RANOREX

HANDS-ON APPLICATION TOPICS USERGUIDE

TABLE OF CONTENTS

HANDS-ON APPLICATION TOPICS **3**

STRUCTURE YOUR TESTS 4

Structure types and choosing one *5*

Best practices for structuring *10*

SOLUTIONS TO COMMON PROBLEMS 23

BEST PRACTICES INTRODUCTION 29

Creating a Ranorex snapshot *29*

Creating a compressed Ranorex solution *30*

Creating a compressed Ranorex report *31*

Add a solution settings file to a solution *35*

Fix 'element not found' error *35*

RANOREX CODE EXAMPLES 41

Hands-on application topics

In this section, you'll find practice-oriented recommendations and solutions for problems that help you create tests of high quality, reliability, and performance. The information in the contained chapters is based on numerous customer project analyses and advice from Ranorex experts with years of experience.

Structure your test

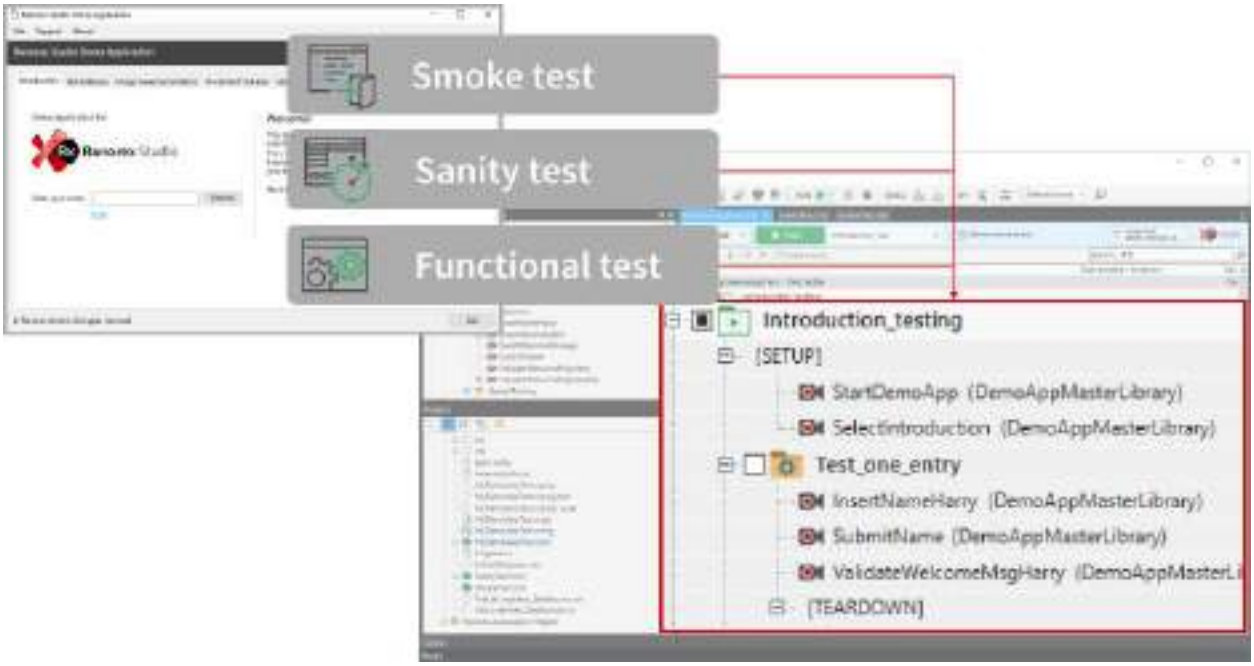
Solutions to common problems

Best Practices

Ranorex Code Examples

Structure your tests

This chapter deals with the question of how to translate a planned test into a test structure in Ranorex Studio. You'll learn the ways you can structure your tests, which structures are useful for which purposes, and best practices in structuring your tests.



Right away we need to emphasize that, unfortunately, there is no one answer to the question of how to structure your tests. There are several different basic structure models that have different advantages and disadvantages. Which one you choose depends mostly on what you want to test and how you want to test it.

Recommended knowledge

To get the most out of this chapter, you should be familiar with the following topics and their terminology:

Reference

Solutions and projects in Ranorex studio are explained in
Ranorex Studio fundamentals > Ranorex Studio > ⇢ Create a new solution
Ranorex Studio fundamentals > Ranorex Studio > ⇢ Create a new project

Reference

Test suites and their items are explained in

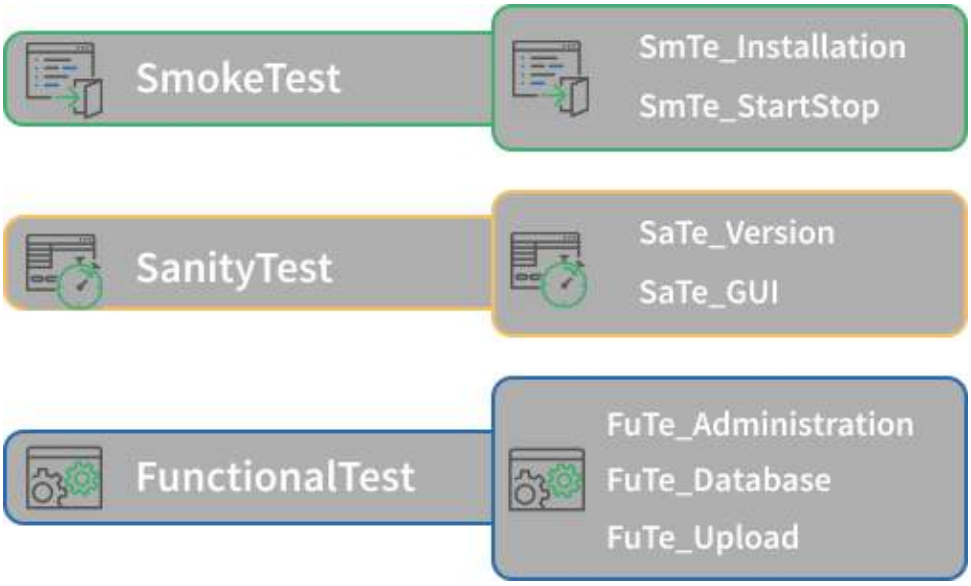
Ranorex Studio fundamentals > Ranorex Studio Overview > ⇢ Test suite structure

Structure types and choosing one

On this page, you'll learn the three basic ways of structuring your test in Ranorex Studio and how to determine which one works best for your requirements.

Example test requirements

To compare the different test structures, we'll use the following test requirements:



First, **two smoke tests** ensure that the AUT is installed correctly and can be started and closed by Ranorex Studio.

Then, **two sanity tests** ensure the correct version of the AUT is used and that all UI elements are implemented and work according to specifications.

Finally, **three functional UI tests** ensure that the AUT behaves correctly for its administration, database, and file importing functions.

