Export Python Scripts for

# TEST AUTOMATION

## Objectives

At the end of this section, you will be able to

- Create a Python script for an existing set of test vectors
- Execute the Python script from the command line
- Configure the script to generate HTML and/or JUnit reports

testIDEA

SCRIPTING FOR

# TEST AUTOMATION

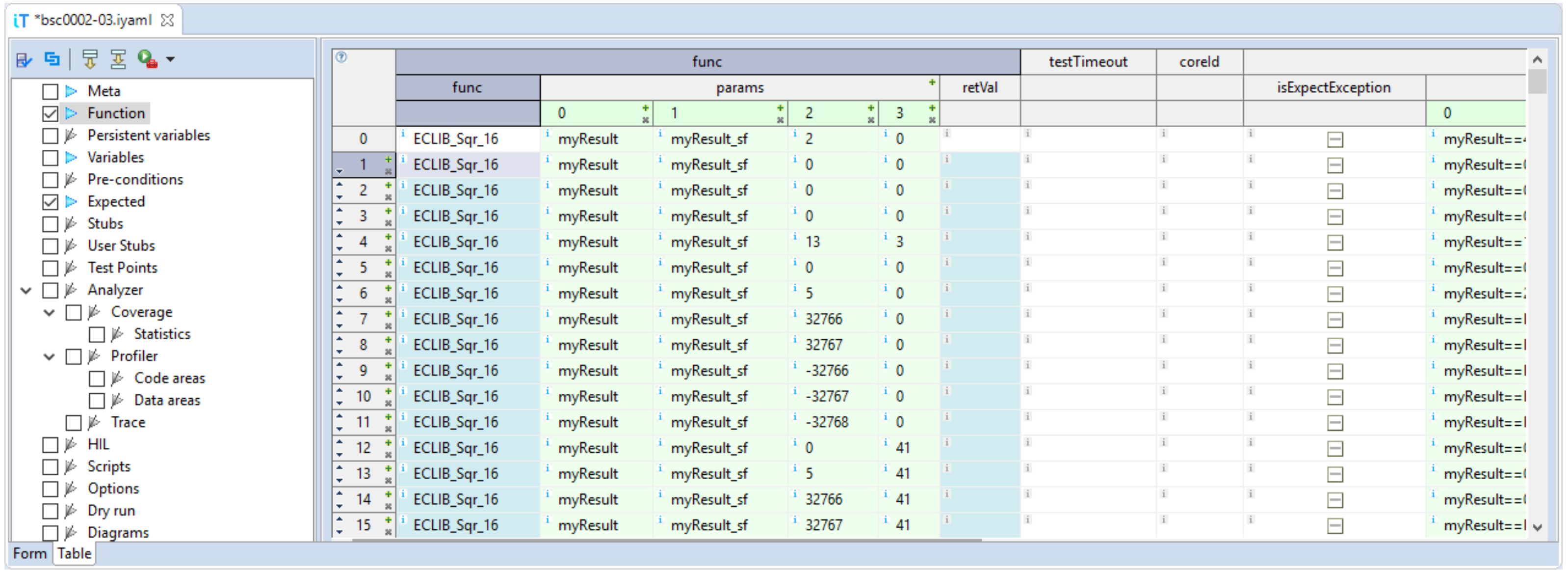
testIDEA

# 1 BACK TO THE EC-LIB© PROJECT

Continuous Integration (CI) tools, such as Jenkins and Bamboo, make it possible to check out code from a repository, build it and automatically test it, saving the results for later review.

testIDEA, with its Script Export functionality, makes it possible to export test cases in Python, which is easy to integrate into various automation tools or processes.

To demonstrate the export functionality of testIDEA we will reuse the EC-LIB© testing example as seen in Unit 05.



*bsc0002-03.iyaml

- ☐ Meta
- ☑ Function
- ☐ Persistent variables
- ☐ Variables
- ☐ Pre-conditions
- ☑ Expected
- ☐ Stubs
- ☐ User Stubs
- ☐ Test Points
- ☐ Analyzer
  - ☐ Coverage
    - ☐ Statistics
  - ☐ Profiler
    - ☐ Code areas
    - ☐ Data areas
  - ☐ Trace
- ☐ HIL
- ☐ Scripts
- ☐ Options
- ☐ Dry run
- ☐ Diagrams

| | func | | | | | | testTimeout | coreId | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | func | params | | | | retVal | | | isExpectException | |
| | | 0 | 1 | 2 | 3 | | | | | 0 |
| 0 | ECLIB_Sqr_16 | myResult | myResult_sf | 2 | 0 | | | | ⊟ | myResult== |
| 1 | ECLIB_Sqr_16 | myResult | myResult_sf | 0 | 0 | | | | ⊟ | myResult== |
| 2 | ECLIB_Sqr_16 | myResult | myResult_sf | 0 | 0 | | | | ⊟ | myResult== |
| 3 | ECLIB_Sqr_16 | myResult | myResult_sf | 0 | 0 | | | | ⊟ | myResult== |
| 4 | ECLIB_Sqr_16 | myResult | myResult_sf | 13 | 3 | | | | ⊟ | myResult== |
| 5 | ECLIB_Sqr_16 | myResult | myResult_sf | 0 | 0 | | | | ⊟ | myResult== |
| 6 | ECLIB_Sqr_16 | myResult | myResult_sf | 5 | 0 | | | | ⊟ | myResult== |
| 7 | ECLIB_Sqr_16 | myResult | myResult_sf | 32766 | 0 | | | | ⊟ | myResult== |
| 8 | ECLIB_Sqr_16 | myResult | myResult_sf | 32767 | 0 | | | | ⊟ | myResult== |
| 9 | ECLIB_Sqr_16 | myResult | myResult_sf | -32766 | 0 | | | | ⊟ | myResult== |
| 10 | ECLIB_Sqr_16 | myResult | myResult_sf | -32767 | 0 | | | | ⊟ | myResult== |
| 11 | ECLIB_Sqr_16 | myResult | myResult_sf | -32768 | 0 | | | | ⊟ | myResult== |
| 12 | ECLIB_Sqr_16 | myResult | myResult_sf | 0 | 41 | | | | ⊟ | myResult== |
| 13 | ECLIB_Sqr_16 | myResult | myResult_sf | 5 | 41 | | | | ⊟ | myResult== |
| 14 | ECLIB_Sqr_16 | myResult | myResult_sf | 32766 | 41 | | | | ⊟ | myResult== |
| 15 | ECLIB_Sqr_16 | myResult | myResult_sf | 32767 | 41 | | | | ⊟ | myResult== |

*The import functionality of testIDEA shown in this unit requires the testIDEA Pro license.*

PRO

# 2 THREE STEPS OF AUTOMATION

In this unit we will go through three steps to create a scripted unit test.

Firstly a Python script will be generated from the testIDEA GUI. This will be created using an existing set of test cases.

Next, the script will be executed from the command line.

Finally, the script will be updated to automatically generate a test report in either HTML format or a format that can be used by CI tools such as Jenkins.

1. Generate a Python test script

   Go to “Generate a Python test script”

2. Call a Python test script

   Go to “Call a Python test script for test automation”

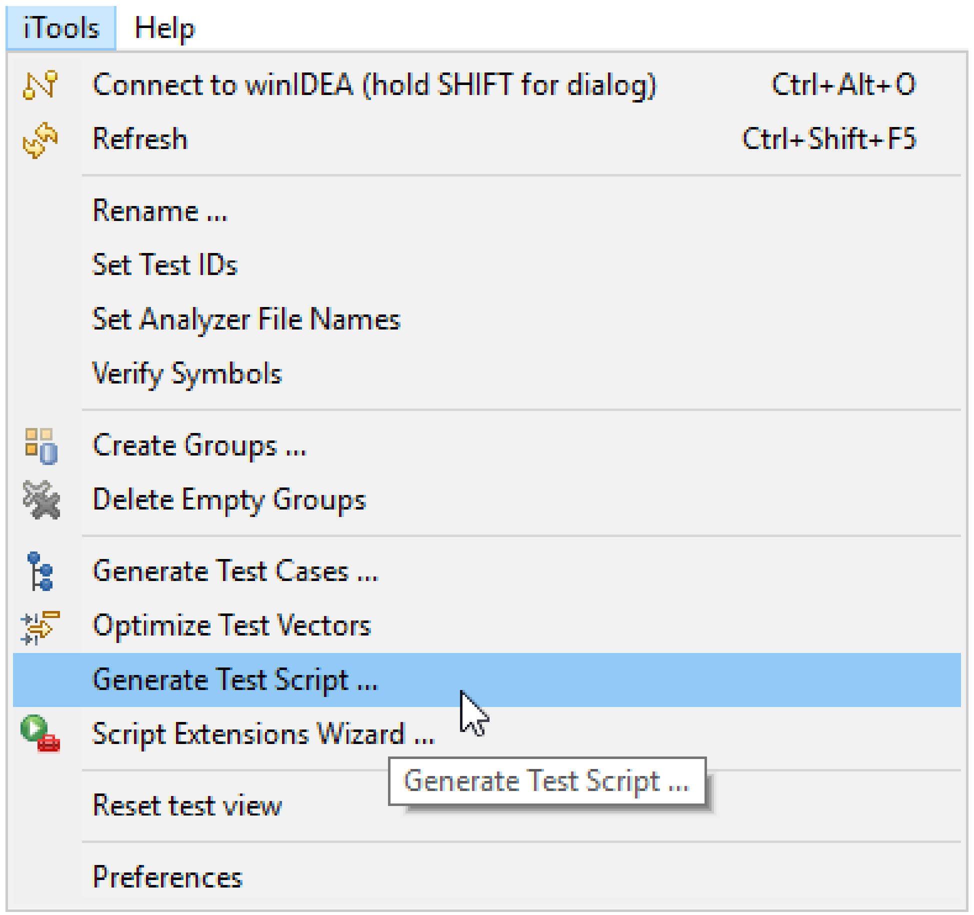
3. Automate test report creation

   Go to “Automate test report creation“

## 3 STEP 1 - GENERATE A PYTHON TEST SCRIPT

Start by creating a set of test cases or, as here, opening an existing iYAML file. Here we are using the imported tests vectors generated for the EC-LIB© library, the version without the generation of code coverage, which will save us some time during test execution.

