WorkflowMaker text processing samples user guide

For WorkflowMaker version **1.0.5** and later
Windows 10 & 11
64-bit only

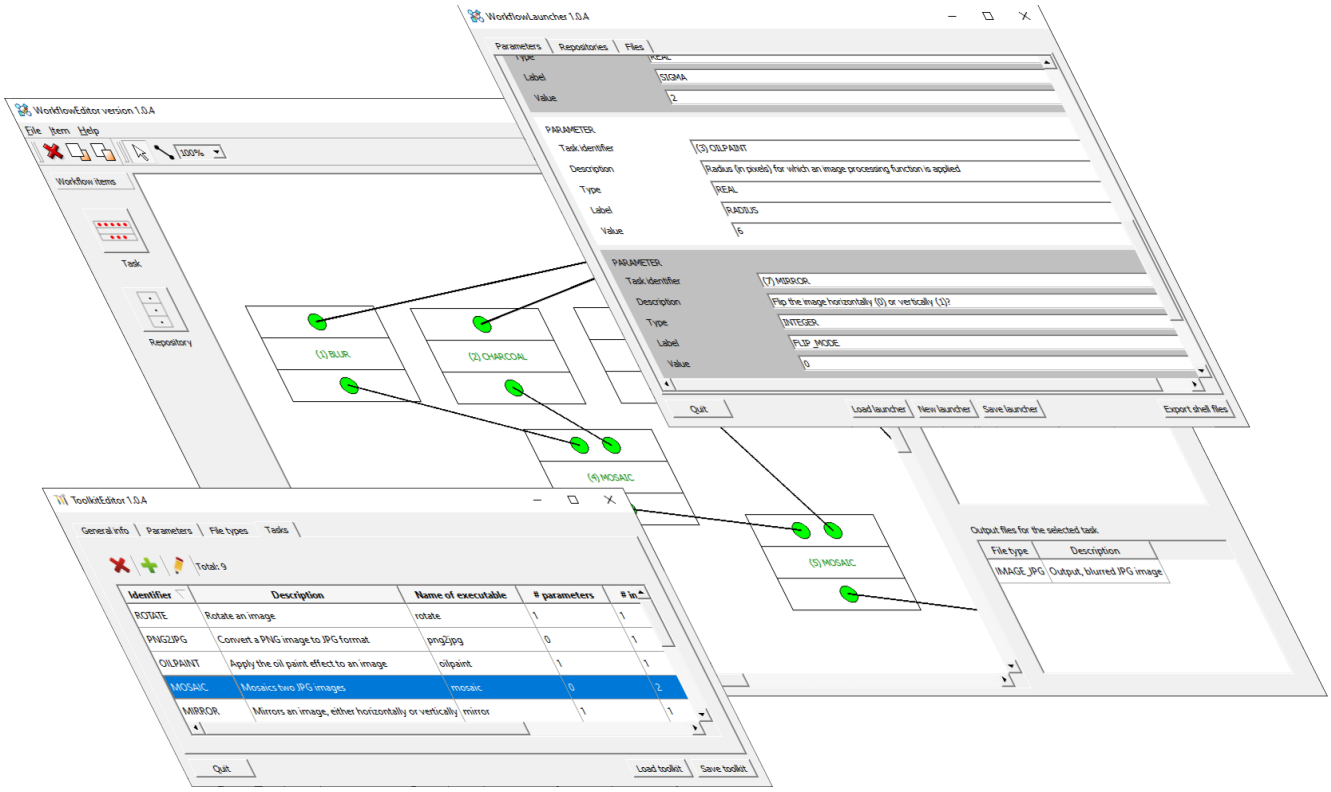

Table of contents

List of figures

List of tables

1 Introduction

This document presents a sample set of tools compatible with WorkflowMaker targeted at (very simple) text file processing.

The tools included in this sample package do not intend to perform any useful task; these have been designed to be deliberately simple so analyzing its source code will let potentials users learn how to create or modify their own console applications to meet the requirements set by WorkFlowMaker. Furthermore, this example toolkit will also serve so that users can start practicing the visual design (and execution) of workflows, since a complete (WorkflowMaker files and executable tasks) toolkit is available.

Note that this document is not a user guide about WorkflowMaker, but just the definition of a complete (although very simplistic) toolkit. For information about how to use WorkFlowMaker, please, refer to the official user's guide.