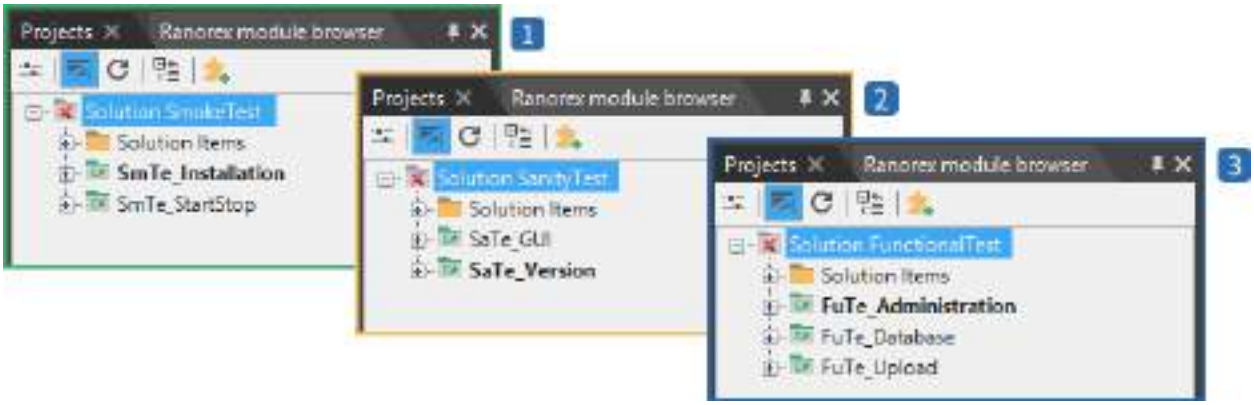
Three basic test structures

Here we list the advantages and disadvantages of each structure type and give an example of how the above test requirements could look when translated into those structures.

Again, no single structure is the best. It all depends on your requirements. With what follows, you should be able to make an informed choice regarding which structure is appropriate for you.

Solution-based structure

In this type, the primary structure is based on creating one solution for each of the three different test types. The actual tests are then structured in projects, with one test suite per project.



1. Ranorex Studio solution for the **smoke tests**
2. Ranorex Studio solution for the **sanity tests**
3. Ranorex Studio solution for the **functional tests**

Advantages

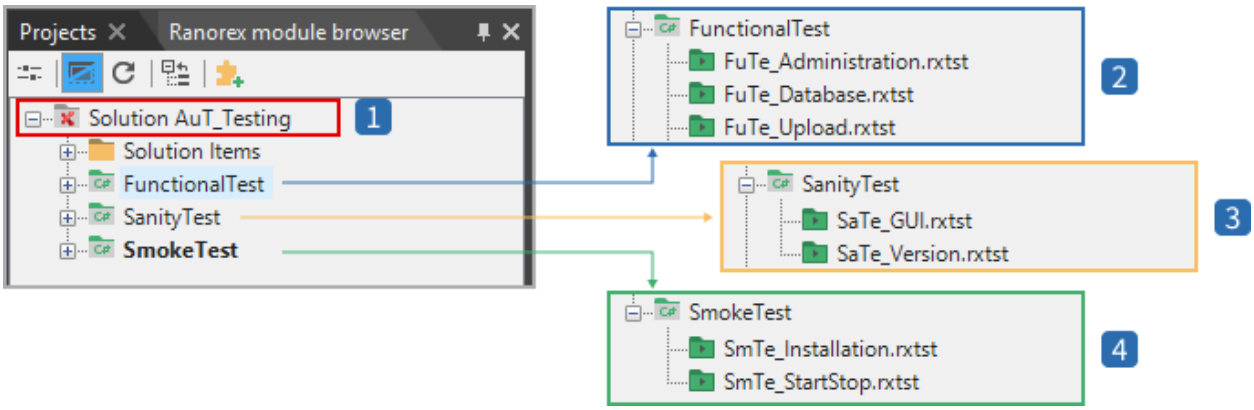
- Very clear structure at the top level. Useful for extensive and complex test requirements.
- Particularly useful when testing an AUT that has different UIs and requires multiple repositories, e.g. testing frontend and backend UIs.
- Easier to keep test suite structures simple and efficient.
- Can make test execution quicker.

Disadvantages

- Higher maintenance: Every solution has its own settings, repository, and module library that you need to take care of.
- It's not possible to execute all three solutions automatically in sequence from Ranorex Studio. You can only do so with external scripts, e.g. in a continuous integration environment.
- Each solution has its own reports. This can make it harder to collect and present results.
- There is no easy way to exchange data between the three functional tests.

Project-based structure

In this type, the primary structure is based on creating one solution that contains one project for each of the three different test types. The actual tests are then structured in multiple test suites, with one test suite per test.



1. The solution **AuT_Testing** contains all three projects
2. Project **FunctionalTest** for the implementation of the three **functional tests**
3. Project **SanityTest** for the implementation of the two **sanity tests**
4. Project **SmokeTest** for the implementation of the two **smoke tests**

Advantages

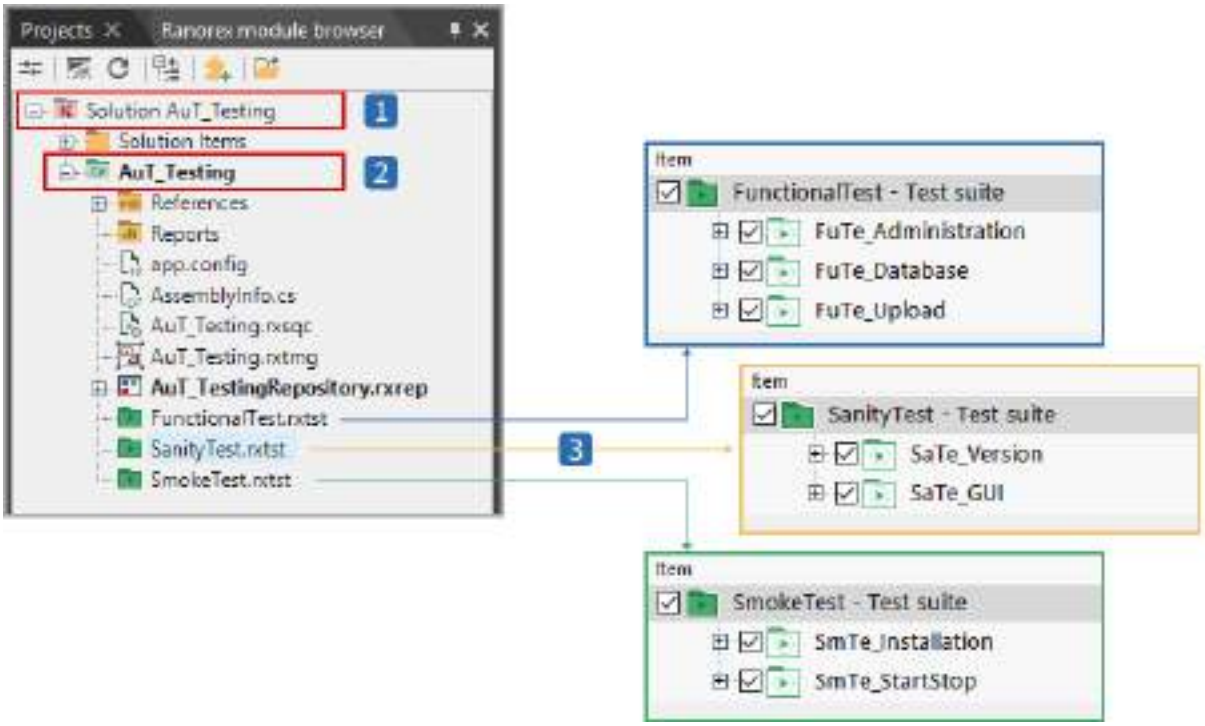
- Convenient: One solution contains all the tests.
- Easier maintenance: No differing settings, and usually only one module library and repository.
- Even though the structure is less clear at the solution level, it's very clear at the project level. Therefore, keeps the advantage of simple and efficient test suite structures.
- You can easily run the projects and the contained test suites in sequence from Ranorex Studio by using the test sequence functionality .