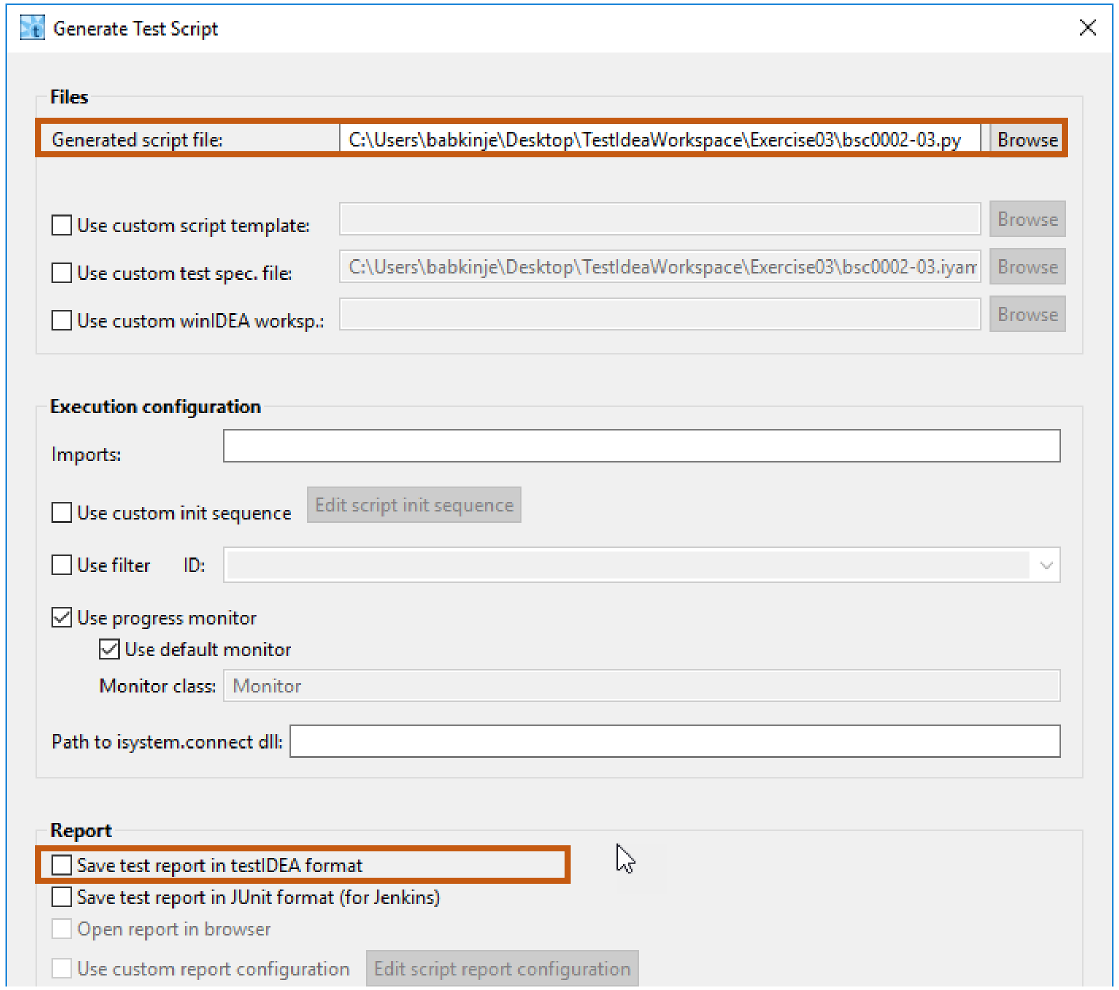
To generate a Python test script, start by opening the *iTools* menu and select the *Generate Test Script…* option.



# 3 STEP 1 - GENERATE A PYTHON TEST SCRIPT

Start by naming your Python script file and, if required, modifying the path where it is to be saved.

Initially we will not use the reporting feature, so we can uncheck the box *Save test report in testIDEA format* in the section *Report*.



Generate Test Script

**Files**

Generated script file: C:\Users\babkinje\Desktop\TestIdeaWorkspace\Exercise03\bsc0002-03.py Browse

☐ Use custom script template: Browse

☐ Use custom test spec. file: C:\Users\babkinje\Desktop\TestIdeaWorkspace\Exercise03\bsc0002-03.iyam Browse

☐ Use custom winIDEA worksp.: Browse

**Execution configuration**

Imports:

☐ Use custom init sequence — Edit script init sequence

☐ Use filter ID:

☑ Use progress monitor

☑ Use default monitor

Monitor class: Monitor

Path to isystem.connect dll:

**Report**

☐ Save test report in testIDEA format

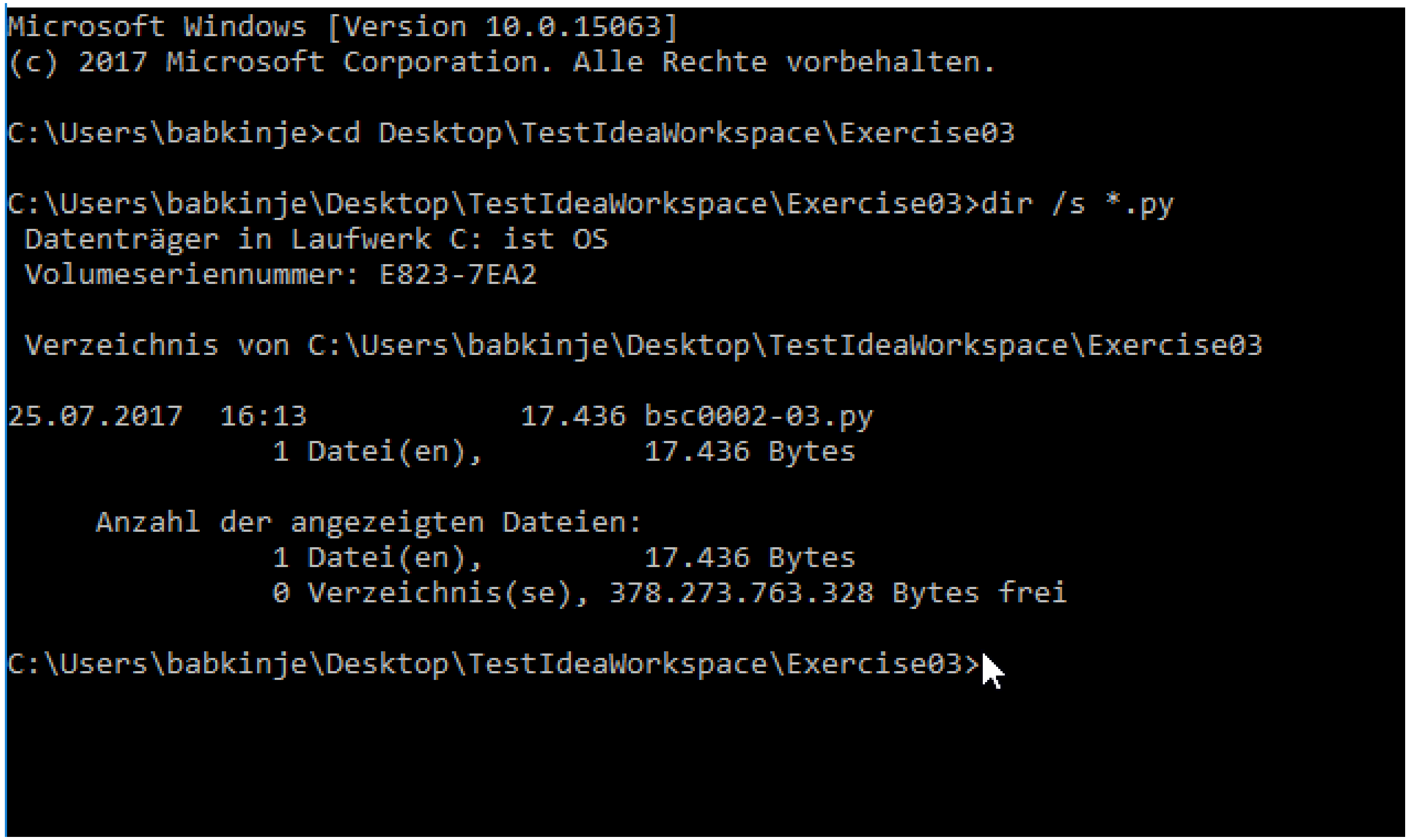
☐ Save test report in JUnit format (for Jenkins)

