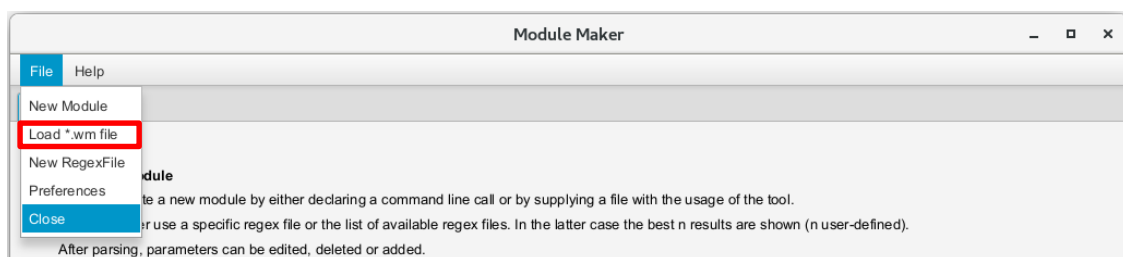


Most of the options are the same as described before. Instead of specifying additional parameters that are required to obtain the usage output from the command line call, a path to a file must be set.

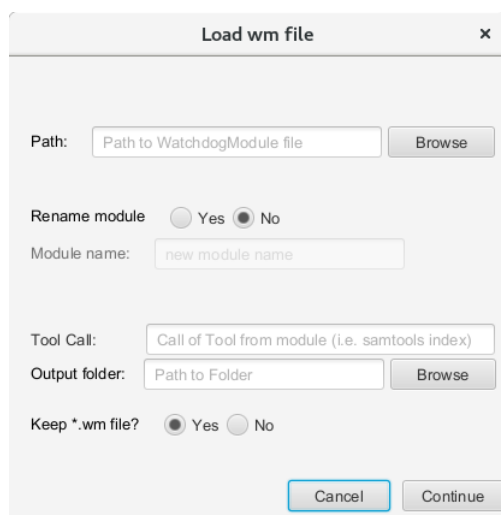
- **Usage File:** Specifies the file containing the usage description that should be parsed by *ModuleMaker*.

3.1.3 Modify already saved module

If you want to make changes to a previously created module, you can load a previously saved project file (*.wm) by selecting "Load *.wm file" in the "File" menu. Afterwards you can continue to adjusting the parameter and flags as usual and save it as a new module.



This opens a window with the following fields:



- **Path:** The *.wm file to be loaded.
- **Rename module:** If the module should be named differently than previously, a new module name can be specified.
- **Tool Call:** The command-line call used in the generated Bash script.
- **OutputFolder:** Location where the newly generated module will be saved.
- **Keep *.wm file:** Defines if the updated project file should be saved.

3.1.4 Customize the detected options

After either creating a new module or loading a project file, the summary screen opens up to display all parsed options.

All detected options are listed with identified name, short option, long option, type, minimal and maximal allowed occurrences, call type, default value, restriction and description. Empty values are indicated by "-".

- **Name:** Name of the option in the resulting module XSD file. It should be noted that not all characters are suitable. For example, special characters such as "@,;," are not allowed. Additionally, the name cannot consist only of a single number.
- **shortOpt:** The short option that was parsed from the usage. Usually it is a single character.
- **longOpt:** The long option that was parsed from the usage. In most cases this will also be used as option name.
- **Type:** Determines what type of input is valid. Possible types are: integer, double, string, boolean, AbsoluteFilePath and AbsoluteFolderPath. If the type could not be inferred from the help page, it is set to string by default.
- **Min Occurence:** Specifies how often this parameter has to be used at minimum. If 0 is specified, the parameter is optional.
- **Max Occurence:** Specifies how often this parameter can be used at maximum. If any number is possible, "unbounded" must be specified.
- **Call Type:** With call type you can specify how the parameter is passed to the software when calling it in the module. You can choose between "short", "long" and "none". The latter is required if parameters are passed to the software without specifying an option. This is often used when input files are provided as arguments (e.g. gzip [FILE]).
- **Default:** The default value is used if a value for an optional parameter is not set explicitly in the workflow.
- **Restriction:** Sets further restrictions to the parameters using regular expressions.
Examples:
 - "red|blue": only "red" and "blue" are allowed
 - "[a-zA-Z]+": only letters allowed
 - "(10|[1-9])": only numbers 1-10 are allowed
- **Description:** Contains the descriptions of the options.