Disadvantages

- Each project/test suite has its own reports. This can make it harder to collect and present results.
- There is no easy way to exchange data between the three functional tests.

Test suite-based structure

In this type, the primary structure is based on creating one solution that contains one project. This project then contains one test suite for each of the three different test types. The actual tests are then structured in test cases inside those test suites.



1. The solution **AuT_Testing** contains one project
2. The project **AuT_Testing** contains one test suite per test type.
3. The individual test suites contain the actual tests as test cases.

Advantages

- Convenient: One solution contains all the tests.
- Easier maintenance: No differing settings, and usually only one module library and repository.
- You can easily run the test suites in sequence from Ranorex Studio by using the test sequence functionality.
- Exchanging data between the three functional tests is now easy, but still difficult between the different test types.

Disadvantages

- Quickly leads to bloated and overly complex test suite structures, making maintenance, troubleshooting, and collaboration much harder and affecting test performance negatively.

Which test structure do I choose?

This section describes the factors you should consider when choosing your structure.

Avoid extremes

For most requirements, the project-based structure is a good choice.

It offers a flexible approach with good organization, simple test logic, and high convenience.

Both the purely solution-based and the test suite-based models are extremes. We recommend using them only if your test requirements fit them perfectly.

Use the solution-based structure for very large, complex tests. The very high maintenance required by this approach is the most important disadvantage. You have different settings, repositories, and module libraries for each solution.

Use the test suite-based structure for small, simple tests. It's easy to set up a test suite-based structure and quickly get a test running. Maintenance is also low — but only if your

tests stay small and simple! With more complexity, you quickly get bloated, hard to manage test suites.

Consider your UI elements

The UI elements involved in your tests are one of the more important criteria when choosing a structure. If your tests deal with multiple different UIs, like a frontend and a backend, it makes sense to use multiple repositories.

With multiple repositories, the solution-based structure becomes more attractive. However, you can also use multiple repositories in the other structure types. So if your test isn't very complex despite two different UIs, a project-based structure with two repositories is probably still better.

Data exchange as a requirement is overrated

Exchanging data between tests is hard when they're not in the same test suite. However, it's often not truly necessary. You can avoid data exchange by designing reference data and test cases differently.

Best practices for structuring

On this page, you'll find tips on how to structure your tests for better quality, maintainability, troubleshooting, and collaboration.

Three principles of test design

Follow these principles to get tests that are well structured, easy to maintain, and more efficient – regardless of test size or complexity.

Keep it simple

As simple as possible, as complex as necessary. Apply this to test structures, test cases, modules, everything.

Keep it logical

Your structures, test cases, modules, etc. should make sense in terms of your test requirements. Don't split up items unnecessarily (e.g. having username/password input and clicking a "Log in" button in two different modules). Don't mix things that should be separate (e.g. putting an entire webshop end-to-end workflow in one module).

Be lazy

Reuse anything you can. Always make reusability a priority.

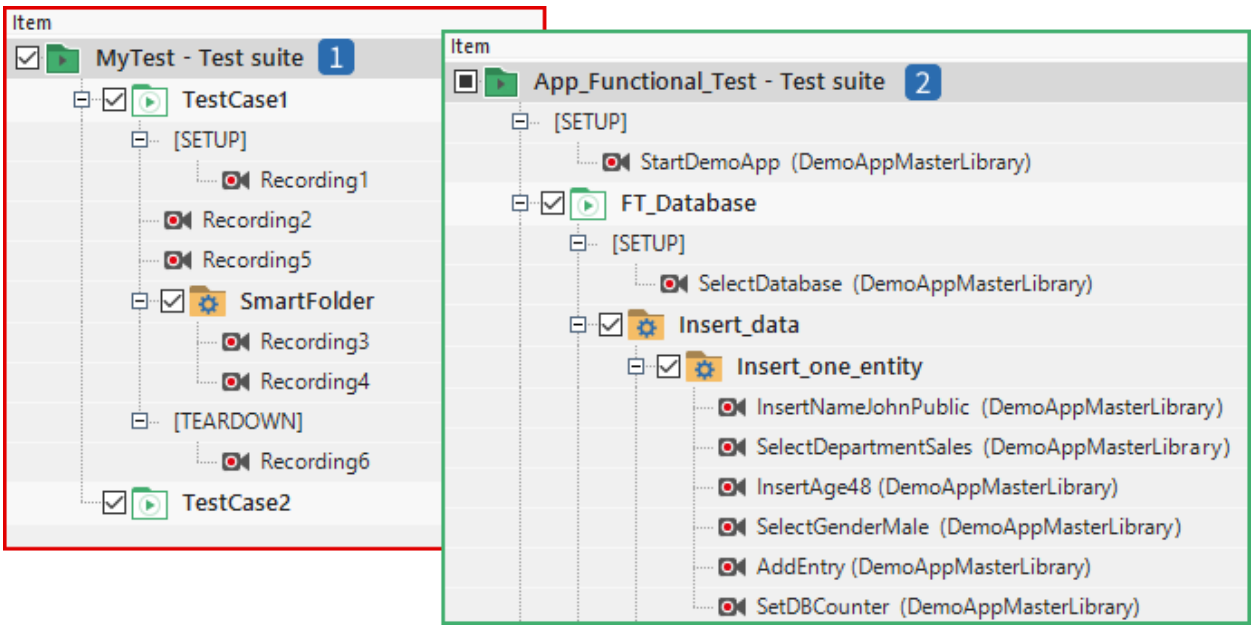
Document your test

Ranorex Studio offers several ways of documenting your test with naming, descriptions and special reporting actions. These may seem unnecessary at first, but good documentation keeps your test suite organized, greatly increases maintainability, and makes it much easier to work in teams.

Undocumented tests are unfinished tests.

Name everything

Always give your test containers, modules, repository items, data connectors, etc. good names. When working in teams, use naming conventions. Default names are okay in the short term, but never in the long term.