☐ Open report in browser

☐ Use custom report configuration — Edit script report configuration

# 4 STEP 2 - CALL PYTHON SCRIPT FOR TEST AUTOMATION

Next, open a command line shell and go to the directory where we saved our script.



```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\babkinje>cd Desktop\TestIdeaWorkspace\Exercise03

C:\Users\babkinje\Desktop\TestIdeaWorkspace\Exercise03>dir /s *.py
 Datenträger in Laufwerk C: ist OS
 Volumeseriennummer: E823-7EA2

 Verzeichnis von C:\Users\babkinje\Desktop\TestIdeaWorkspace\Exercise03

25.07.2017  16:13            17.436 bsc0002-03.py
               1 Datei(en),         17.436 Bytes

     Anzahl der angezeigten Dateien:
               1 Datei(en),         17.436 Bytes
               0 Verzeichnis(se), 378.273.763.328 Bytes frei

C:\Users\babkinje\Desktop\TestIdeaWorkspace\Exercise03>
```

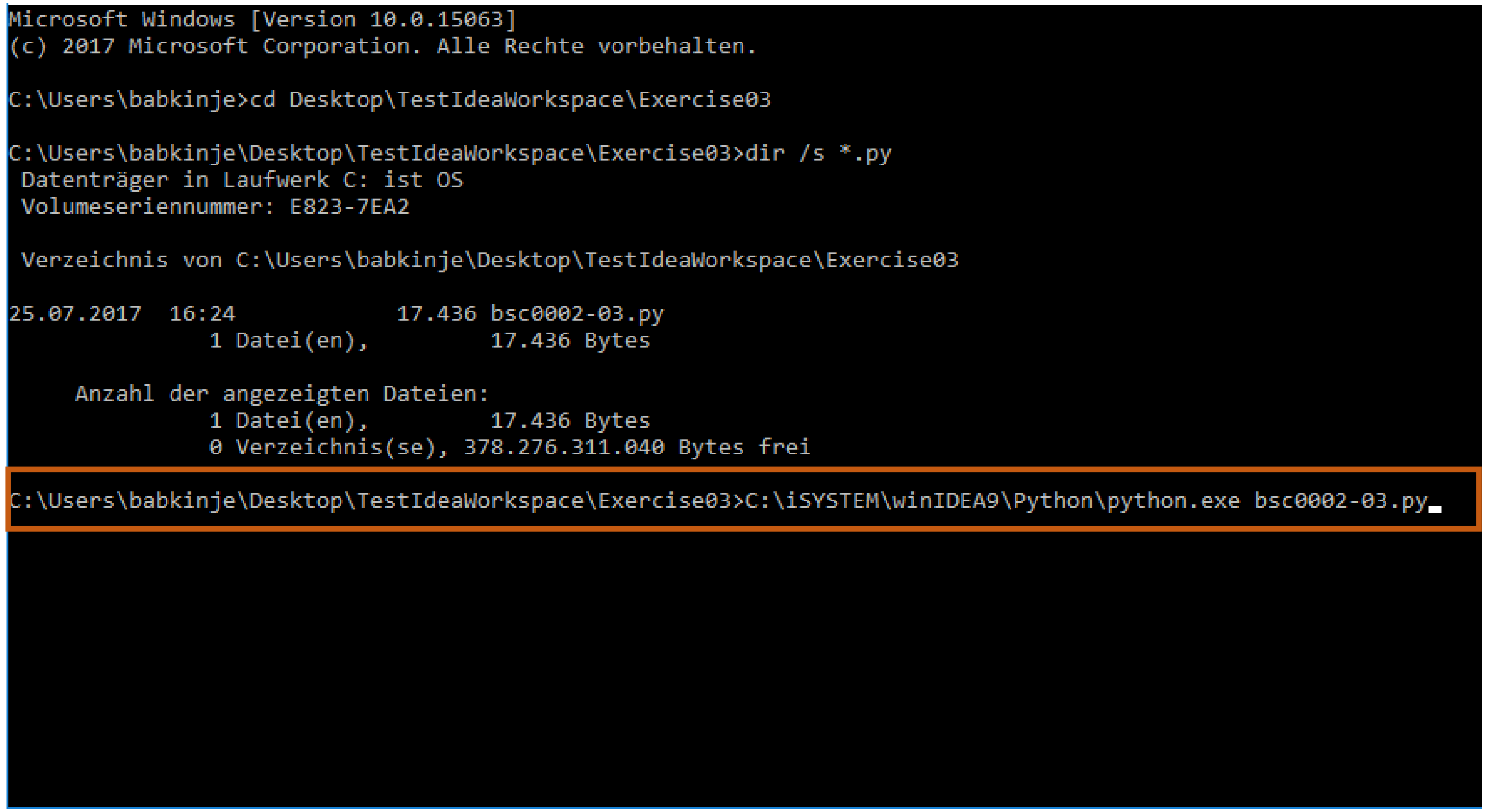
## 4 STEP 2 - CALL PYTHON SCRIPT FOR TEST AUTOMATION

winIDEA installs its own version of Python with the necessary isystem.connect libraries to support scripted use of iSYSTEM's BlueBox™ On-Chip Analyzers. For beginners we recommend to use this version of Python.

In this example, Python is called from the installation path of winIDEA as follows:

```
C:\iSYSTEM\winIDEA9\
             Python\python.exe
```

The names of the Python script, *bsc0002-03.py*, is passed as a parameter.



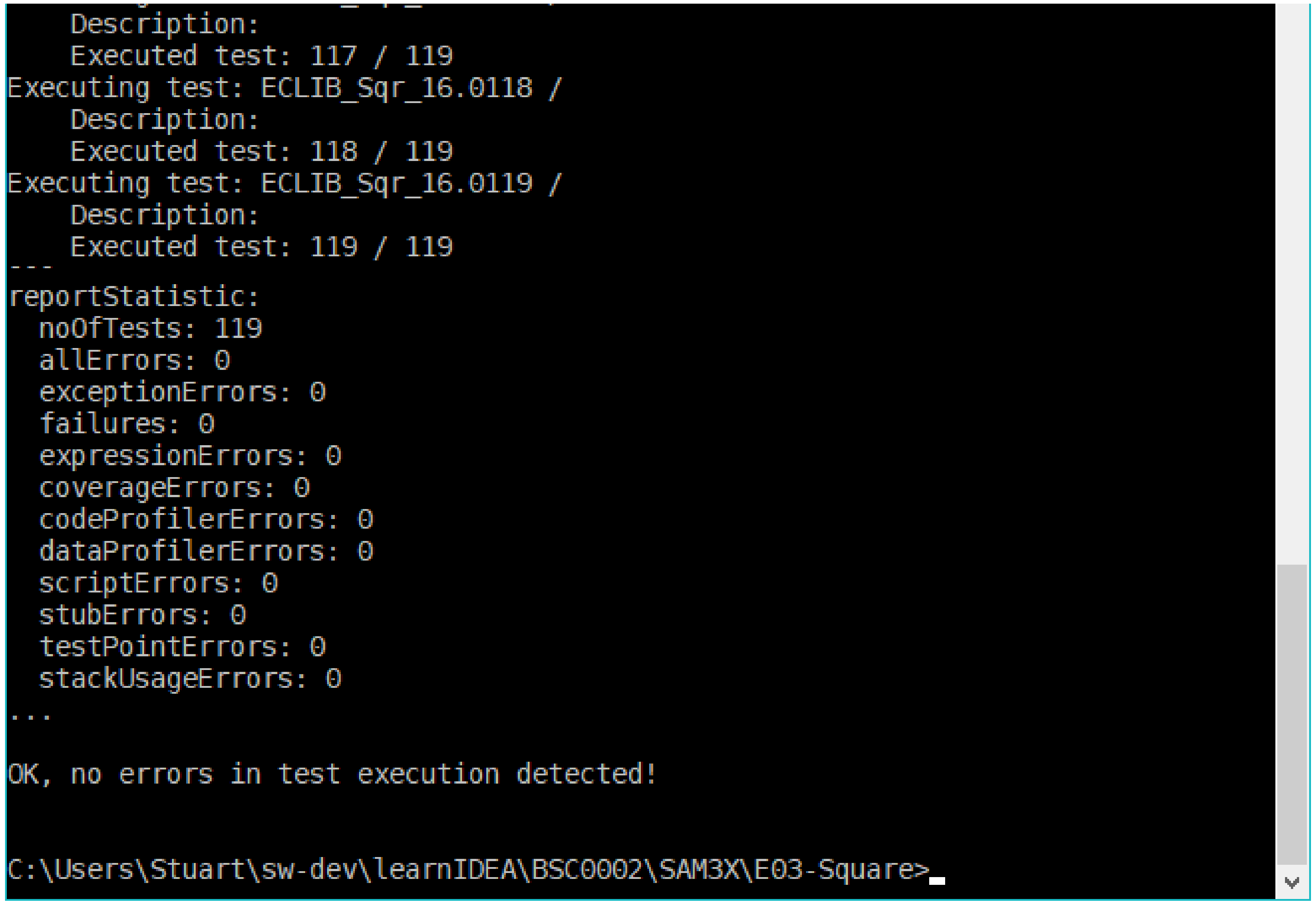
```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\babkinje>cd Desktop\TestIdeaWorkspace\Exercise03

C:\Users\babkinje\Desktop\TestIdeaWorkspace\Exercise03>dir /s *.py
 Datenträger in Laufwerk C: ist OS
 Volumeseriennummer: E823-7EA2

 Verzeichnis von C:\Users\babkinje\Desktop\TestIdeaWorkspace\Exercise03

25.07.2017  16:24            17.436 bsc0002-03.py
               1 Datei(en),         17.436 Bytes

     Anzahl der angezeigten Dateien:
               1 Datei(en),         17.436 Bytes
               0 Verzeichnis(se), 378.276.311.040 Bytes frei

C:\Users\babkinje\Desktop\TestIdeaWorkspace\Exercise03>C:\iSYSTEM\winIDEA9\Python\python.exe bsc0002-03.py_
```

# 4 STEP 2 - CALL PYTHON SCRIPT FOR TEST AUTOMATION

Upon execution, the same process is performed as you would expect from executing the tests in the testIDEA GUI.