Index	Name	shortOpt	longOpt	Type	Min Occurence	Max Occurence	Call Type	Default	Restriction	Description
0	b	b	-	string	0	1	short	default	-	Generate BAI-format index for BAM files
1	c	c	-	string	0	1	short	-	-	Generate CSI-format index for BAM files
2	m	m	-	integer	0	1	short	14	-	Set minimum interval size for CSI indices to 2*INT
3	noName1	@	-	integer	0	1	short	-	-	Sets the number of threads
4	inbam	-	-	string	1	1	-	-	-	-
5	bc	-	-	string	0	1	-	-	-	-
6	mINT	-	-	string	0	1	-	-	-	-
7	outindex	-	-	string	0	1	-	-	-	-

< 1 of 4 >

Delete Selected New Option Continue

You can switch between different suggested solutions, which were created using different regex files, by clicking on the buttons "<" and ">" on the bottom left.

In the overview you can delete, edit or add options. To delete options, you have to select the options to be deleted and then click on "Delete Selected".

New options can be added by clicking on "New Option".

New Option

Name:

shortOpt:

longOpt:

Type:

Min Occurence:

Max Occurence:

CallType:

Default:

Restriction:

Description:

In the dialog you can enter all the values you normally see in the overview.

The only mandatory field is the name for the new option. Missing values will be replaced with "-".

To edit options, just select the field and change the value. You have to confirm your change by pressing return.

If you changed something and go to the next page by clicking "Continue", you will be asked if you want to save the changes.

Save changes?

On the next page you can make further adjustments.

File Help

Which files or folders should exist?

☒ inbam

☐ outindex

Seperation of parameter maxOccurence of 2+

call options:

call of short options:

call of long options:

call of short options (flag):

call of long options (flag):

Return type elements:

returnTyp...	Paramete...	
No content in table		

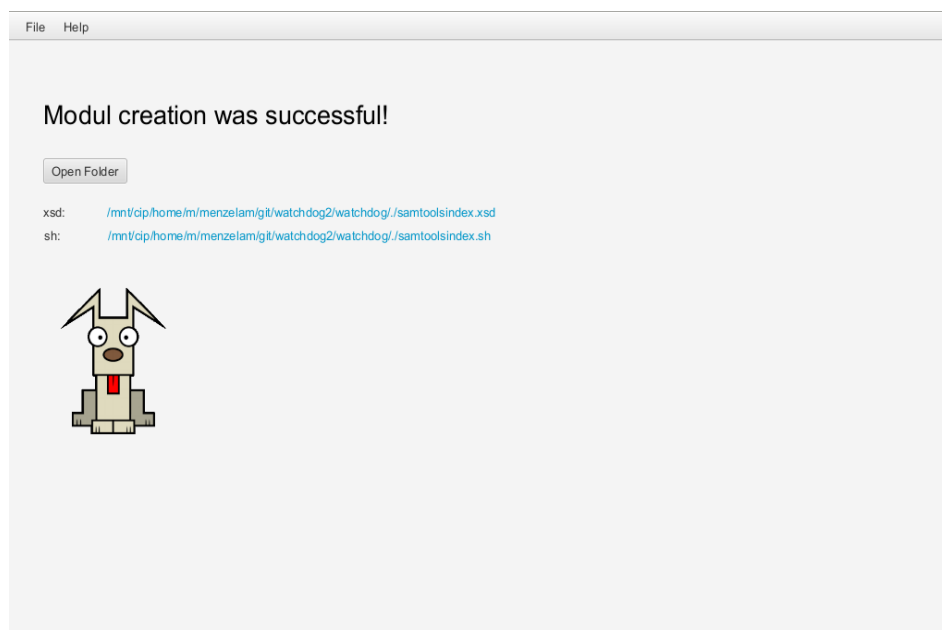
If files and directories have been specified, you can choose whether the Bash script should enforce existence of these files or folders or not.

You also have to specify how exactly the short and long options are passed to the software. Placeholders for the option name and parameter value are "option" and "parameter", respectively. Normally this is pre-filled with the correct values and just needs to be checked.

In addition, you can specify how multiple values of the same parameter are passed to the software. Usually, multiple values are separated by comma, but you also can specify a different separator.