2 The text file processing example toolkit

The next sections will describe (tasks, list of file types, list of parameters) the text files processing sample toolkit.

A very simple toolkit consisting of some console applications (tasks) will be briefly explained. The field of application of the toolkit is that of text files processing. All the tasks included take one or two input files, apply some kind of change (such as concatenating the inputs) and save the result in an output text file.

Due to the simplicity of the example, only five tasks, including two keyboard parameters and just a single file type, "text file". This, however, is not the usual situation; in real-world applications, several kinds of files are involved, and the number of keyboard parameters tend to be much higher; the unique reason for keeping such a low profile in this example is facilitate the comprehension of the WorkFlowMaker tools avoiding the use of a complicated toolkit.

2.1 The file types

As stated above, a single file type is uses across the toolkit. Table 1 describes the "text file" file type.

Identifier	Description	Default extension
TXT	A text file	.txt

Table 1: The list of file types in the text file processing example toolkit.

2.2 The keyboard parameters

The list of unique parameters is shown in Table 2.

Identifier	Description	Data type
CUT_TAIL_MAX_LINES	Maximum number of lines to preserve at the beginning of some files when its tail is about to be removed. It must be greater than 0.	Integer
INTERLEAVE_COPY_QUEUE	Flag stating if the tail of the longest file must be copied (1) or not (0) when interleaving the lines of two files and the shorter one is exhausted.	Integer

Table 2: The list of unique parameters in the text file processing example toolkit.

The list of tasks is described in Table 3 below. Note the use of the identifiers in Table 2 above when referring to keyboard parameters. No references have been included to file types since there is only one making any identification unnecessary.

2.3 The tasks

Table 3 on page 6 briefly summarizes the tools included in the toolkit. In the case of keyboard parameters, the **boldfaced** text stands for the identifier of such parameters; for both input and output files, the **boldfaced** text corresponds to the labels that will be used in the option files. As stated above, no indication about the kind (type) of file exist, since this is a very simplistic toolkit and all files are text files.

Figure 1 on page 7 depicts how ToolkitEditor would define task INTERLEAVE. Note the coincidence of parameters, input and output files between such figure and Table 3.

Identifier	Description	Parameters	Input files	Output files
APPEND	Concatenates (appends) two files.	None	**APPEND_INPUT_FILENAME_1** The first input file to concatenate. **APPEND_INPUT_FILENAME_1** The second input file to concatenate.	**APPEND_OUTPUT_FILENAME** The output file, concatenating both inputs.
CUT_TAIL	Remove the tail of some input file, leaving only some lines at its beginning.	**CUT_TAIL_MAX_LINES** The number of lines at the beginning of the input file that must be preserved.	**CUT_TAIL_INPUT_FILENAME** The input file whose tail will be cut.	**CUT_TAIL_OUTPUT_FILENAME** The output file, keeping only the requested number of lines.
INTERLEAVE	Merges two input files, interleaving its lines. Optionally, may copy the tail of the longest file (when no more lines are available in the shorter one)	**INTERLEAVE_COPY_QUEUE** Flag stating whether the tail of the longest file must be copied to the output when the shorter file includes no more lines to interleave. Set this value to 1 to copy the tail, or to 0 to avoid such a copy.	**INTERLEAVE_INPUT_FILENAME_1** The first input file. **INTERLEAVE_INPUT_FILENAME_2** The second input file.	**INTERLEAVE_OUTPUT_FILENAME** The output file including the interleaved lines of the two input files and, depending the user's decision, the tail of the longest file.
REVERSE	Reverse all the lines in some input file. For example: "Hello, Dolly" becomes "ylloD ,olleH".	None.	**REVERSE_INPUT_FILENAME** The input files whose lines must be reversed..	**REVERSE_OUTPUT_FILENAME** – The output file, with the lines reversed.
TO_UPPERCASE	Capitalizes all letters in the input file.	None	**TO_UPPERCASE_INPUT_FILENAME** The input file whose letters will be capitalized.	**TO_UPPERCASE_OUTPUT_FILENAME** The output file. In this file, all letters are capitalized.

Table 3: Characterization of the tasks in the example text files processing toolkit.