The test results are shown at the end of the command line output, as shown opposite.

*This Python script can be executed directly from the CI tool Jenkins in the same manner as shown here. Later slides will show how to generate reports suitable for display in that platform.*



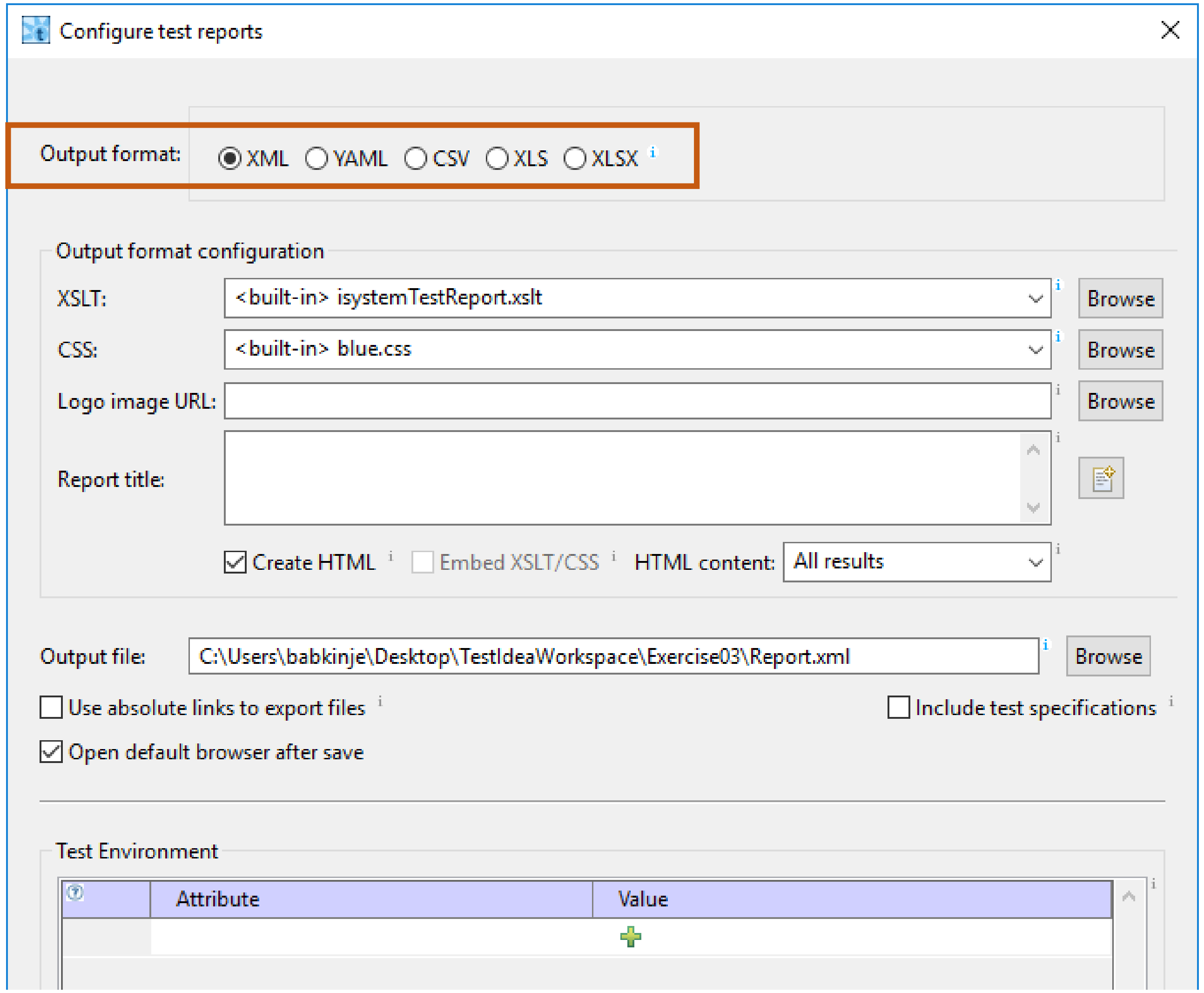
```
    Description:
    Executed test: 117 / 119
Executing test: ECLIB_Sqr_16.0118 /
    Description:
    Executed test: 118 / 119
Executing test: ECLIB_Sqr_16.0119 /
    Description:
    Executed test: 119 / 119
---
reportStatistic:
  noOfTests: 119
  allErrors: 0
  exceptionErrors: 0
  failures: 0
  expressionErrors: 0
  coverageErrors: 0
  codeProfilerErrors: 0
  dataProfilerErrors: 0
  scriptErrors: 0
  stubErrors: 0
  testPointErrors: 0
  stackUsageErrors: 0
...

OK, no errors in test execution detected!


C:\Users\Stuart\sw-dev\learnIDEA\BSC0002\SAM3X\E03-Square>_
```

# 5 STEP 3 - CONFIGURE TEST REPORT CREATION

In order to create test reports, the report format must be configured

Start by selecting *Test* menu and clicking *Configure test report* to open the dialog seen here.

1 Define Output format


1

Configure test reports

Output format: [● XML] ○ YAML ○ CSV ○ XLS ○ XLSX

Output format configuration

XSLT: <built-in> isystemTestReport.xslt

CSS: <built-in> blue.css

Logo image URL:

Report title:

☑ Create HTML ☐ Embed XSLT/CSS HTML content: All results

Output file: C:\Users\babkinje\Desktop\TestIdeaWorkspace\Exercise03\Report.xml

☐ Use absolute links to export files ☐ Include test specifications

☑ Open default browser after save

Test Environment

| Attribute | Value |
|---|---|
| | |

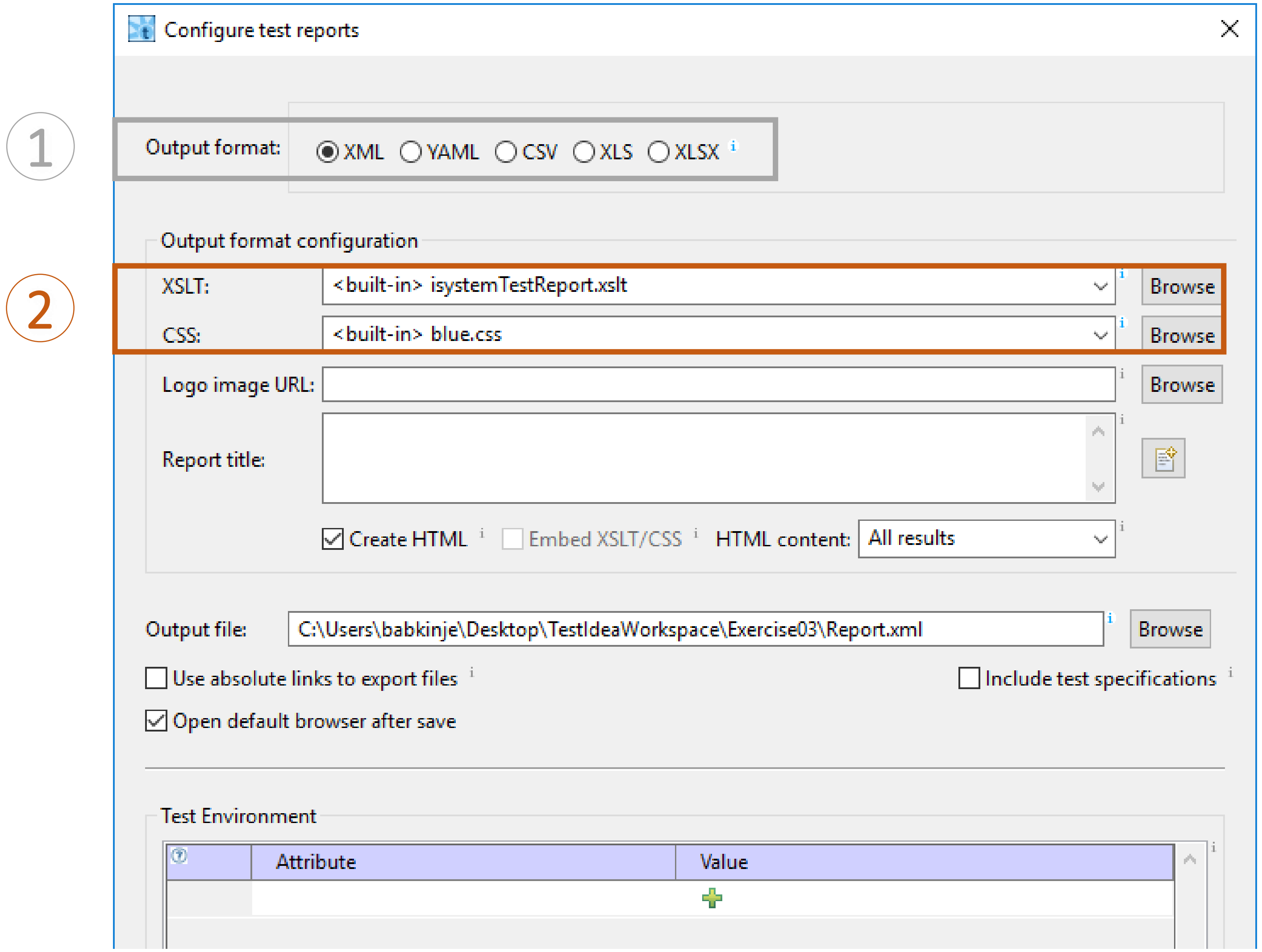
# 5 STEP 3 - CONFIGURE TEST REPORT CREATION

In order to create test reports, the report format must be configured

Start by selecting *Test* menu and clicking *Configure test report* to open the dialog seen here.

