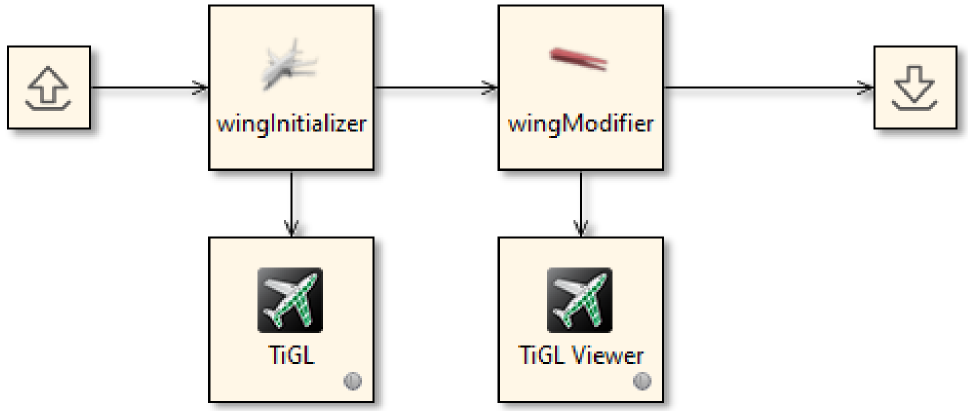
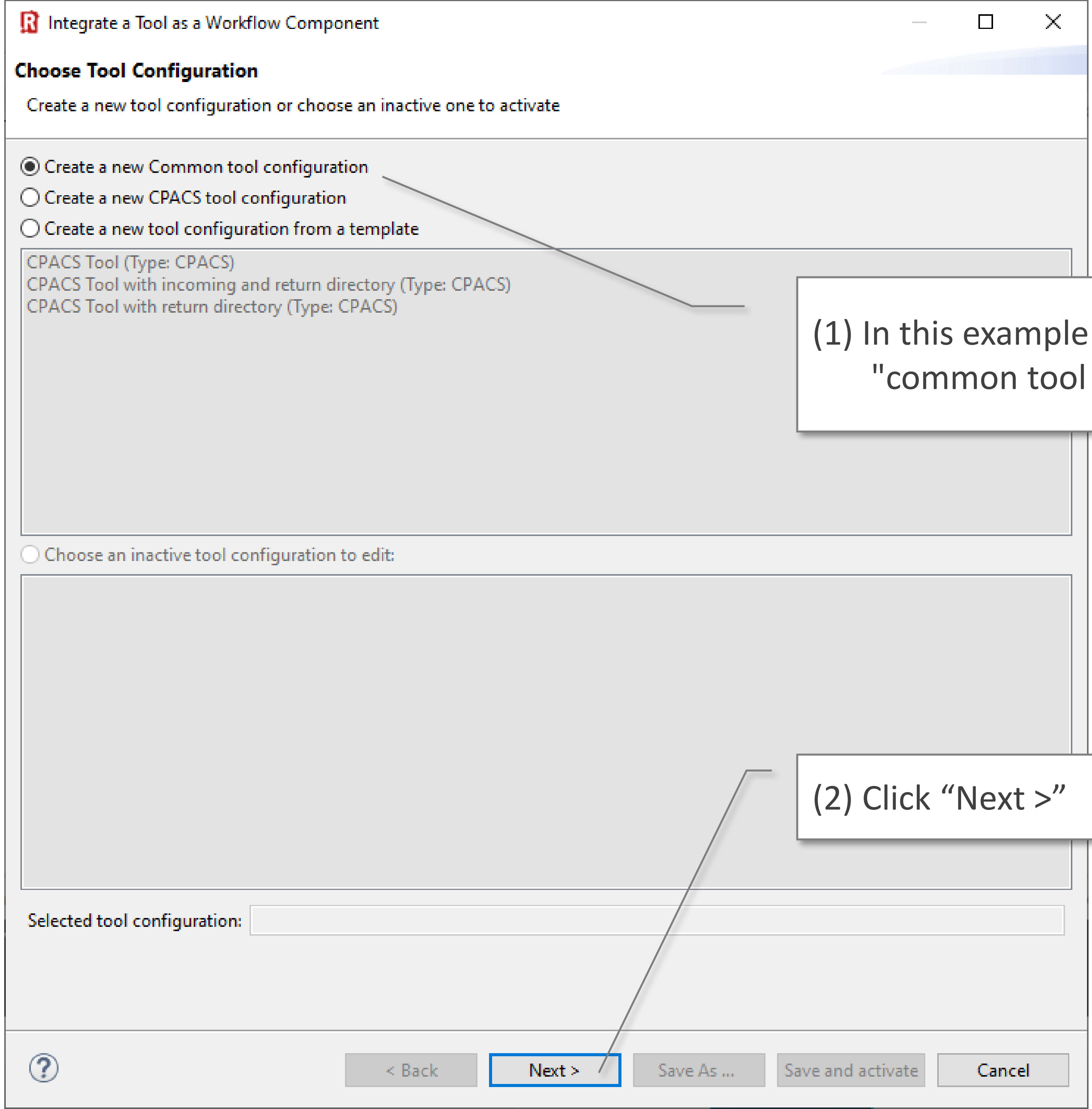
RCE tool integration

wingModifier




Integrate a Tool as a Workflow Component

Choose Tool Configuration

Create a new tool configuration or choose an inactive one to activate

(● Create a new Common tool configuration)
○ Create a new CPACS tool configuration
○ Create a new tool configuration from a template