1. Test suite with only default names. Confusing, extremely hard to maintain, a nightmare for collaboration
2. Test suite with good names for all items. You can see what everything does at a glance

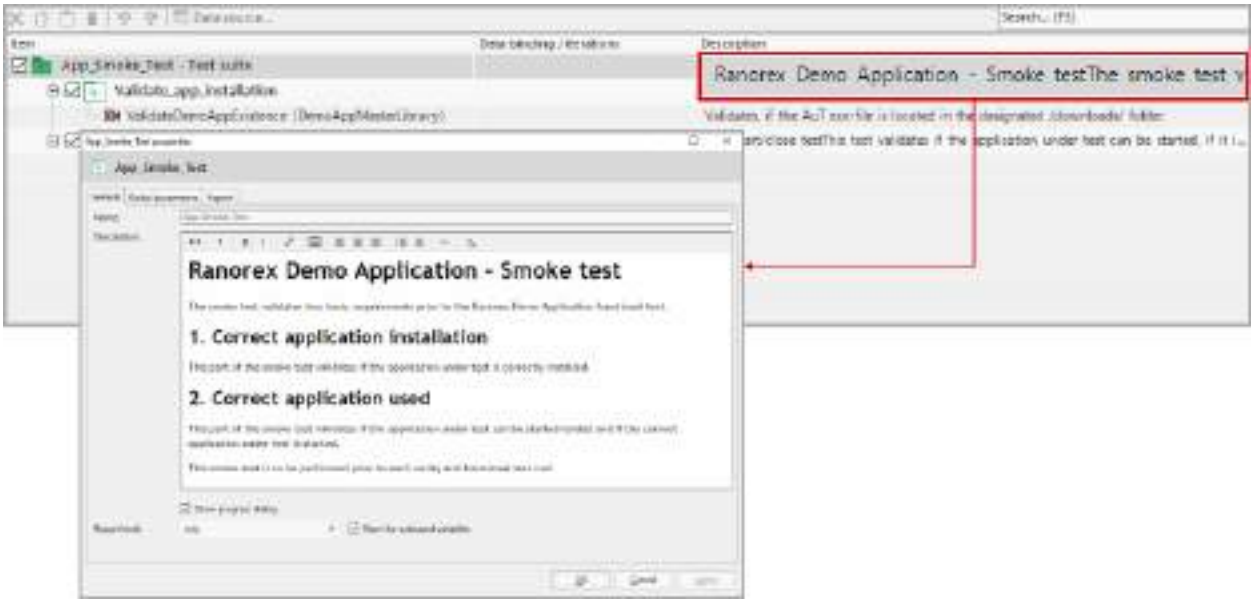
Describe everything

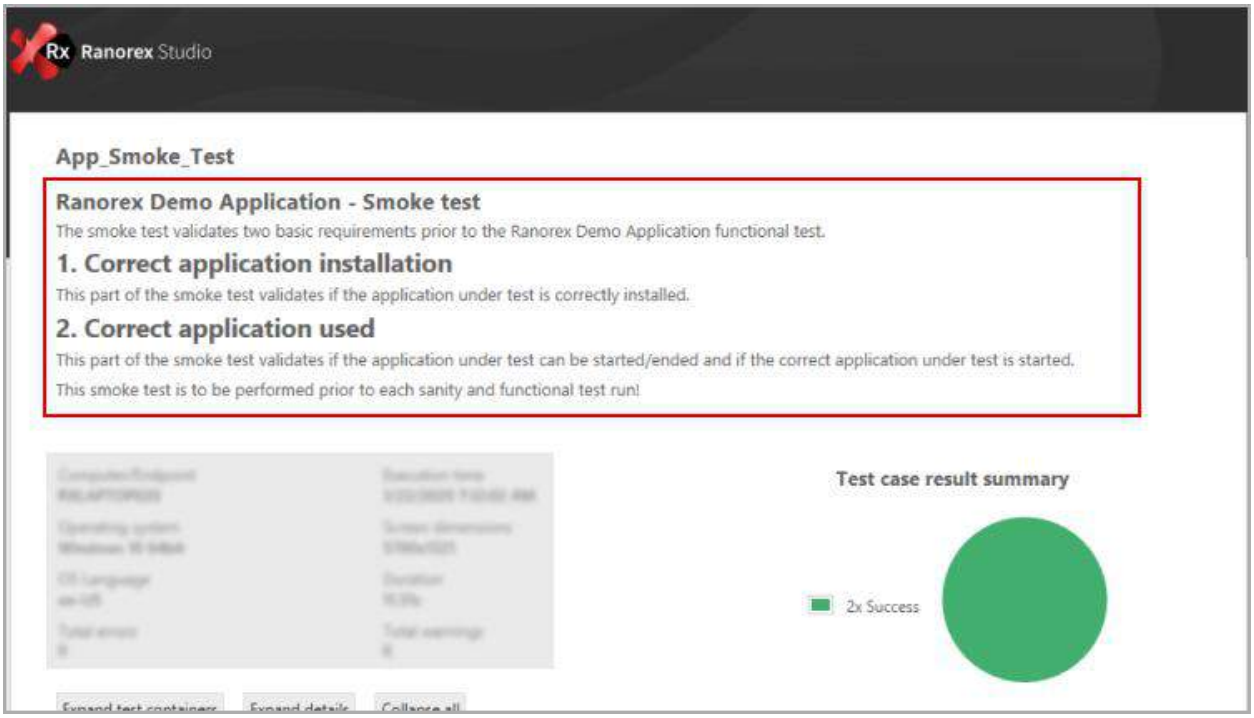
In the test suite view, you can add descriptions to the test suite itself, test containers, and modules. Descriptions are also automatically included in the report, making it easier to understand.

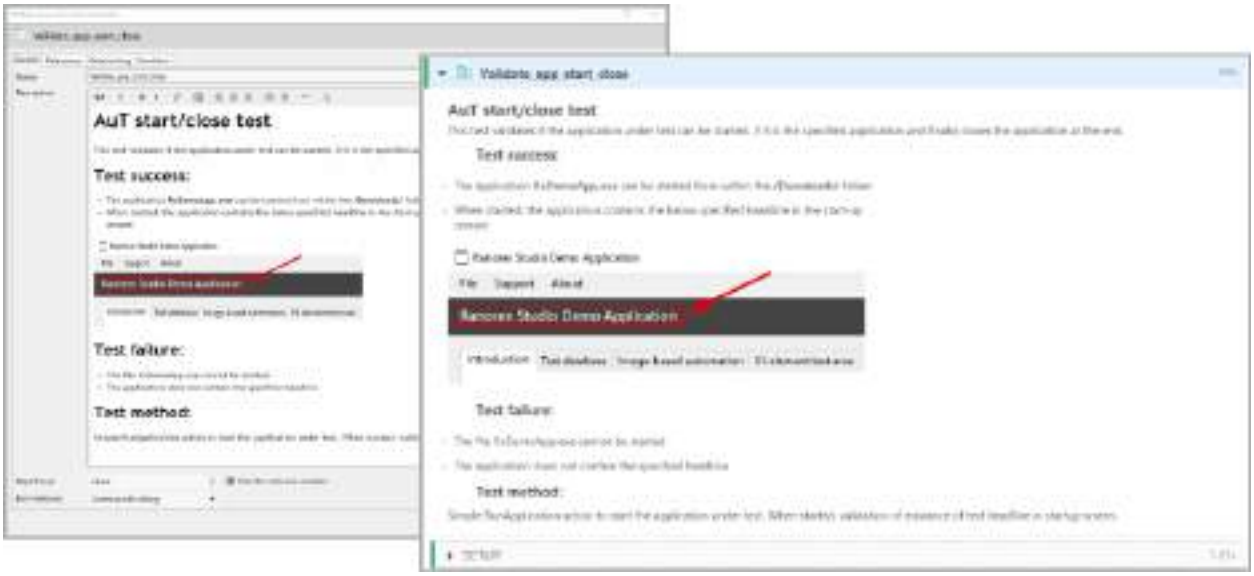
To add descriptions:

1. **Double-click** in the description column next to an item.
2. An editor opens that offers various formatting options. You can even include images.