1 Define Output format

2 Define a XSLT and a CSS template, otherwise the report can't be displayed properly



Output format: [✓ XML] YAML CSV XLS XLSX

Output format configuration

XSLT: <built-in> isystemTestReport.xslt

CSS: <built-in> blue.css

Logo image URL:

Report title:

[✓] Create HTML   [ ] Embed XSLT/CSS   HTML content: All results

Output file: C:\Users\babkinje\Desktop\TestIdeaWorkspace\Exercise03\Report.xml

[ ] Use absolute links to export files   [ ] Include test specifications

[✓] Open default browser after save

Test Environment

# 5 STEP 3 - CONFIGURE TEST REPORT CREATION

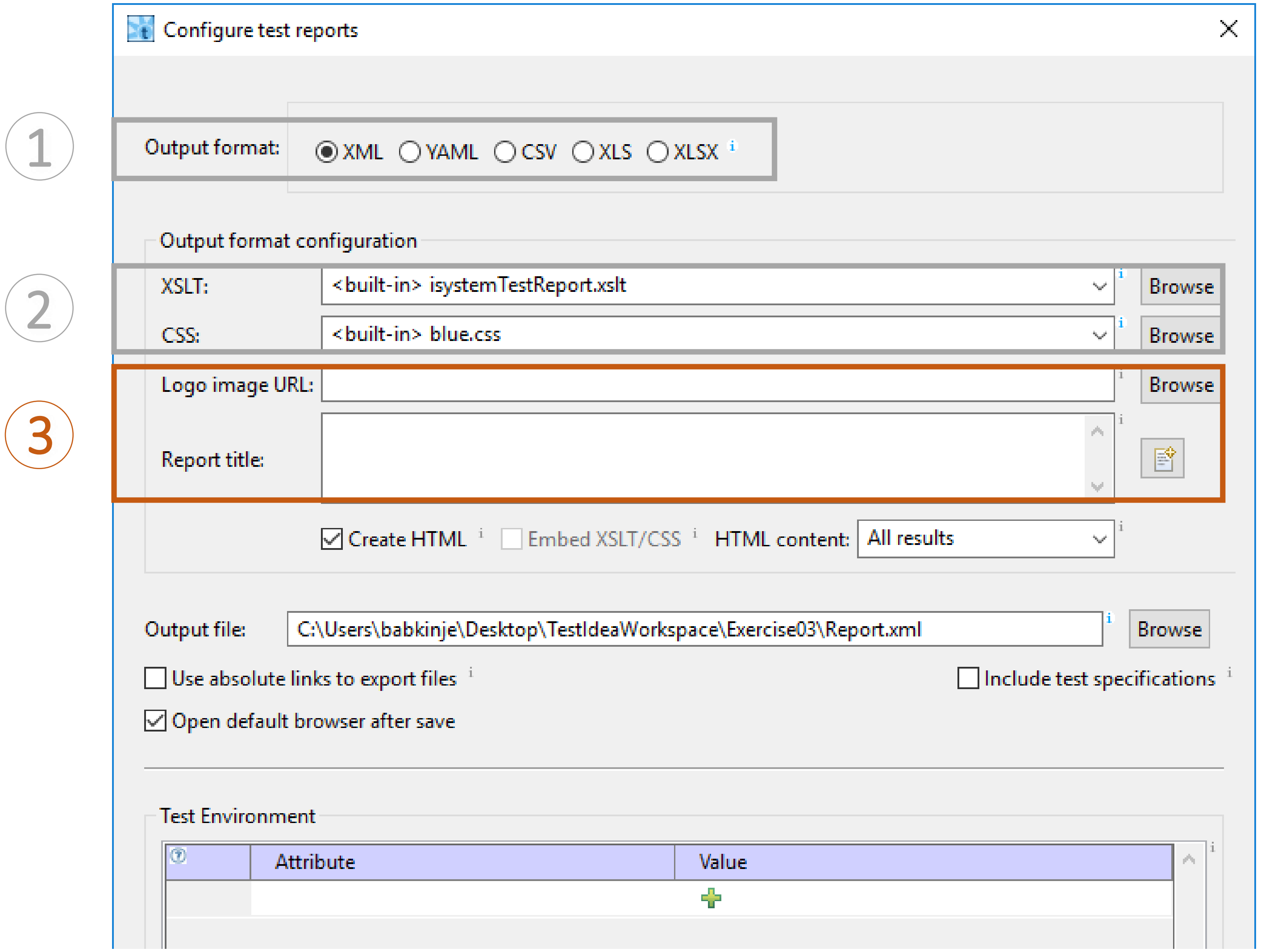
In order to create test reports, the report format must be configured

Start by selecting *Test* menu and clicking *Configure test report* to open the dialog seen here.

1 Define Output format

2 Define a XSLT and a CSS template, otherwise the report can't be displayed properly

3 Logo can be pasted in the report as well as a report title



Configure test reports

Output format: [✓ XML] YAML CSV XLS XLSX

Output format configuration

XSLT: <built-in> isystemTestReport.xslt Browse

CSS: <built-in> blue.css Browse

Logo image URL: Browse

Report title:

[✓] Create HTML  ☐ Embed XSLT/CSS  HTML content: All results

Output file: C:\Users\babkinje\Desktop\TestIdeaWorkspace\Exercise03\Report.xml Browse

☐ Use absolute links to export files  ☐ Include test specifications

[✓] Open default browser after save

Test Environment

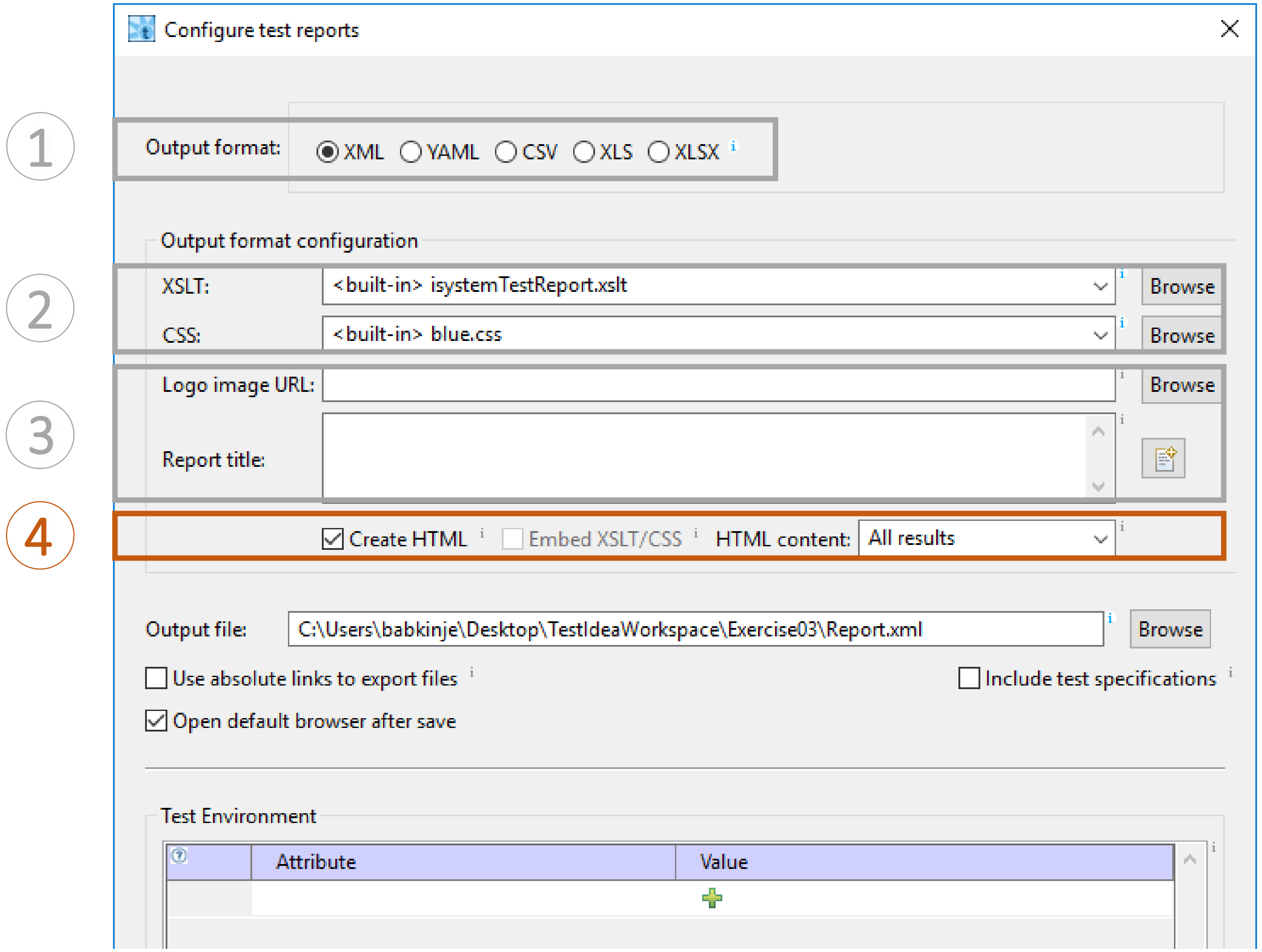
| Attribute | Value |
|---|---|
| | |

# 5 STEP 3 - CONFIGURE TEST REPORT CREATION

In order to create test reports, the report format must be configured

Start by selecting *Test* menu and clicking *Configure test report* to open the dialog seen here.