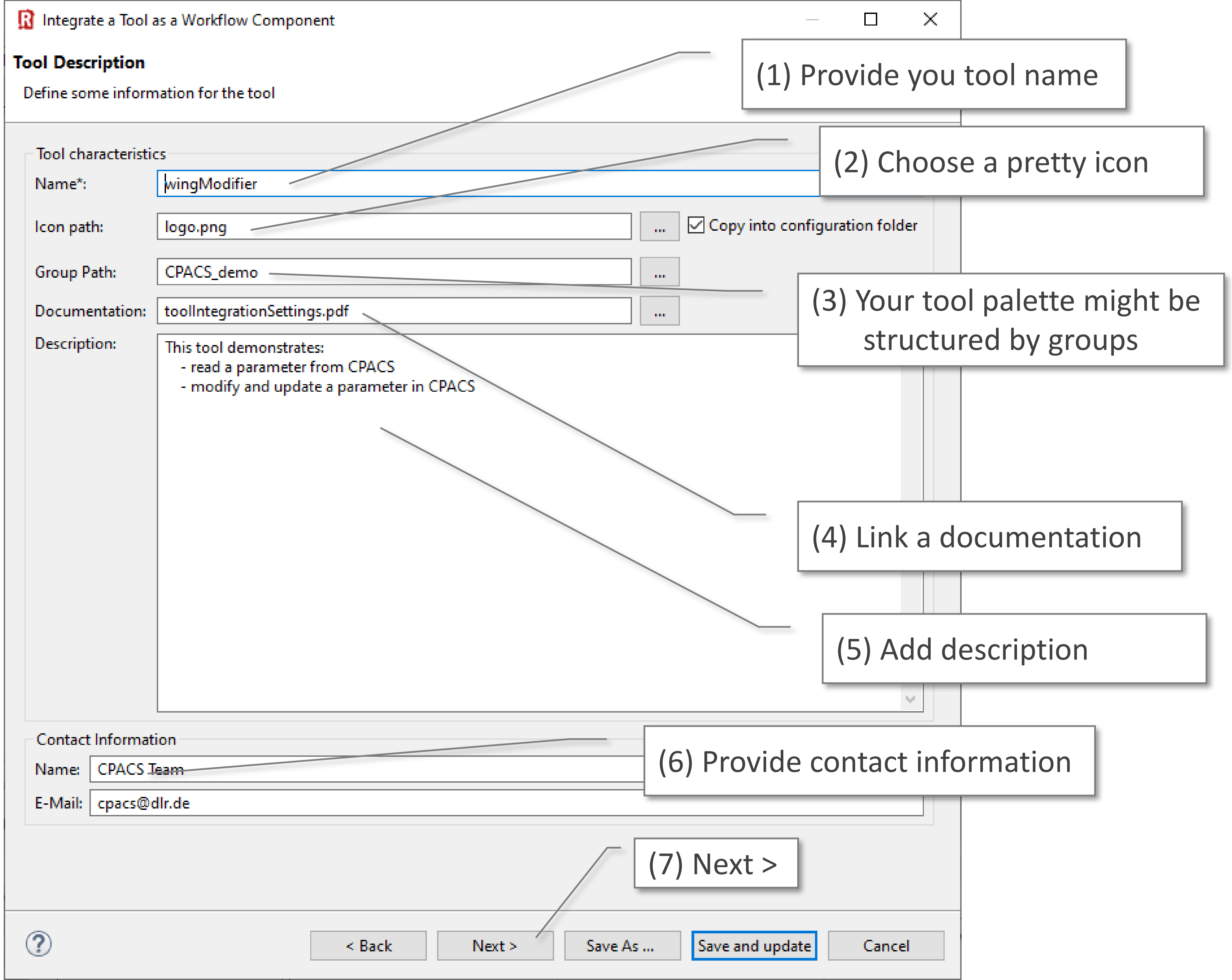
CPACS Tool (Type: CPACS)
CPACS Tool with incoming and return directory (Type: CPACS)
CPACS Tool with return directory (Type: CPACS)

○ Choose an inactive tool configuration to edit:

Selected tool configuration:

(1) In this example we use the "common tool configuration".

(2) Click “Next >”



Integrate a Tool as a Workflow Component

Tool Description

Define some information for the tool

Tool characteristics

Name*: wingModifier

Icon path: logo.png

☑ Copy into configuration folder

Group Path: CPACS_demo

Documentation: toolIntegrationSettings.pdf

Description:

This tool demonstrates:
- read a parameter from CPACS
- modify and update a parameter in CPACS