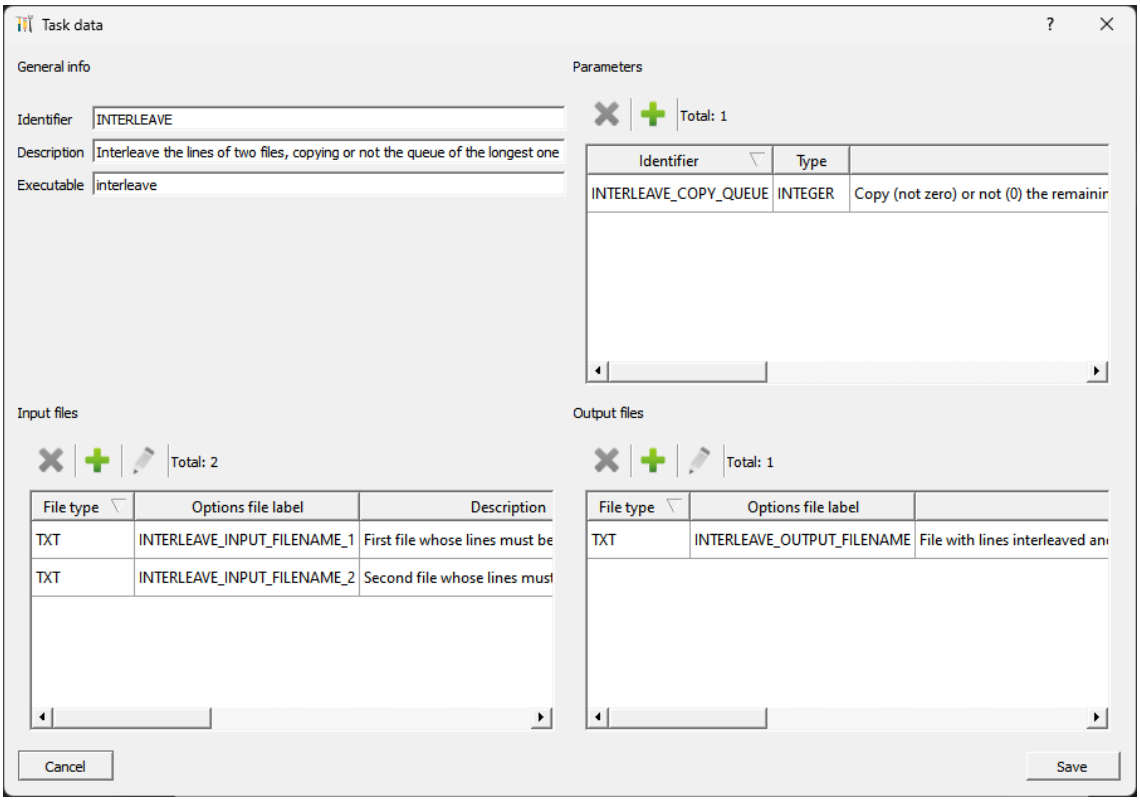

Figure 1: The definition of task INTERLEAVE using the ToolkitEditor tool.

2.4 The toolkit sample files

Three sample WorkFlowMaker files have been included:

- text_file_processing_toolkit.xml – The definition of the toolkit as defined in sections 2.1 to 2.3, tables 1 to 3 and Figure 1. This file may be opened using the tool ToolkitEditor.
- text_file_processing_workflow_01.xml – A simple workflow relying on said toolkit. May be displayed / edited using WorkflowEditor.
- text_file_processing_workflow_02.xml – A more complicated, but still very simple workflow relying on the same toolkit. As in the previous workflow, use WorkflowEditor to display or edit this workflow.

Note that there are no launcher sample files in this example. The reason is these include the actual paths to some repositories and that these depend on the computer used to create them.

Warning: it is better to copy the sample files to some local directory to edit / modify them safely. These samples reside in folders with no writing privileges, so it would be no possible to save any changes made when editing them.

Figure 2 depicts the workflow defined by text_file_processing_workflow_01.xml.