1. Define Output format
2. Define a XSLT and a CSS template, otherwise the report can't be displayed properly
3. Logo can be pasted in the report as well as a report title
4. *"Create HTML"*



Output format: [✓ XML] YAML CSV XLS XLSX

Output format configuration

XSLT: <built-in> isystemTestReport.xslt

CSS: <built-in> blue.css

Logo image URL:

Report title:

[✓ Create HTML] Embed XSLT/CSS HTML content: All results

Output file: C:\Users\babkinje\Desktop\TestIdeaWorkspace\Exercise03\Report.xml

☐ Use absolute links to export files

☐ Include test specifications

☑ Open default browser after save

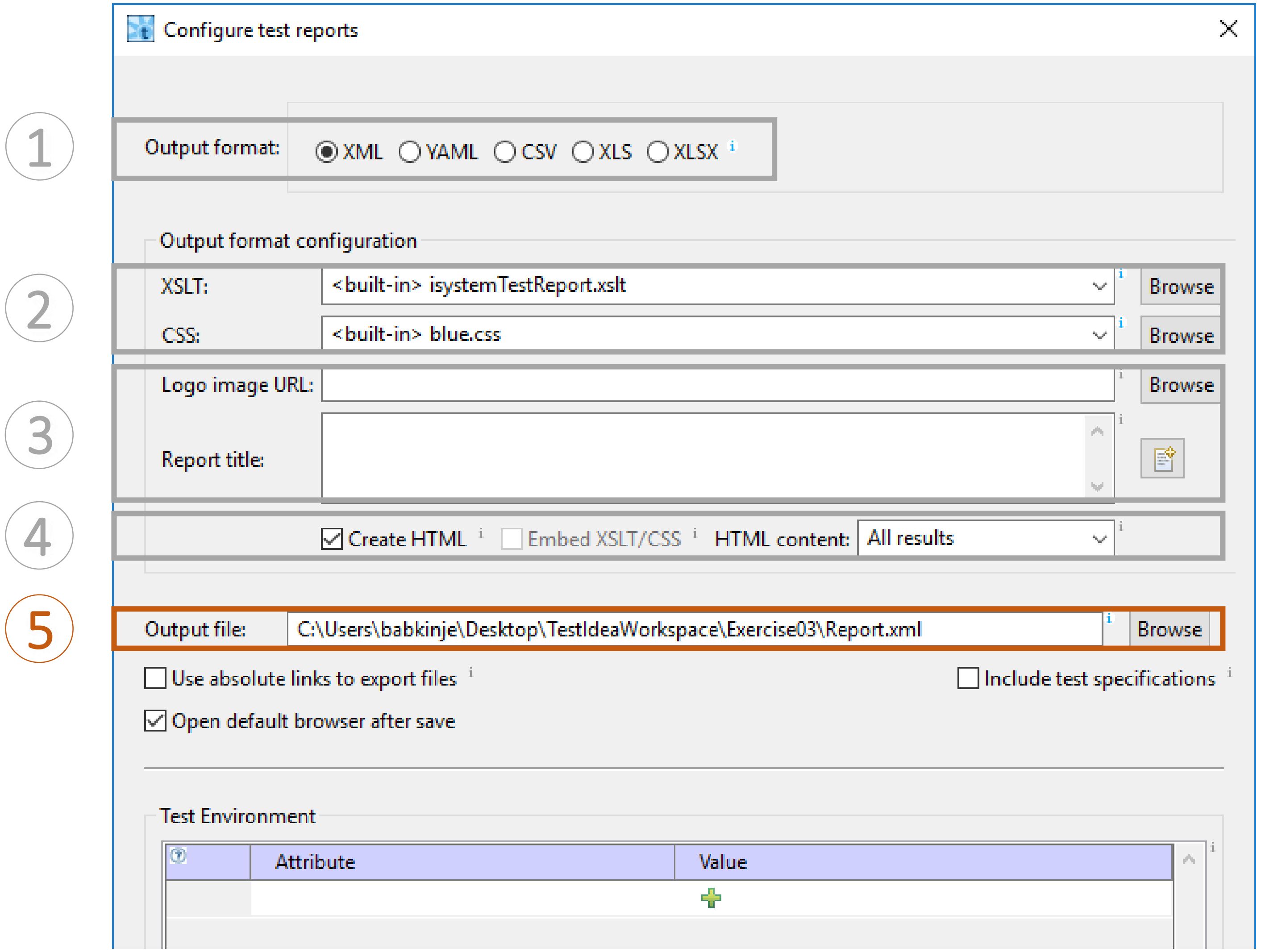
Test Environment

# 5 STEP 3 - CONFIGURE TEST REPORT CREATION

In order to create test reports, the report format must be configured

Start by selecting *Test* menu and clicking *Configure test report* to open the dialog seen here.

1. Define Output format
2. Define a XSLT and a CSS template, otherwise the report can't be displayed properly
3. Logo can be pasted in the report as well as a report title
4. *"Create HTML"*
5. Output file: Where the report should be saved



Output format: [● XML] ○ YAML ○ CSV ○ XLS ○ XLSX

Output format configuration

XSLT: <built-in> isystemTestReport.xslt

CSS: <built-in> blue.css

Logo image URL:

Report title:

☑ Create HTML ☐ Embed XSLT/CSS HTML content: All results

Output file: C:\Users\babkinje\Desktop\TestIdeaWorkspace\Exercise03\Report.xml

☐ Use absolute links to export files ☐ Include test specifications

☑ Open default browser after save

Test Environment

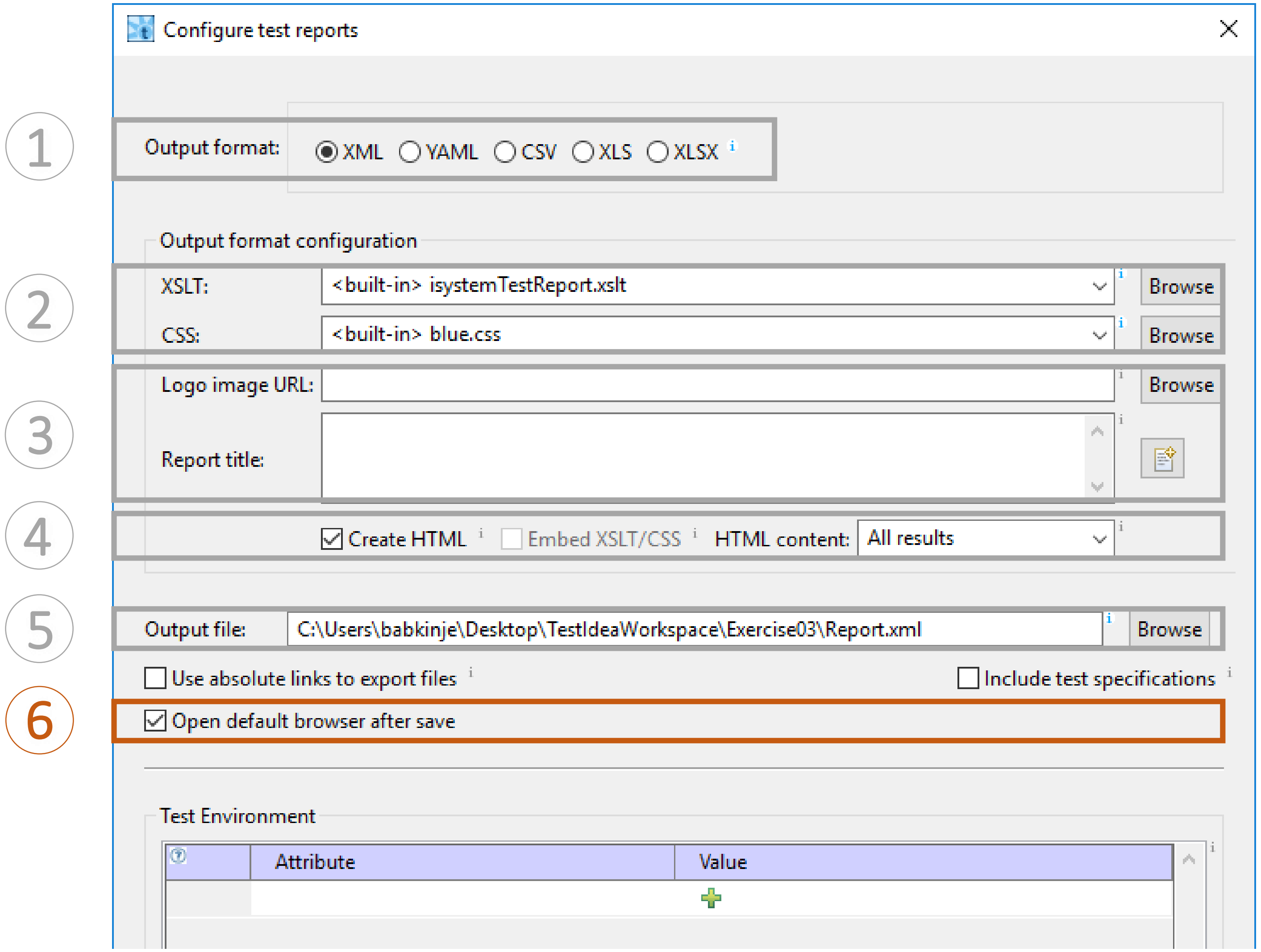
| Attribute | Value |
|---|---|
| | |

# 5 STEP 3 - CONFIGURE TEST REPORT CREATION

In order to create test reports, the report format must be configured

Start by selecting *Test* menu and clicking *Configure test report* to open the dialog seen here.