Contact Information

Name: CPACS Team

E-Mail: cpacs@dlr.de

(1) Provide you tool name

(2) Choose a pretty icon

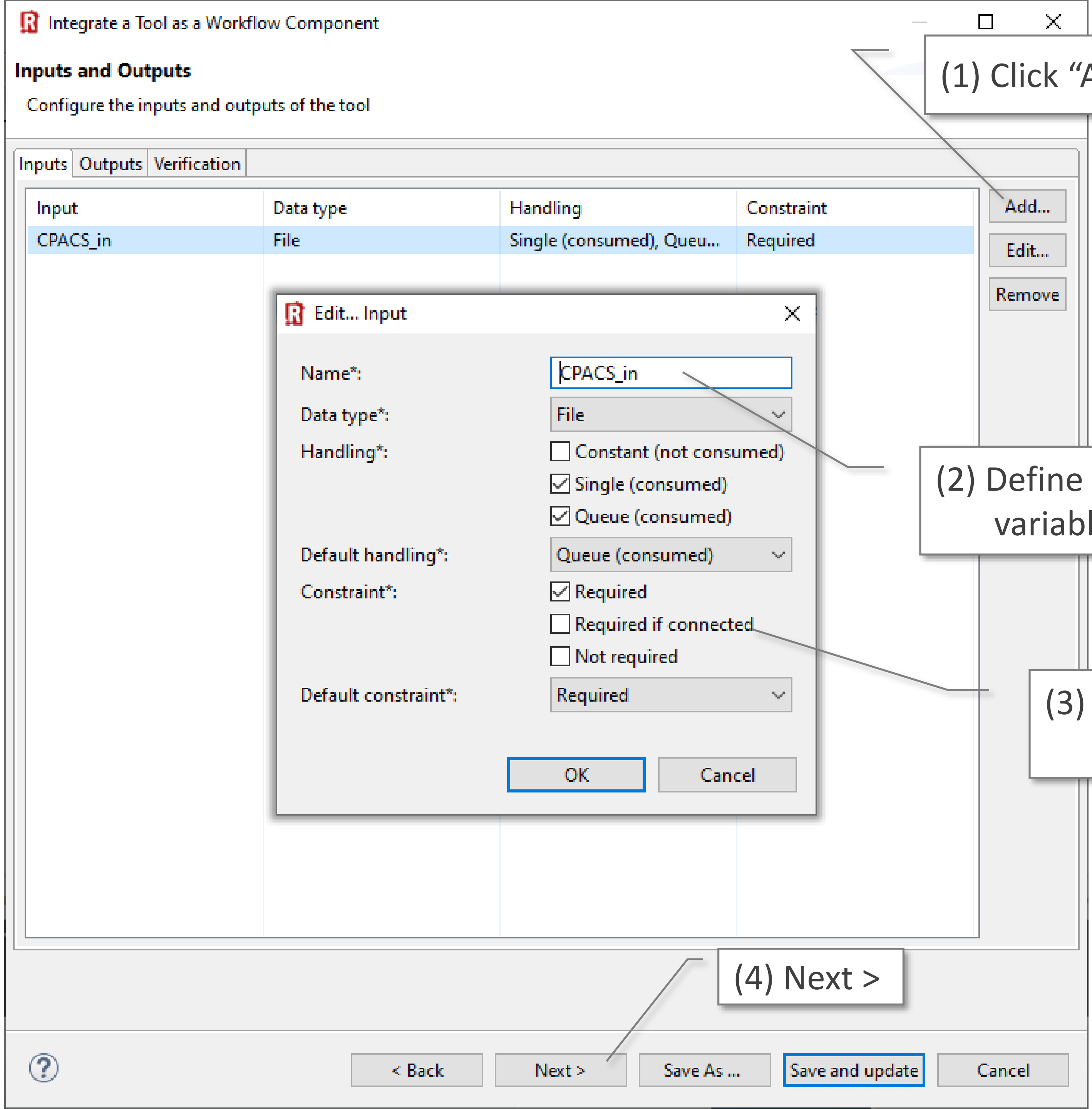
(3) Your tool palette might be structured by groups

(4) Link a documentation

(5) Add description

(6) Provide contact information

(7) Next >



Integrate a Tool as a Workflow Component

Inputs and Outputs

Configure the inputs and outputs of the tool

Inputs | Outputs | Verification

Input	Data type	Handling	Constraint
CPACS_in	File	Single (consumed), Queu...	Required

Edit... Input

Name*: CPACS_in

Data type*: File

Handling*: [ ] Constant (not consumed) [✓] Single (consumed) [✓] Queue (consumed)

Default handling*: Queue (consumed)