Some examples of descriptions for the different items:





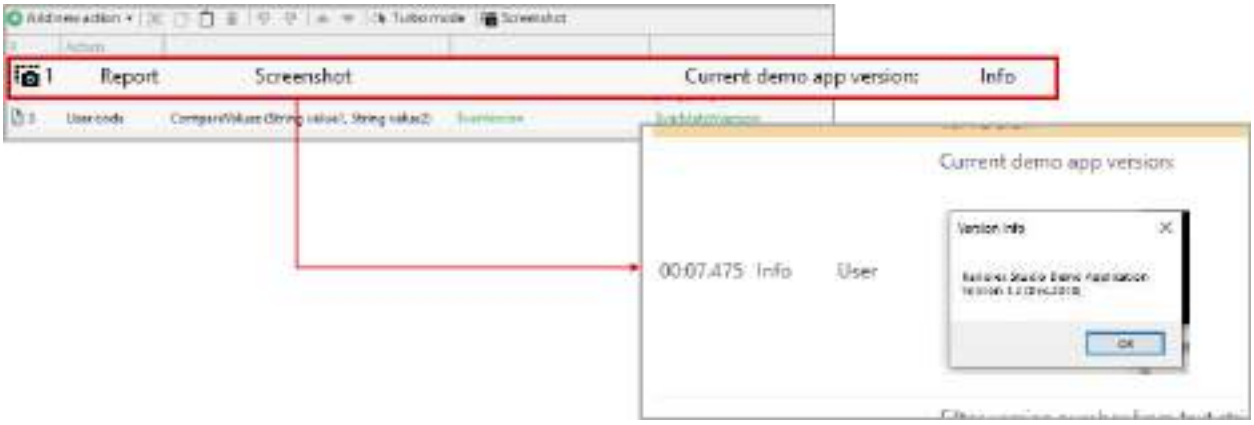


Add manual reporting actions

With the Capture screenshot and Log actions, you can add images and messages to the report manually. They can make the report easier to read, especially for other people who may not be as familiar with the test structure.

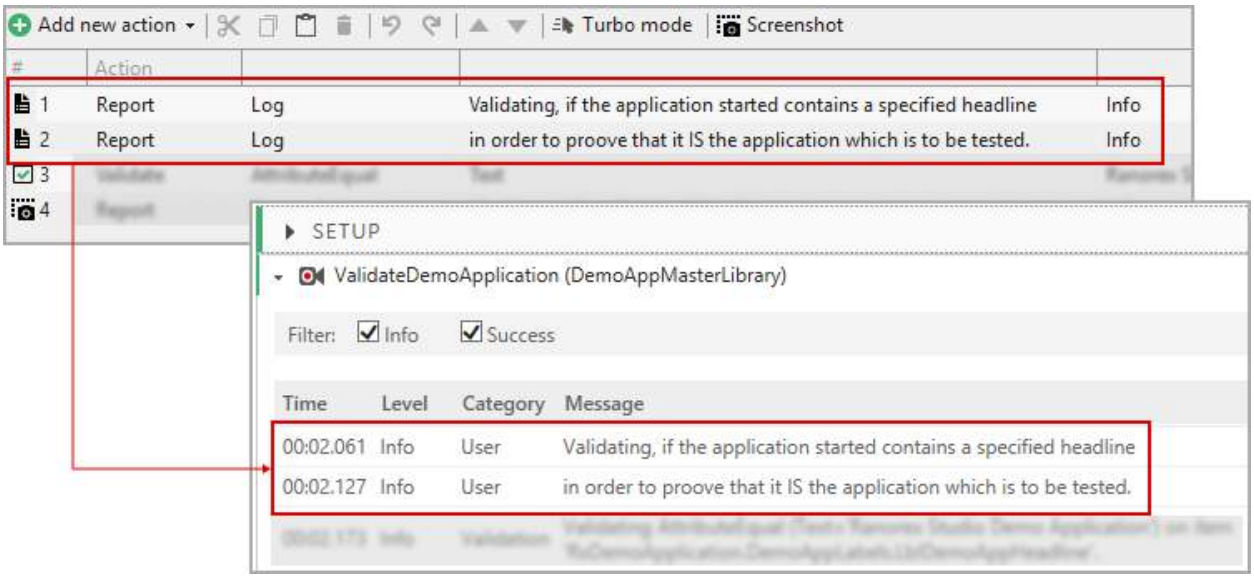
Add Capture screenshot actions

Ranorex Studio normally only includes screenshots in the report if an error or a failure occurred. With the screenshot action, you can insert screenshots at any point, e.g. to show which UI element an action was performed on.



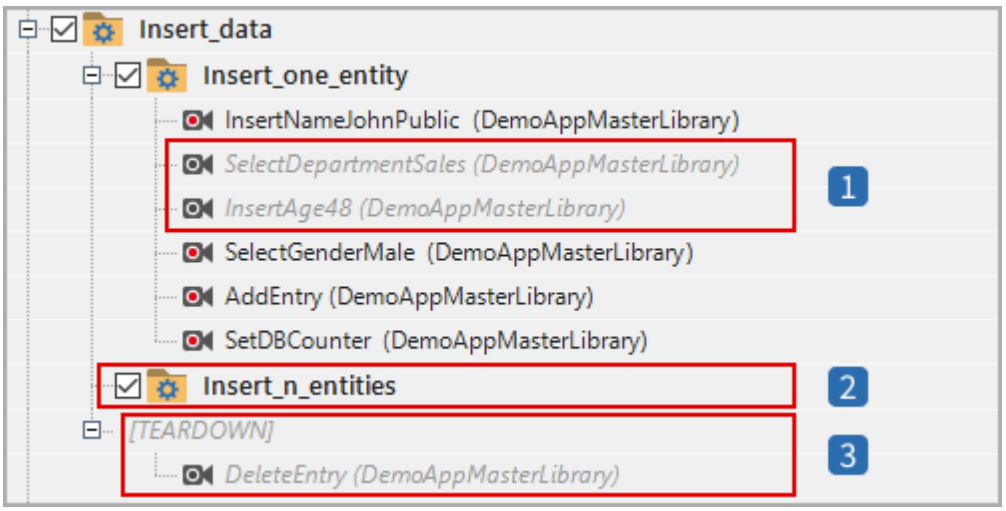
Add Log actions

Ranorex Studio already documents test runs quite extensively, but sometimes, you may want to include additional information. You can do so easily with the Log action.