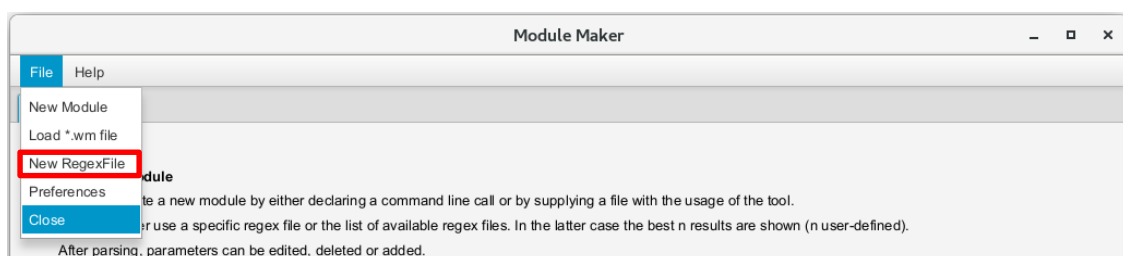
At the bottom of this window, you can define return values of the module.

If you now click on "Continue", the module XSD file and Bash script will be created. The path to these files is shown on the final page. You can either open the module folder using your default file browser by clicking on "Open Folder" or open the files with the default program associated with these file endings by clicking on the file links.



3.2 Define a new regular expression set

To create a new regex file, you have to go to the menu and click on "New RegexFile".



A dialog then opens to create a new set of regular expressions. Format descriptions and example values are shown in gray text in individual fields. Placeholders for option names and parameter values are again "option" and "parameter", respectively.

The following fields are available:

- **Start of Usage line:** To distinguish the usage line from other text, the start of the usage line is required. In most usage pages it starts with: "Usage:" (for example in the form: Usage: samtools index [-bc] [-m INT] <in.bam> [out.index])
- **Labeling of required parameter in usage:** The required arguments are often specified in the usage line in a specific form, e.g. in angle brackets (e.g. <option>). To indicate this, enter the term "option" surrounded with the appropriate characters into this field (e.g. "<option>" for the above example or " option " if spaces on both sides are indicative of a required parameter).
- **Labeling of optional parameter in usage:** Similar to required arguments, optional arguments are also marked in a particular way. Often square brackets are used (e.g. [option]). Again, enter the term "option" surrounded with the appropriate characters (e.g. "[option]").
- **Short option:** Here you have to specify how short options are specified for parsing them from the usage (left field) and for passing a parameter to the software (right field). For the left field, it is sufficient to describe the option without considering parameters. As before, the keyword recognized by *ModuleMaker* is "option". The right field must contain the placeholder "parameter" for the parameter values. Often, the short option has the form: "-option" for parsing and "-option parameter" for passing the parameter to the software. The same information has to be specified for flags in the two fields below. Flags are boolean options that take no parameters.
- **Long option:** Here you specify how the long options is specified (see short option for description). Often the long options are described in the following form: "--option" and for the second field "--option parameter". However, long options are often also specified in the following form: "--option=" and "--option=parameter". In that case it is important that the "=" immediately follows the "option" keyword in the left field.
- **Separator of short and long options:** Defines how short and long options are separated in the usage page if both are indicated. Keywords are "short" and "long". An example entry for this field would be "short, long".
- **Type:** Detection of six different parameter types can be defined.

- **Labeling of type:** Here it is specified how the types are marked (e.g. in angle brackets). The required keyword is "type".
- **Typedefinition of integer:** Defines how the type integer is called in the usage (e.g. "INT").
- **Typedefinition of string:** Defines how the type string is called in the usage (e.g. "string").
- **Typedefinition of boolean:** Defines how the type boolean is called in the usage (e.g. "bool").
- **Typedefinition of double:** Defines how the type double is called in the usage (e.g. "FLOAT"). Please note that the types float and double are not differentiated in Watchdog.
- **Typedefinition of file paths:** Defines how the type file is called in the usage (e.g. "FILE").
- **Typedefinition of folder paths:** Defines how the type folder is called in the usage (e.g. "FOLDER").
- **Default:** If default values are specified, these may be indicated in a particular way. For example, they are often prefixed with "Default" and displayed in parentheses. The keyword here is "def" (e.g. "[Default: def]").
- **Description:** Descriptions can also be indicated in a specific way, e.g. in angle brackets. The keyword here is "des".