Constraint*: [✓] Required [ ] Required if connected [ ] Not required

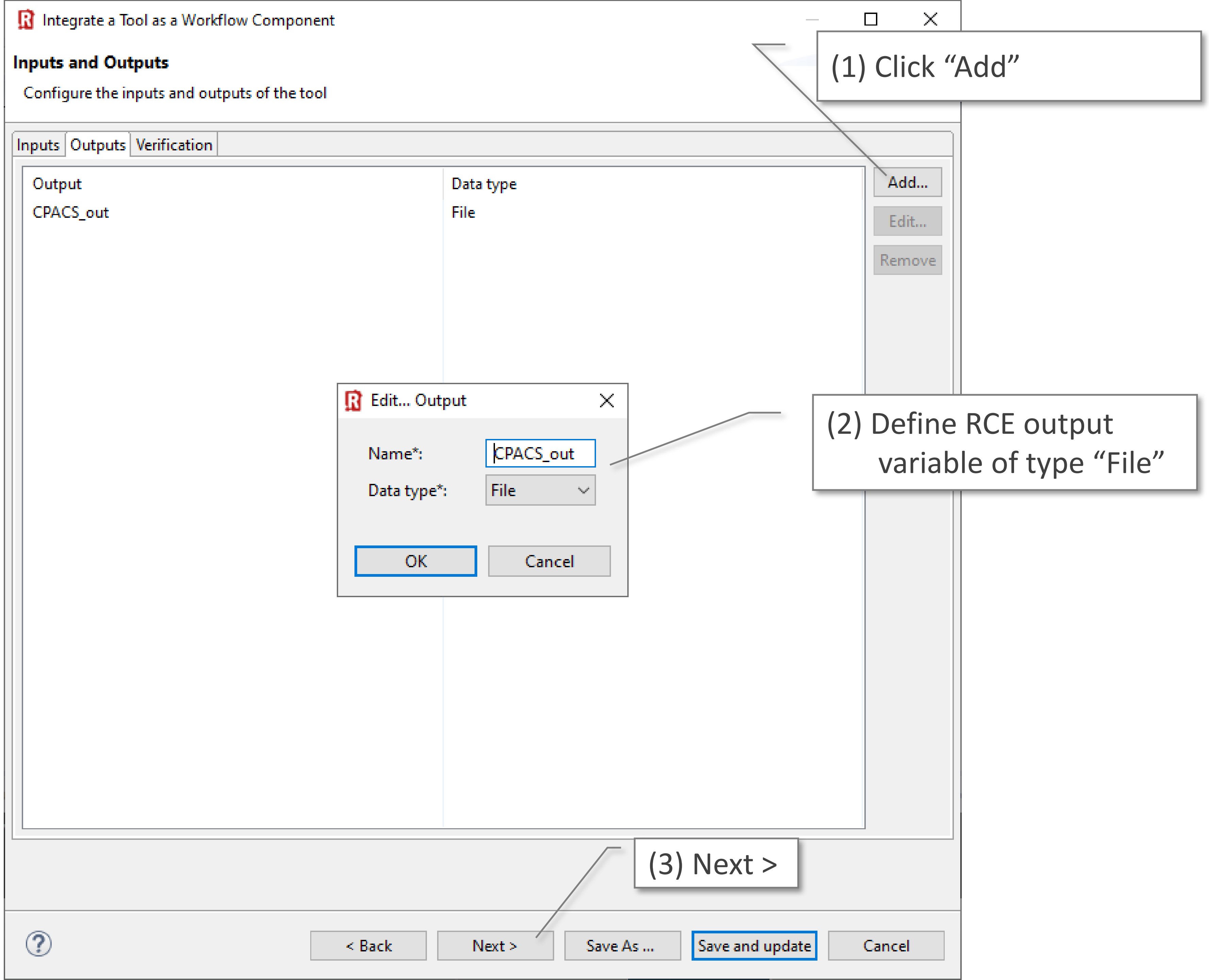
Default constraint*: Required

(1) Click “Add”

(2) Define RCE input variable of type “File”

(3) Choose these file handling settings and click “OK”