6 Set "*Open default browser after save*" check-box. This ensures that the report file will be opened in the browser for viewing upon test completion.



Configure test reports

Output format: [✓ XML] YAML CSV XLS XLSX

Output format configuration

XSLT: <built-in> isystemTestReport.xslt Browse

CSS: <built-in> blue.css Browse

Logo image URL: Browse

Report title:

[✓] Create HTML ☐ Embed XSLT/CSS HTML content: All results

Output file: C:\Users\babkinje\Desktop\TestIdeaWorkspace\Exercise03\Report.xml Browse

☐ Use absolute links to export files ☐ Include test specifications

[✓] Open default browser after save

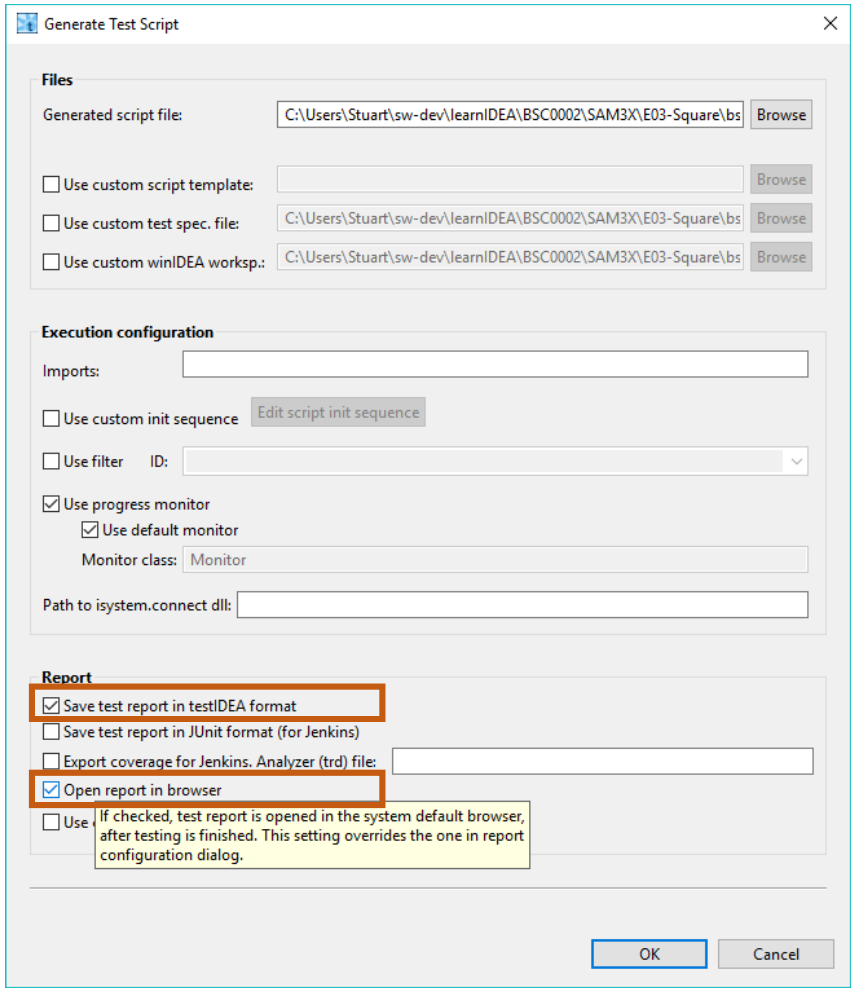
Test Environment

| Attribute | Value |
|---|---|
| | |

# 5 STEP 3 - CONFIGURE TEST REPORT CREATION

Now we can create a new Python script as before (*iTools -> Generate Test Script…* ) and execute from the command line in the same manner as before.

*If you plan to use the Python script together with a Continuous Integration tool such as Jenkins, it is recommended to leave the "Open report in Browser" option unchecked.*


Generate Test Script

**Files**

Generated script file: C:\Users\Stuart\sw-dev\learnIDEA\BSC0002\SAM3X\E03-Square\bs [Browse]

☐ Use custom script template: [Browse]

☐ Use custom test spec. file: C:\Users\Stuart\sw-dev\learnIDEA\BSC0002\SAM3X\E03-Square\bs [Browse]

☐ Use custom winIDEA worksp.: C:\Users\Stuart\sw-dev\learnIDEA\BSC0002\SAM3X\E03-Square\bs [Browse]

**Execution configuration**

Imports:

☐ Use custom init sequence [Edit script init sequence]

☐ Use filter ID:

☑ Use progress monitor

☑ Use default monitor

Monitor class: Monitor

Path to isystem.connect dll:

**Report**

☑ Save test report in testIDEA format

☐ Save test report in JUnit format (for Jenkins)

☐ Export coverage for Jenkins. Analyzer (trd) file:

☑ Open report in browser

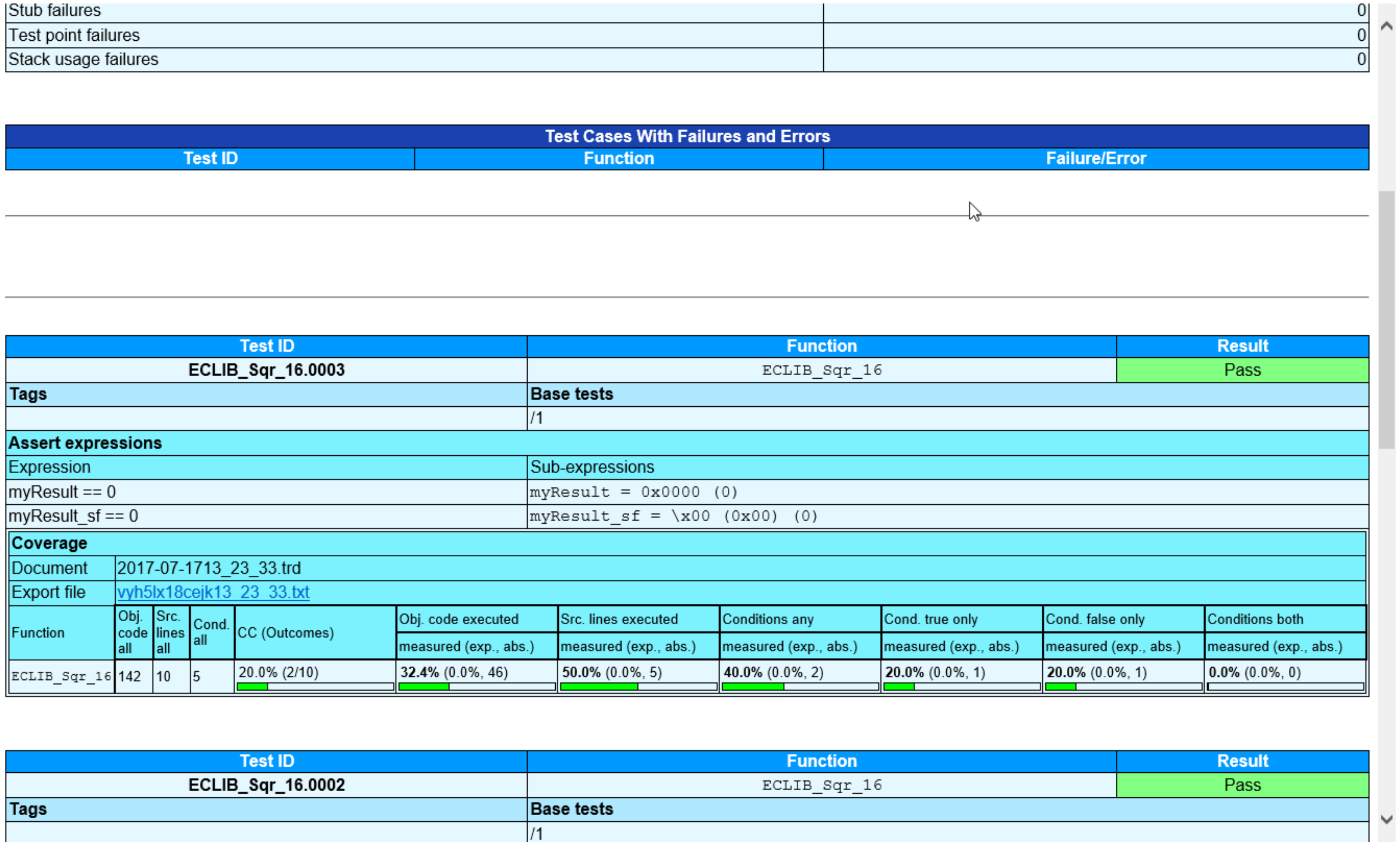
☐ Use

If checked, test report is opened in the system default browser, after testing is finished. This setting overrides the one in report configuration dialog.