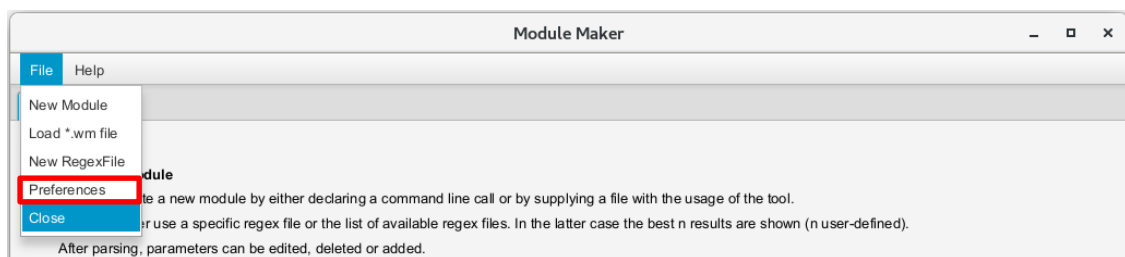
At least the usage line and a regex for either short and/or long option must be specified.

When saving the regex file, the file name extension has to be ".xml" for it be recognized.

If you want to add the created regex file to the list of regex files used by default, you can use the settings menu ("Preferences").

3.3 Preferences

The settings can be reached in the menu by clicking on "Preferences". The dialog that will be opened contains two tabs named "Regex" and "Debug".

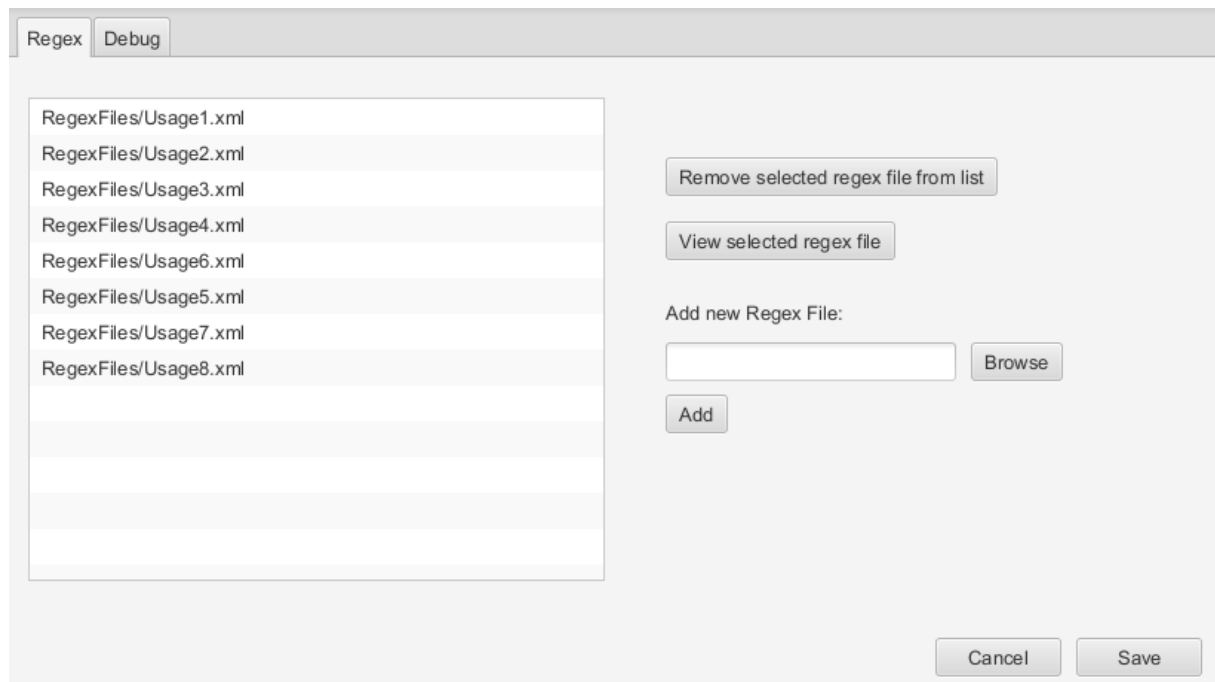


3.3.1 Regex files tab

In the "Regex" tab you can adjust the regex list that is used by default to parse a software help page/usage. You can delete regex files or add new ones.

The regex files currently contained in the list are shown on the left side. You can select multiple files in the list to be deleted.

You can also display the definition of a regular expression set in a separate window.



3.3.2 Debug tab

You can enable the debug output in the "Debug" tab of the settings.