(4) Next >


Integrate a Tool as a Workflow Component

Inputs and Outputs

Configure the inputs and outputs of the tool

Output	Data type
CPACS_out	File

Edit... Output

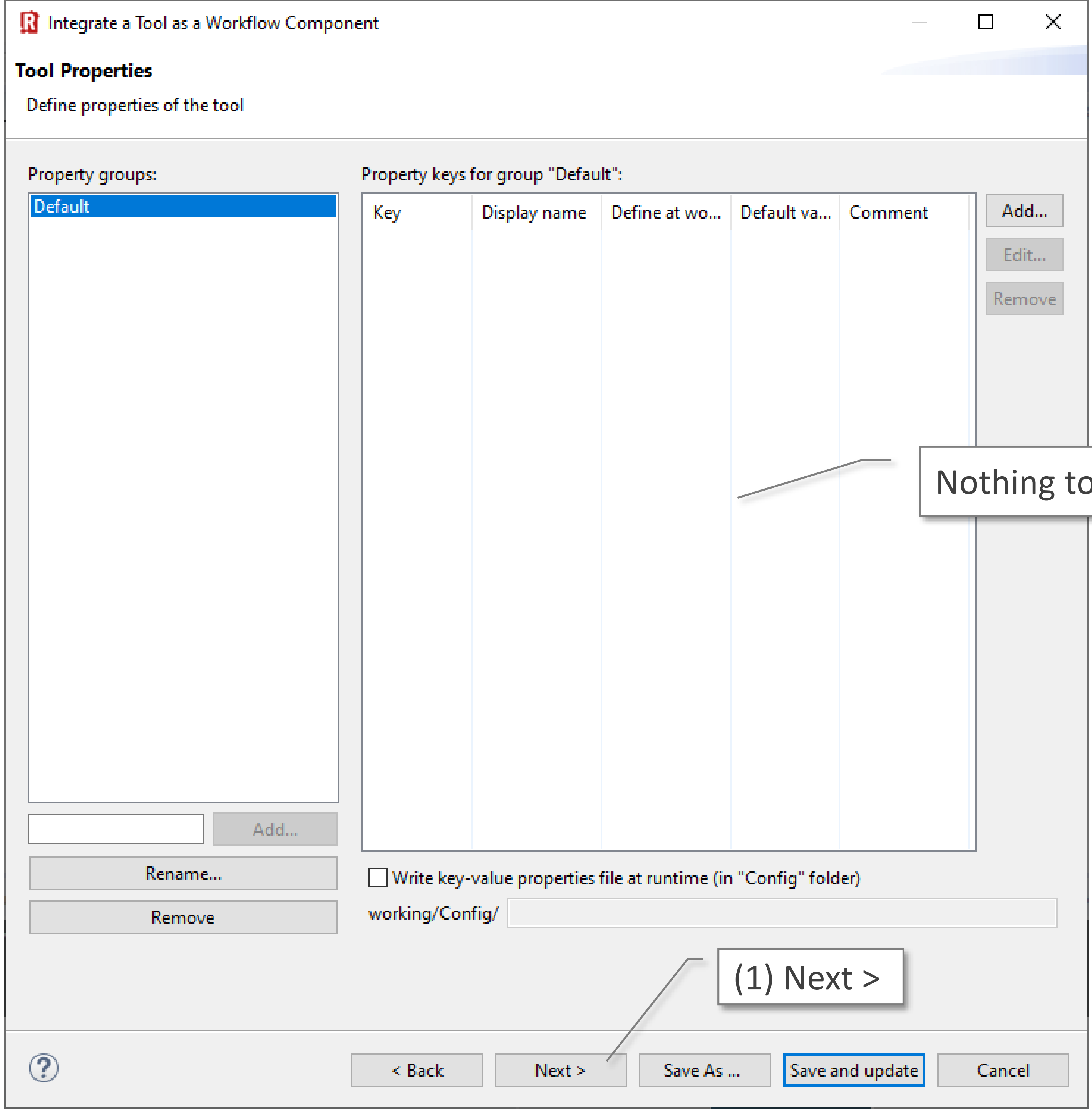
Name*: CPACS_out

Data type*: File

(1) Click “Add”

(2) Define RCE output variable of type “File”

(3) Next >


Integrate a Tool as a Workflow Component

Tool Properties

Define properties of the tool

Property groups:

Default

Property keys for group "Default":

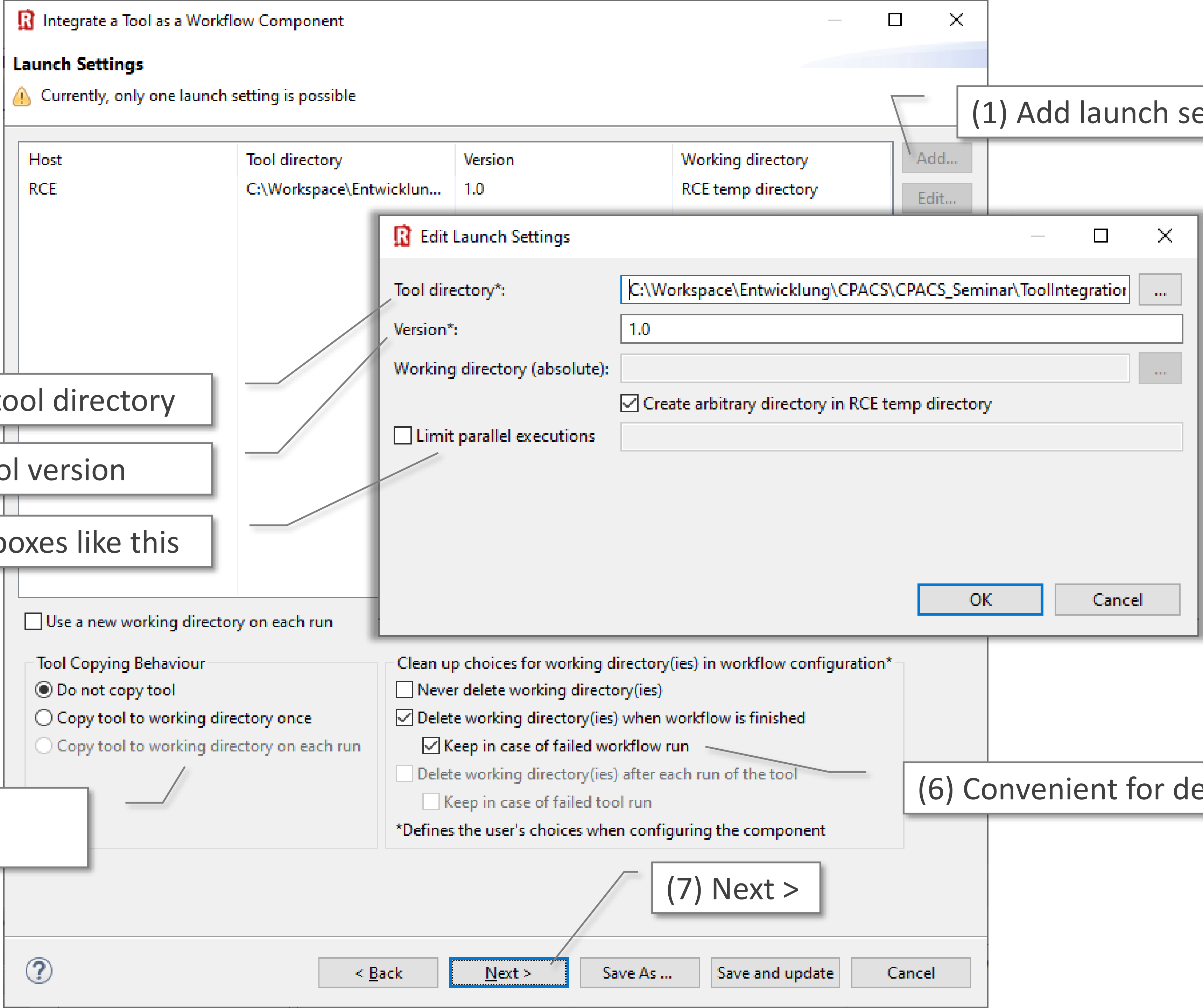
Key	Display name	Define at wo...	Default va...	Comment

☐ Write key-value properties file at runtime (in "Config" folder)

working/Config/

Nothing to do ...