Don’t leave items empty/deactivated

A finished, working test suite should not have empty or deactivated items in it. Empty items are okay if they’re not yet finished. Deactivating items is okay to show they still need work or for troubleshooting.

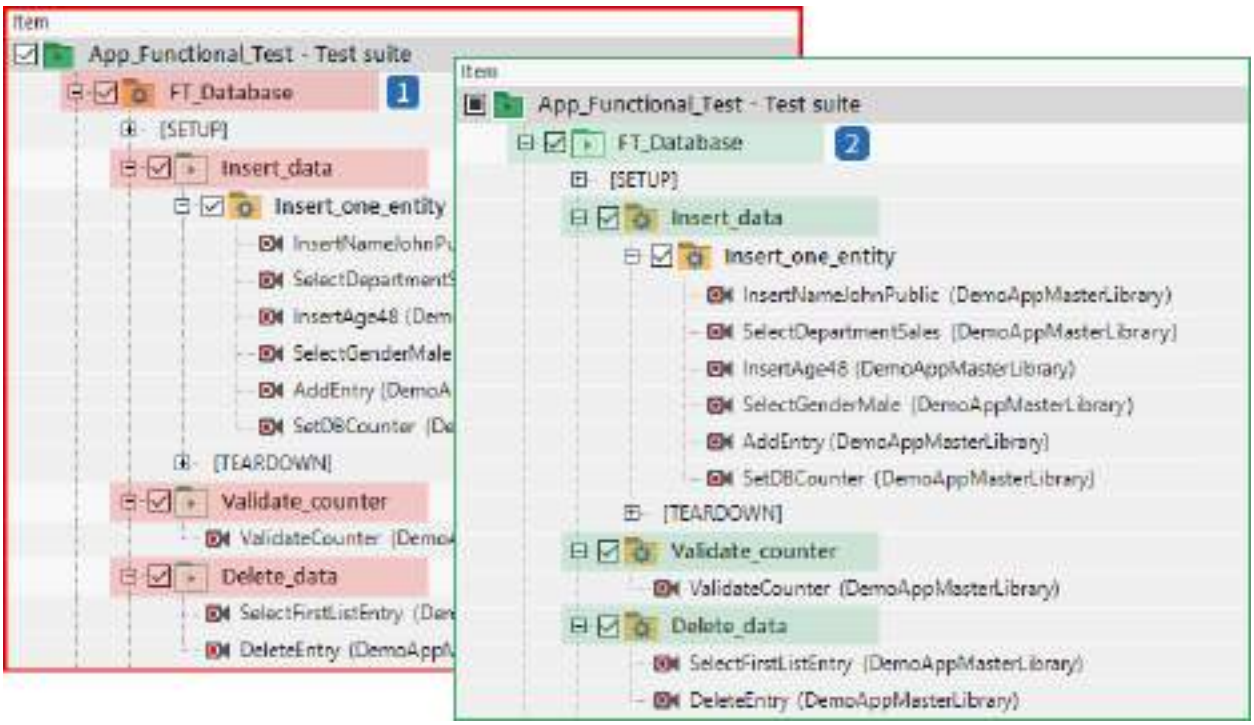


1. Deactivated recording module
2. Empty smart folder
3. Deactivated teardown region

Use the test case as the primary test container

Test cases represent a primary function of your test, like adding entries to a database. Smart folders are for further structuring of test cases.

> **Note**
>
> In the technical sense, the main difference between test cases and smart folders is that only test cases are counted in the success diagram in the report. This reflects the role of test cases as the primary test container.



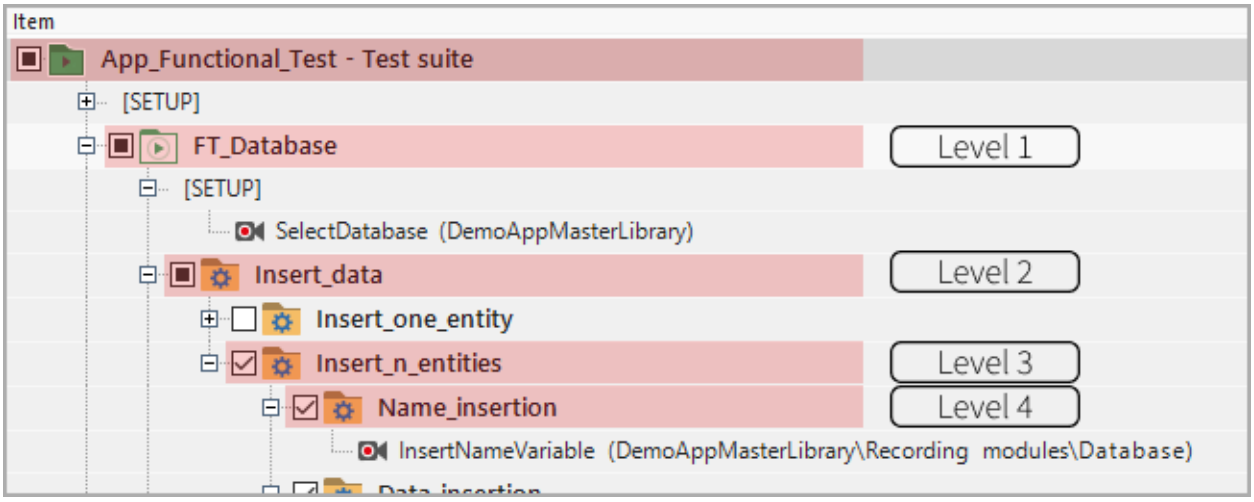
1 Not ideal: Using a smart folder as the primary test container, with the test case as a subitem

2 Ideal: Test case as the primary test container, with smart folders as subitems for structuring

Avoid deep test suite structures

As a general rule, you should avoid going deeper than 5-6 test container levels. Remember: Keep it simple.