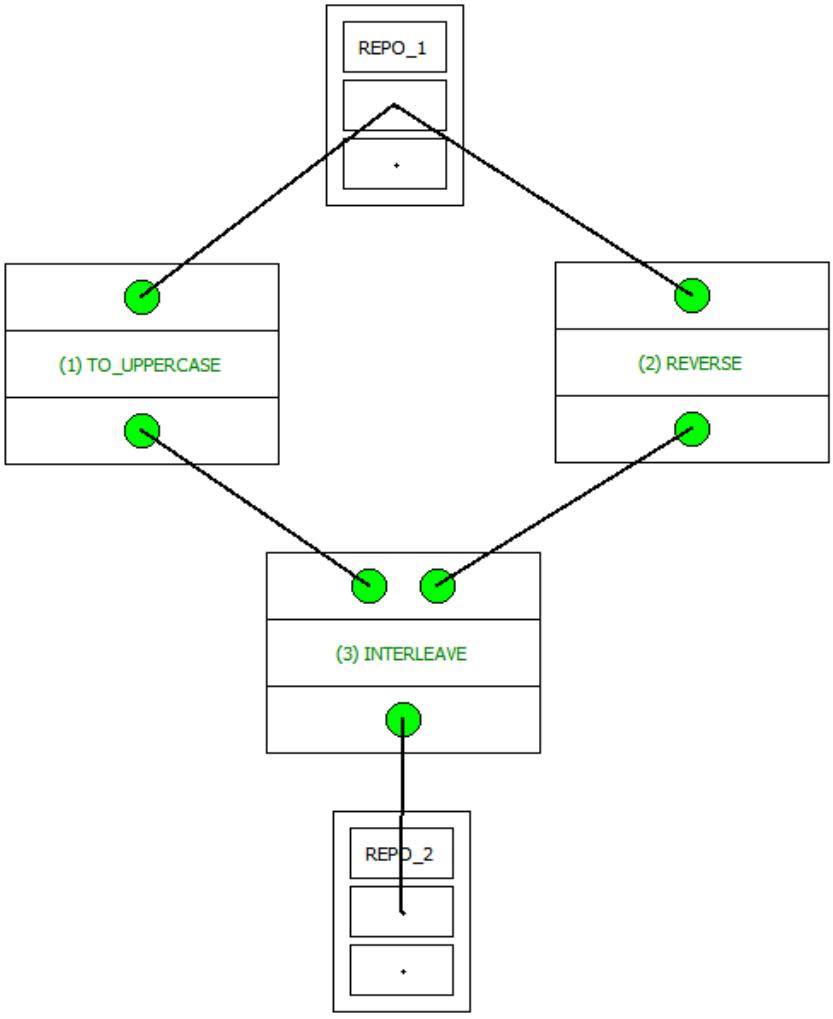

Figure 2: WorkFlowEditor depiction of the workflow in text_file_processing_workflow_01.xml.

This workflow takes two input files; the letters included in the first one are changed to uppercase, while the lines in the second one are reversed. Then, the lines of the two prior results are interleaved.

Figure 3 on page 9 shows the second workflow included with the samples, ext_file_processing_workflow_02.xml. In this case, two input files are concatenated; then the letters in the result and changed to uppercase; later the lines are reversed and, finally, only some of the them at the beginning of the latest intermediate result are preserved, since the tail of this file is removed. Note that, in this case, the result of the REVERSE task is not only used as input of CUT_TAIL but it is also preserved copying it to the output repository.