(1) Next >


Integrate a Tool as a Workflow Component

Launch Settings

⚠ Currently, only one launch setting is possible

Host	Tool directory	Version	Working directory
RCE	C:\Workspace\Entwicklun...	1.0	RCE temp directory

Edit Launch Settings

Tool directory*: C:\Workspace\Entwicklung\CPACS\CPACS_Seminar\ToolIntegratior

Version*: 1.0

Working directory (absolute):

[✓] Create arbitrary directory in RCE temp directory

[ ] Limit parallel executions

[ ] Use a new working directory on each run

Tool Copying Behaviour

● Do not copy tool

○ Copy tool to working directory once

○ Copy tool to working directory on each run

Clean up choices for working directory(ies) in workflow configuration*

[ ] Never delete working directory(ies)

[✓] Delete working directory(ies) when workflow is finished

[✓] Keep in case of failed workflow run

[ ] Delete working directory(ies) after each run of the tool

[ ] Keep in case of failed tool run

*Defines the user's choices when configuring the component

(1) Add launch setting

(2) Point to tool directory

(3) Select tool version

(4) Select checkboxes like this

(5) Select your preferred copying behavior

(6) Convenient for debugging…

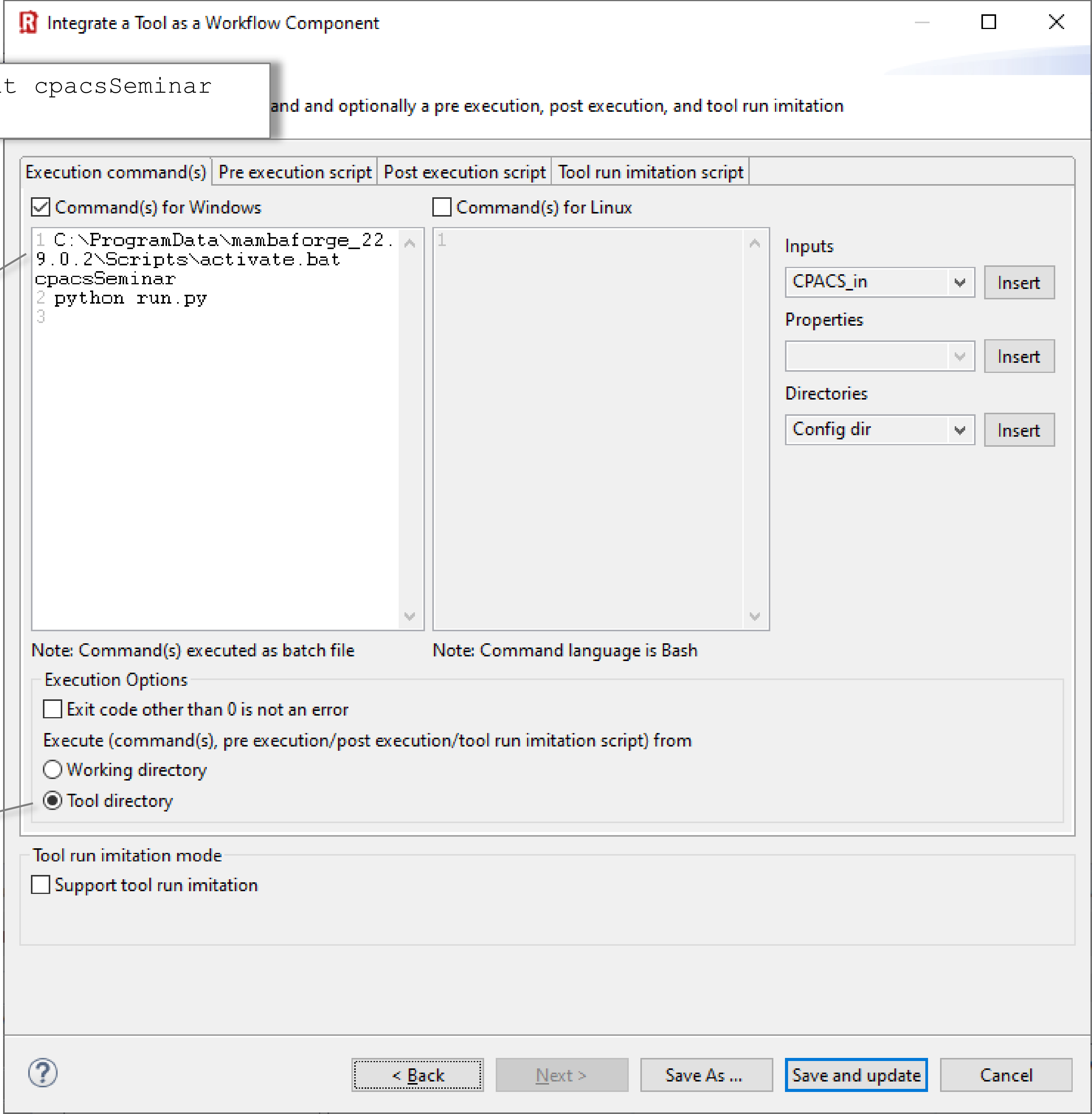
(7) Next >

```
C:\ProgramData\mambaforge_22.9.0.2\Scripts\activate.bat cpacsSeminar
python run.py
```

(1) Specify execution commands for Windows and/or Linux

ⓘ This example is written in Python, so we need to activate the correct interpreter first.