[OK] [Cancel]

# 5 STEP 3 - CONFIGURE TEST REPORT CREATION

Once the tests are finished the test report will be automatically generated and displayed in the default browser.



| | |
|---|---|
| Stub failures | 0 |
| Test point failures | 0 |
| Stack usage failures | 0 |

**Test Cases With Failures and Errors**

| Test ID | Function | Failure/Error |
|---|---|---|

| Test ID | Function | Result |
|---|---|---|
| **ECLIB_Sqr_16.0003** | `ECLIB_Sqr_16` | Pass |
| **Tags** | **Base tests** | |
| | /1 | |

**Assert expressions**

| Expression | Sub-expressions |
|---|---|
| myResult == 0 | `myResult = 0x0000 (0)` |
| myResult_sf == 0 | `myResult_sf = \x00 (0x00) (0)` |

**Coverage**

| | |
|---|---|
| Document | 2017-07-1713_23_33.trd |
| Export file | vyh5lx18cejk13_23_33.txt |

| Function | Obj. code all | Src. lines all | Cond. all | CC (Outcomes) | Obj. code executed | Src. lines executed | Conditions any | Cond. true only | Cond. false only | Conditions both |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | measured (exp., abs.) | measured (exp., abs.) | measured (exp., abs.) | measured (exp., abs.) | measured (exp., abs.) | measured (exp., abs.) |
| `ECLIB_Sqr_16` | 142 | 10 | 5 | 20.0% (2/10) | **32.4%** (0.0%, 46) | **50.0%** (0.0%, 5) | **40.0%** (0.0%, 2) | **20.0%** (0.0%, 1) | **20.0%** (0.0%, 1) | **0.0%** (0.0%, 0) |

| Test ID | Function | Result |
|---|---|---|
| **ECLIB_Sqr_16.0002** | `ECLIB_Sqr_16` | Pass |
| **Tags** | **Base tests** | |
| | /1 | |

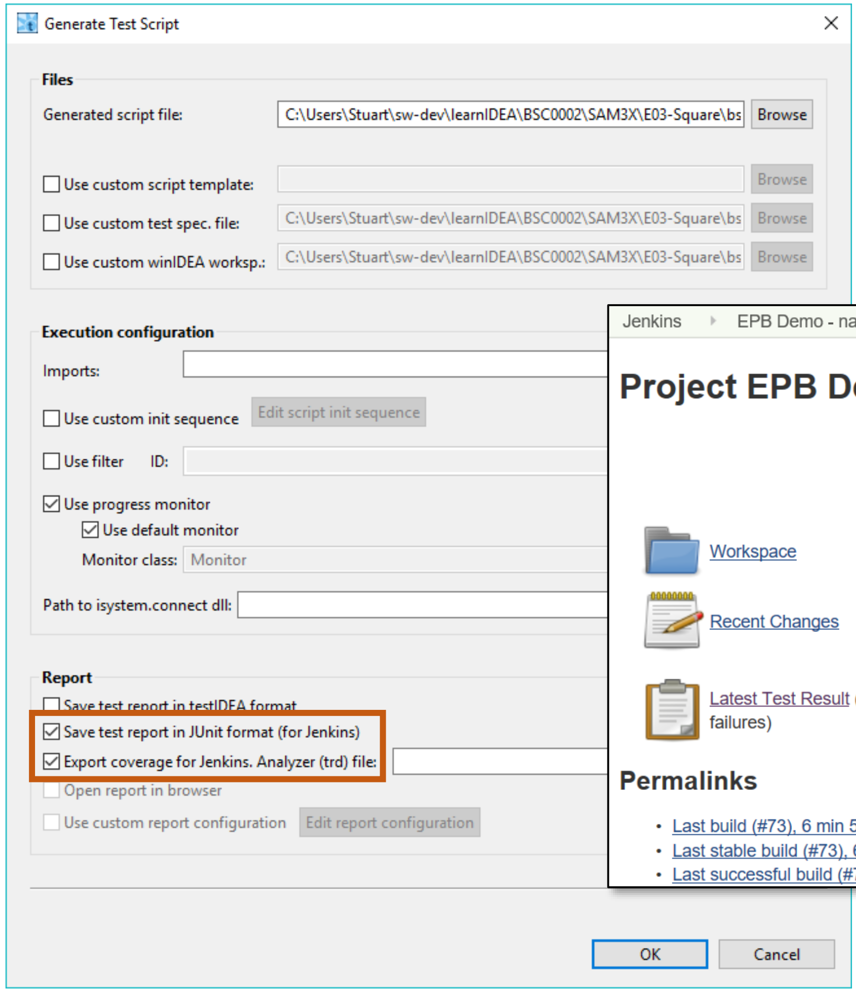
# 5 SUPPORTING CI TOOLS SUCH AS JENKINS

Test reports can be displayed automatically in tools such as Jenkins. Together with the *JUnit* plugin, the number of passed and failed tests can be shown directly on the project’s landing page.

To generate such files, simply select the *Save test report in JUnit format* in the *Generate Test Script* dialog.

There is also some support for the *Cobertura* plugin that displays the progress in code coverage achieved. This is enabled by the *Export coverage for Jenkins* option.

To find out more, search for *“Jenkins”* in the testIDEA help.



Generate Test Script

**Files**

Generated script file: C:\Users\Stuart\sw-dev\learnIDEA\BSC0002\SAM3X\E03-Square\bs [Browse]

☐ Use custom script template: [Browse]

☐ Use custom test spec. file: C:\Users\Stuart\sw-dev\learnIDEA\BSC0002\SAM3X\E03-Square\bs [Browse]

☐ Use custom winIDEA worksp.: C:\Users\Stuart\sw-dev\learnIDEA\BSC0002\SAM3X\E03-Square\bs [Browse]

**Execution configuration**

Imports:

☐ Use custom init sequence [Edit script init sequence]

☐ Use filter ID:

☑ Use progress monitor

☑ Use default monitor

Monitor class: Monitor

Path to isystem.connect dll:

**Report**

☐ Save test report in testIDEA format

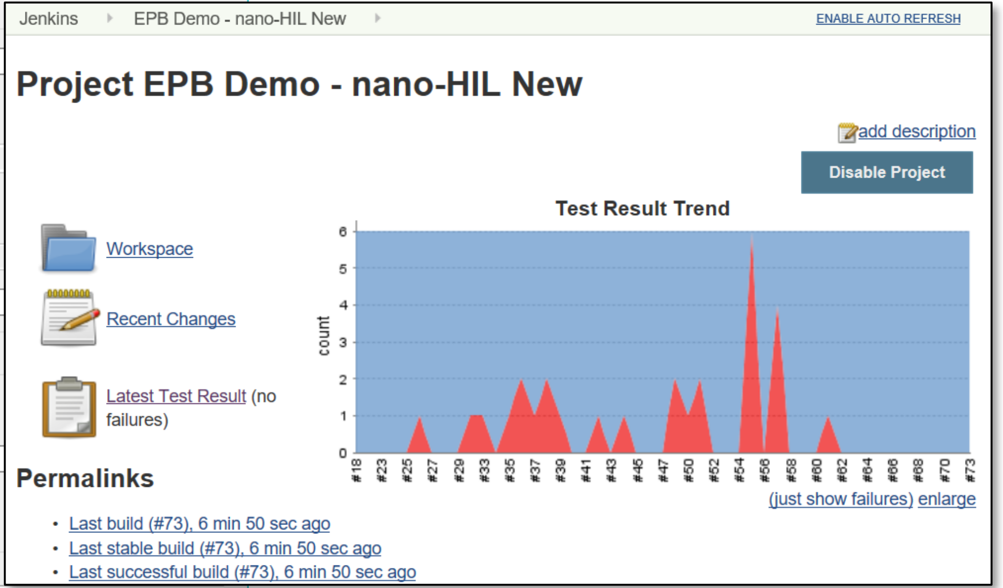
☑ Save test report in JUnit format (for Jenkins)

☑ Export coverage for Jenkins. Analyzer (trd) file:

☐ Open report in browser

☐ Use custom report configuration [Edit report configuration]

[OK] [Cancel]



Jenkins › EPB Demo - nano-HIL New › ENABLE AUTO REFRESH

**Project EPB Demo - nano-HIL New**

add description

Disable Project

Workspace

Recent Changes

Latest Test Result (no failures)

**Permalinks**

- Last build (#73), 6 min 50 sec ago
- Last stable build (#73), 6 min 50 sec ago
- Last successful build (#73), 6 min 50 sec ago

## SUMMARY

testIDEA

# 6 SUMMARY

- To support automation tools, test vectors can be exported as Python scripts
- Python scripts can be executed from the command line or from within CI tools such as Python
- Scripts can also generate test reports in a variety of formats