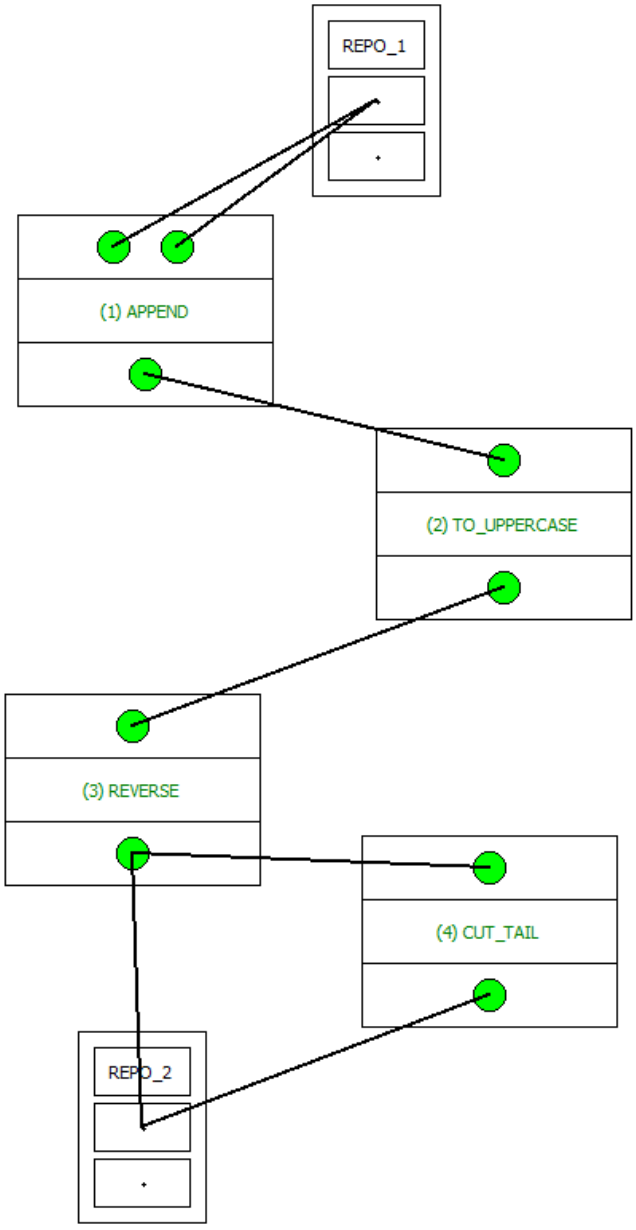

Figure 3: WorkFlowEditor depiction of the workflow in text_file_processing_workflow_02.xml.

Figure 4 shows the Windows script generated by ToolkitEditor for this workflow. The paths shown in the figure have been set by the user using said tool.

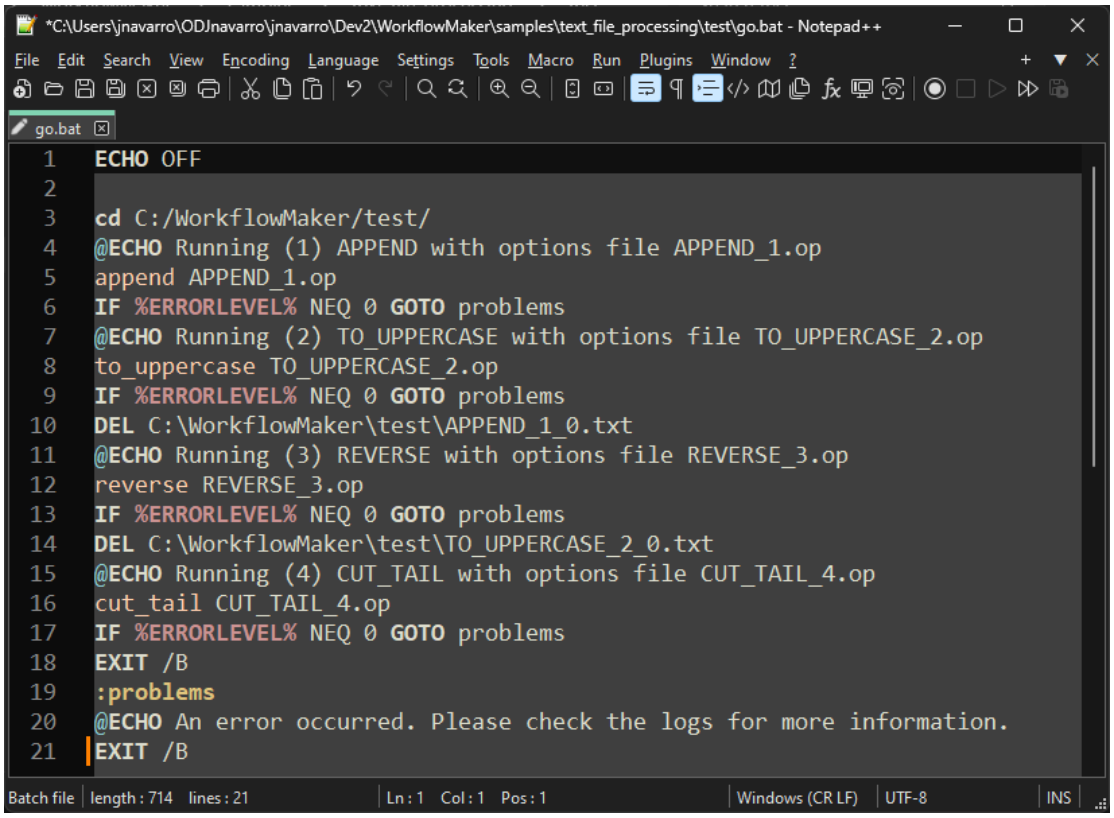

Figure 4: The Windows script created by ToolkitLauncher for the second sample workflow.