(2) Select "Tool directory"



Integrate a Tool as a Workflow Component

and and optionally a pre execution, post execution, and tool run imitation

[✓] Command(s) for Windows

```
C:\ProgramData\mambaforge_22.9.0.2\Scripts\activate.bat cpacsSeminar
python run.py
```

[ ] Command(s) for Linux

Inputs: CPACS_in

Properties:

Directories: Config dir

Note: Command(s) executed as batch file

Note: Command language is Bash

Execution Options

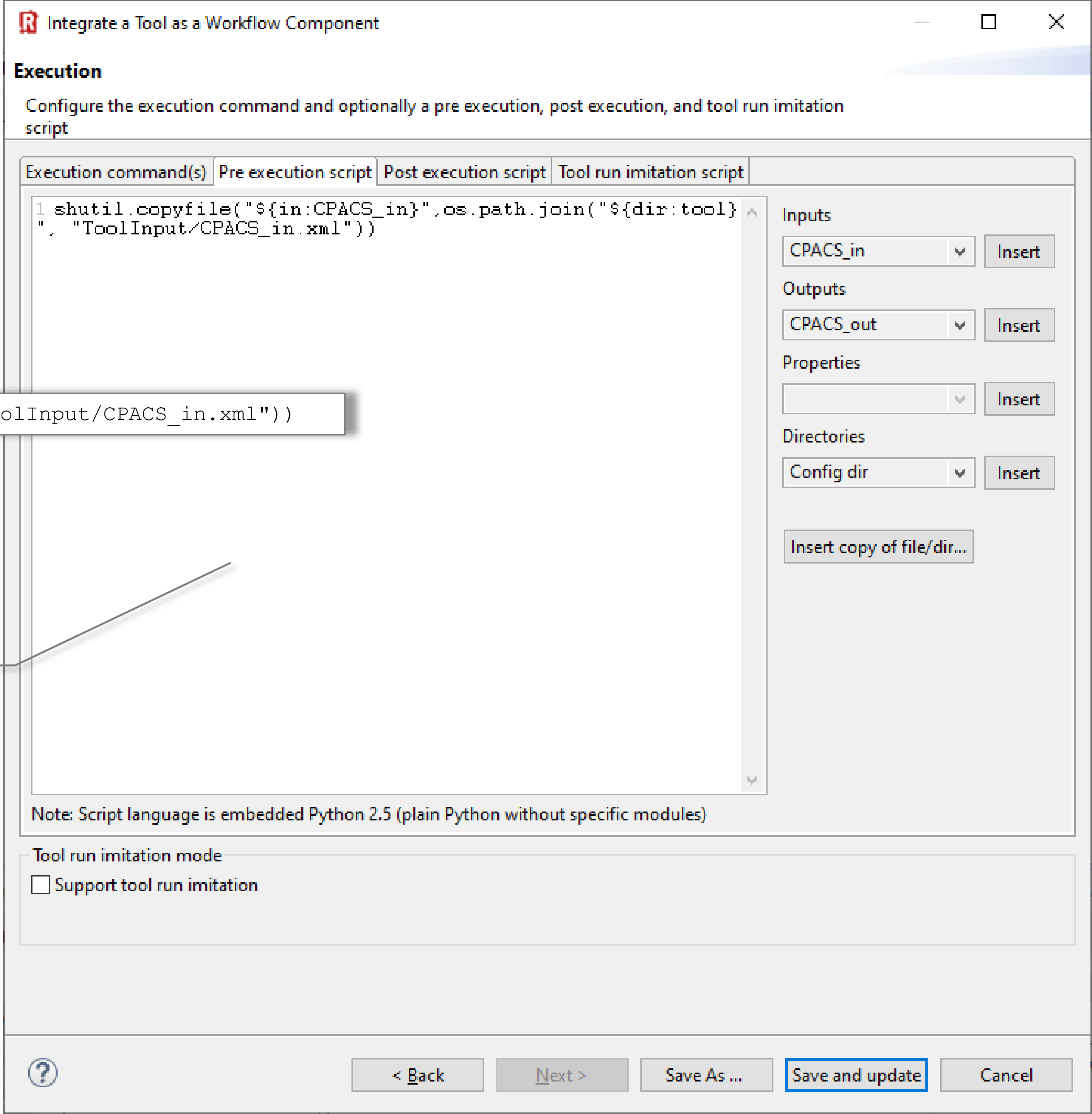
[ ] Exit code other than 0 is not an error

Execute (command(s), pre execution/post execution/tool run imitation script) from

Working directory [✓ Tool directory]

Tool run imitation mode

[ ] Support tool run imitation


Integrate a Tool as a Workflow Component

Execution

Configure the execution command and optionally a pre execution, post execution, and tool run imitation script

```
shutil.copyfile("${in:CPACS_in}",os.path.join("${dir:tool}", "ToolInput/CPACS_in.xml"))
```

Note: Script language is embedded Python 2.5 (plain Python without specific modules)

Tool run imitation mode

☐ Support tool run imitation

```
shutil.copyfile("${in:CPACS_in}",os.path.join("${dir:tool}", "ToolInput/CPACS_in.xml"))
```

(1) Insert pre-execution script in Python

ⓘ Pre-execution: What happens before the actual tool is activated. Use this to create the required folder structure, if necessary, and copy the input file into it.