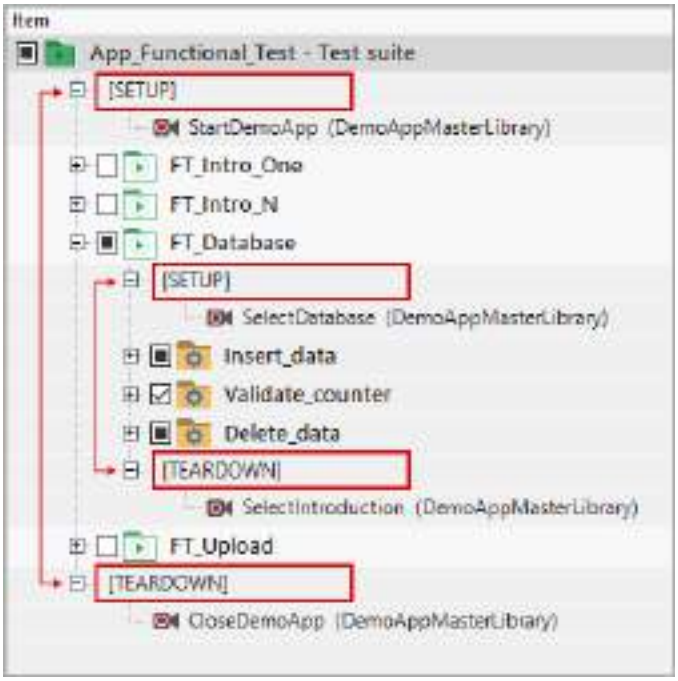
Complex test requirements may seem like they make complex test suite structures necessary, but that's not entirely true. On the ⇢ previous page, we described different test structure types and how you can use projects or even solutions to move away from a test suite-based structure. This allows you to keep test suites simpler, which also keeps maintenance manageable and collaboration feasible.



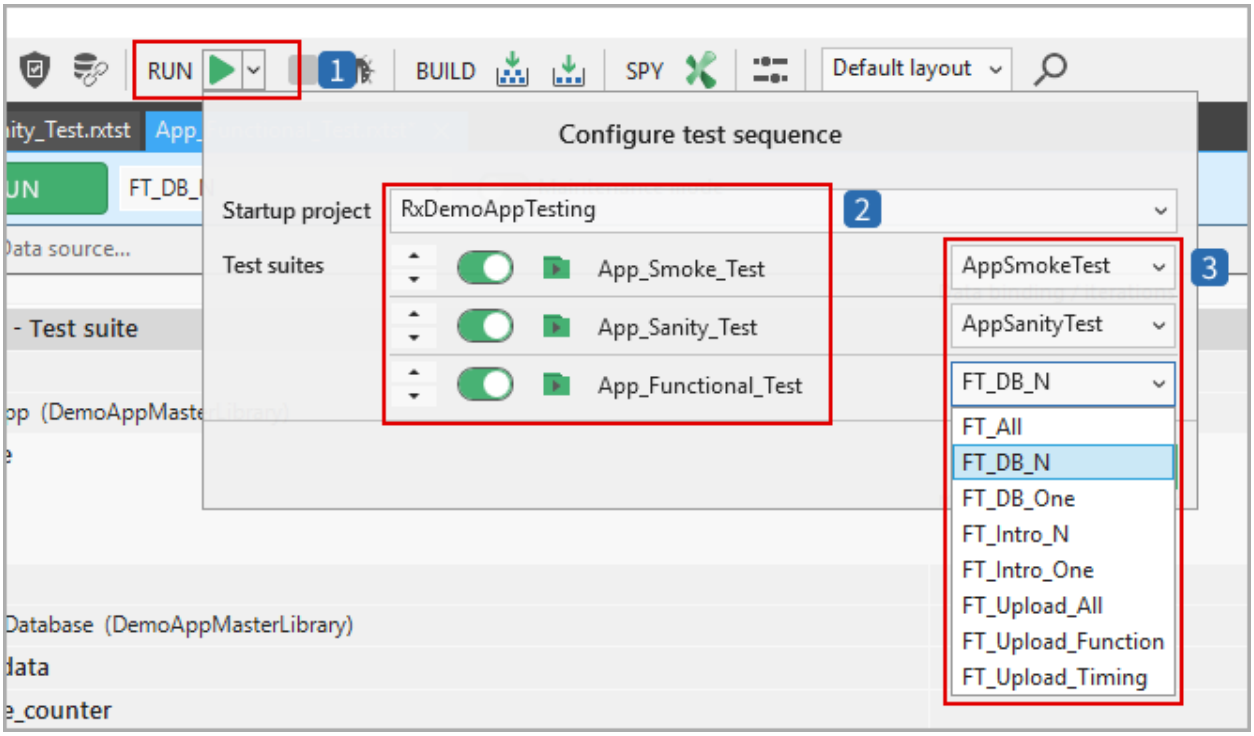
Keep test cases independent

When possible, test cases should not depend on other test cases. This makes troubleshooting and running tests in different configurations easier.

One way to do so is by using ⇢ setup and teardown regions for test suites and test containers. It's a best practice to have test cases wrapped in at least one setup/teardown region.



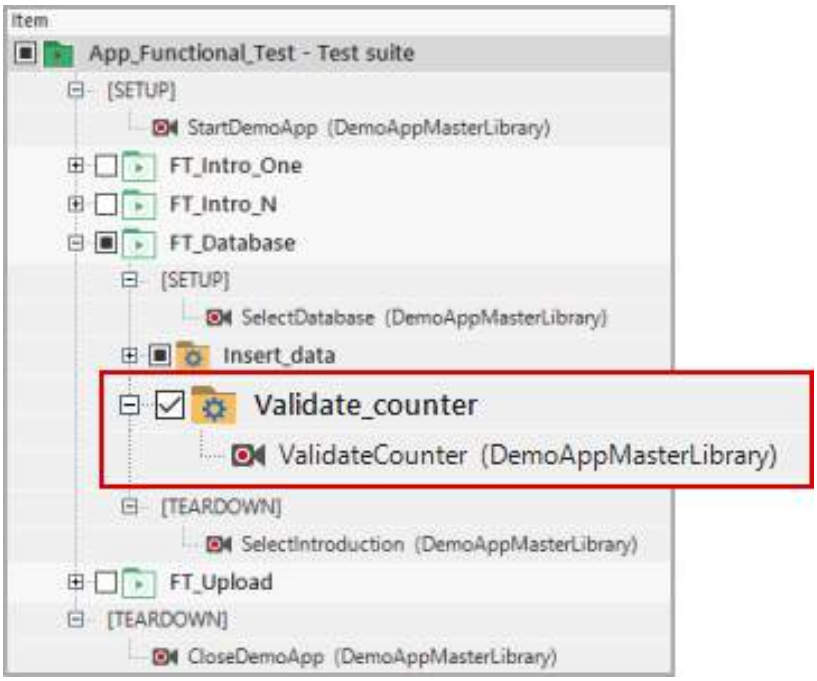
You can run tests in different configurations with ⇢ test sequences and ⇢ run configurations.