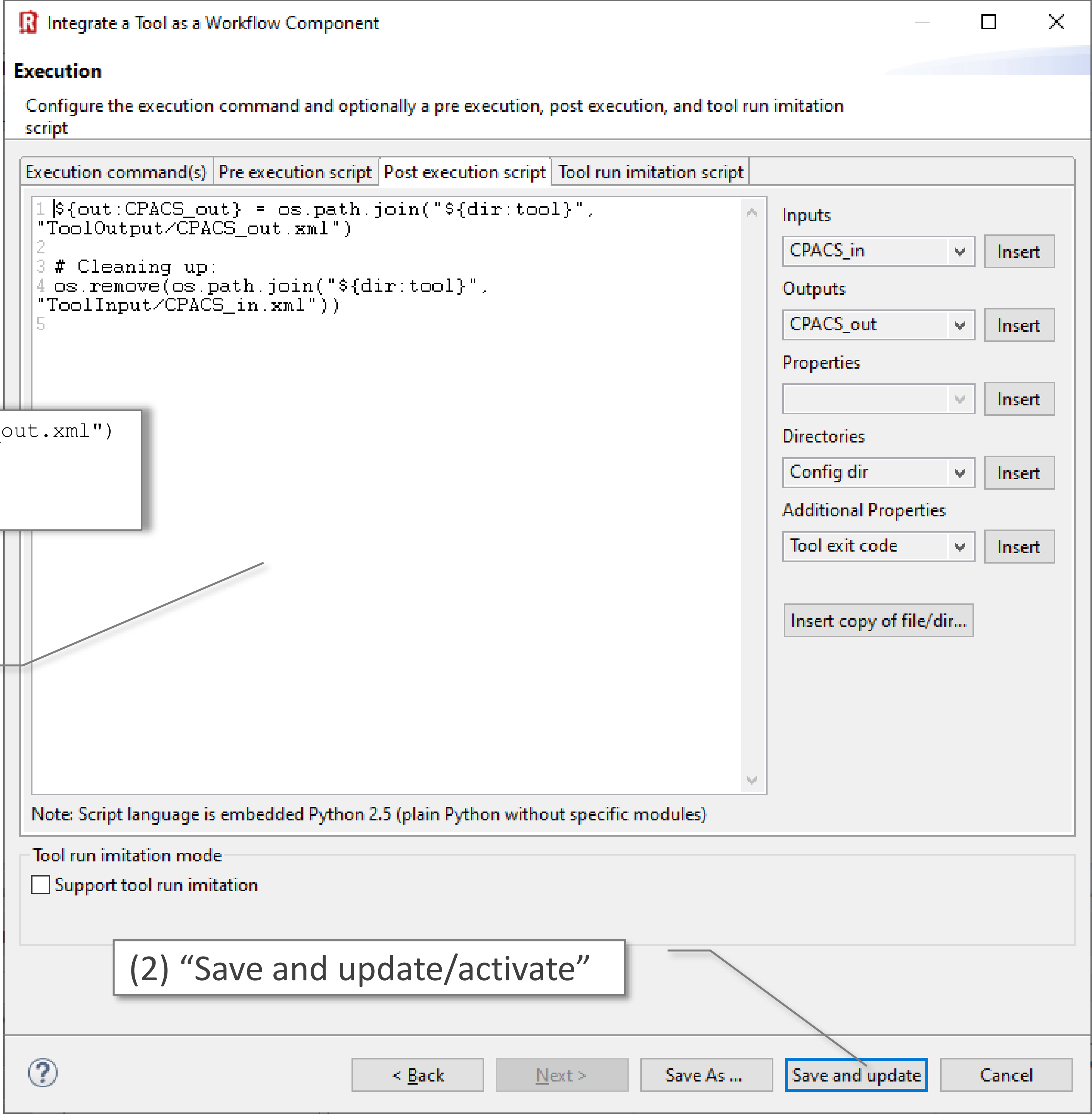
ⓘ Post-execution: What happens after the actual tool has finished. Here we're telling RCE where to find the output file. In this example the tool is not copied, it is run locally. So we clean up the input file to avoid any confusion when the workflow is run again.

```
${out:CPACS_out} = os.path.join("${dir:tool}", "ToolOutput/CPACS_out.xml")

# Cleaning up:
os.remove(os.path.join("${dir:tool}", "ToolInput/CPACS_in.xml"))
```

(1) Insert post-execution script in Python

(2) "Save and update/activate"



Integrate a Tool as a Workflow Component

Execution

Configure the execution command and optionally a pre execution, post execution, and tool run imitation script

Execution command(s) | Pre execution script | Post execution script | Tool run imitation script

```
${out:CPACS_out} = os.path.join("${dir:tool}", "ToolOutput/CPACS_out.xml")

# Cleaning up:
os.remove(os.path.join("${dir:tool}", "ToolInput/CPACS_in.xml"))
```

Inputs: CPACS_in — Insert

Outputs: CPACS_out — Insert

Properties: — Insert

Directories: Config dir — Insert

Additional Properties: Tool exit code — Insert

Insert copy of file/dir...

Note: Script language is embedded Python 2.5 (plain Python without specific modules)

Tool run imitation mode

☐ Support tool run imitation

< Back | Next > | Save As ... | Save and update | Cancel