1. Access to test sequence configuration (possible if the project contains 2 or more test suites)
2. Test sequence configuration
3. Run configuration selection per test suite

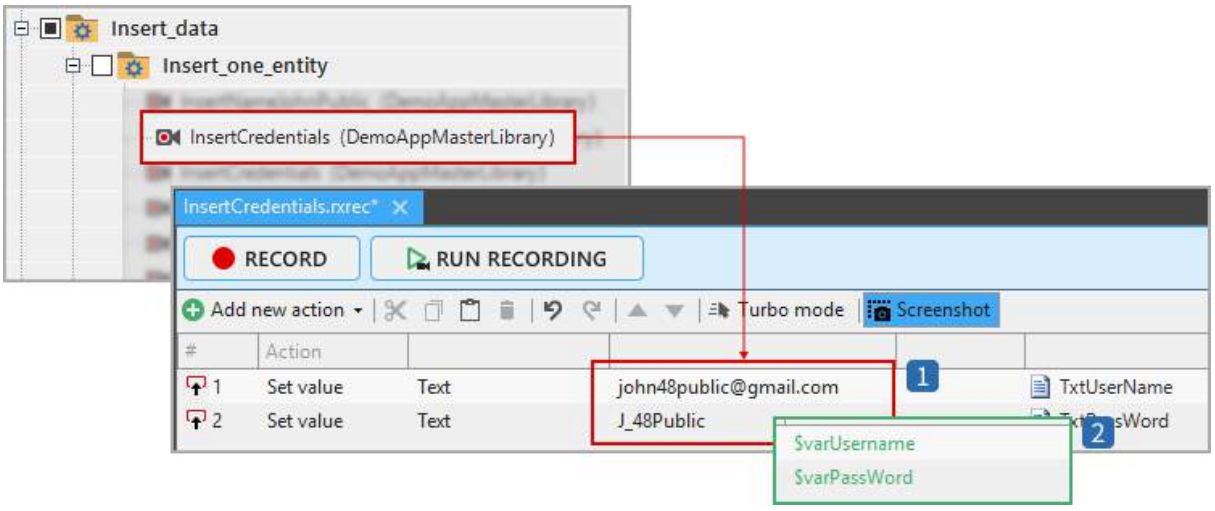
Use validations

Testing software is always about comparing a desired to an actual behavior/state. A test case without a ⇢ validation is a test case without purpose.



Use variables

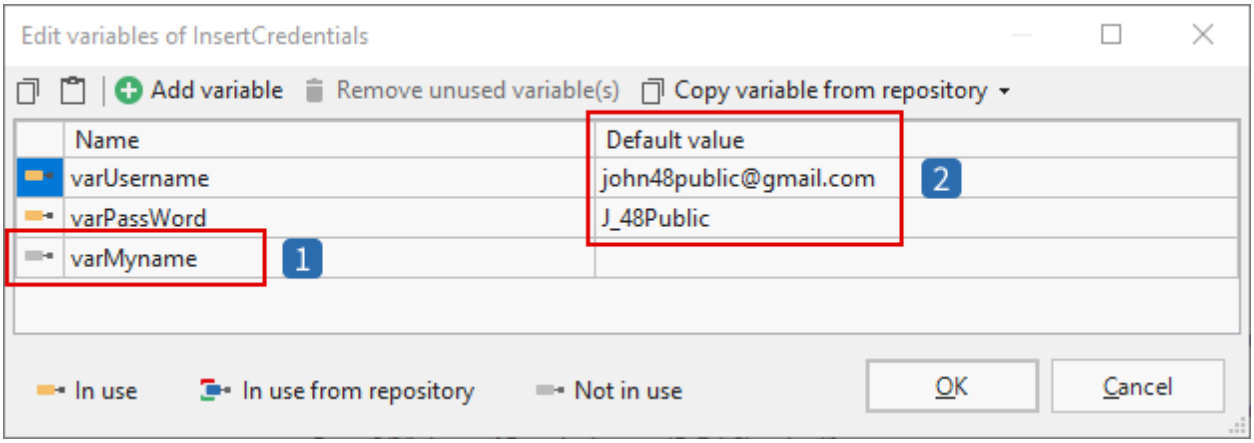
Use variables instead of constant values in modules to make them reusable and your test more flexible.



1. Constant strings for two values set as part of the module
2. Replacing them with variables makes the module reusable

Maintain your variables

Regularly clean up your variables and delete unused ones. Make sure all variables have sensible default values. This is important when values from a bound data column can't be retrieved, like when running a recording module directly and not as part of a test suite.



1 Unused variable

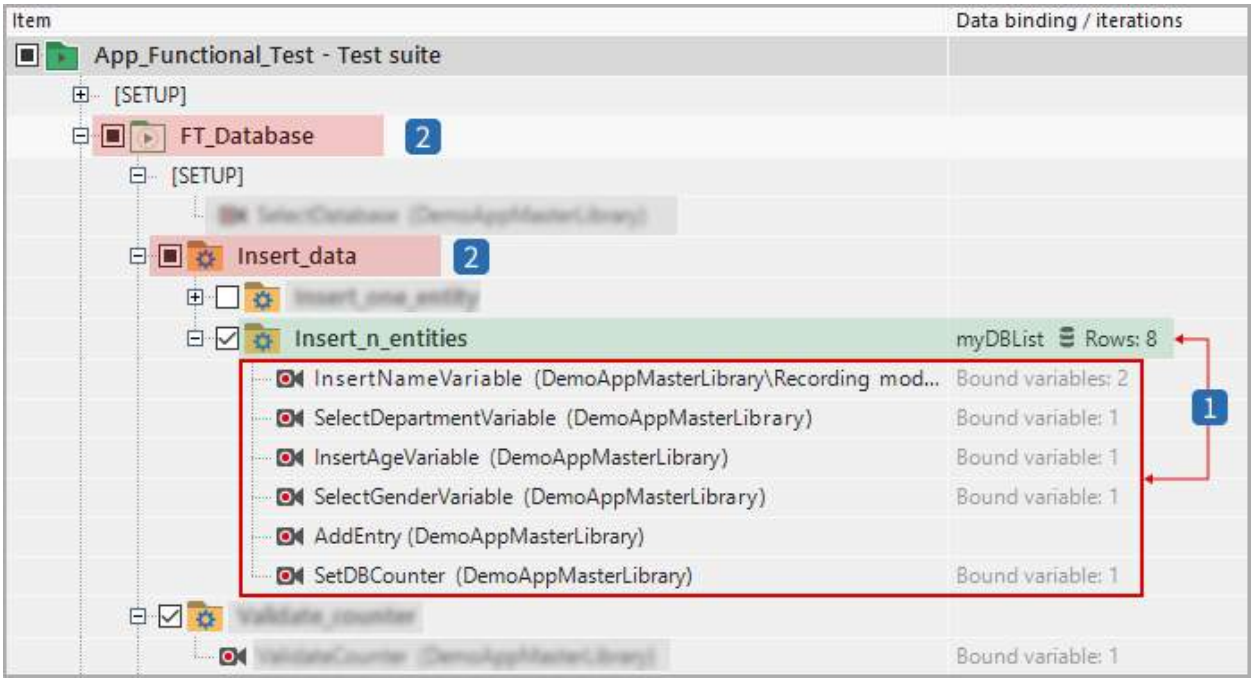
2 The variables' default values

Keep data sources close to where they're used

Assign data sources or parameters as close to the test containers where they'll feed values to variables. **At worst, a data source should be no more than 2 test container levels away from where it's used.**

The descendants of a test container inherit its data sources. However, over several levels, it gets quite hard to tell what is going on.

> **Note**
>
> With data-driven testing in general, it's less the amount of data you feed into the test, but rather the quality that determines whether you'll get significant results. The data should cover as many different test scenarios as possible.



1 Ideal: the data source is assigned to the test container that contains the bound variables

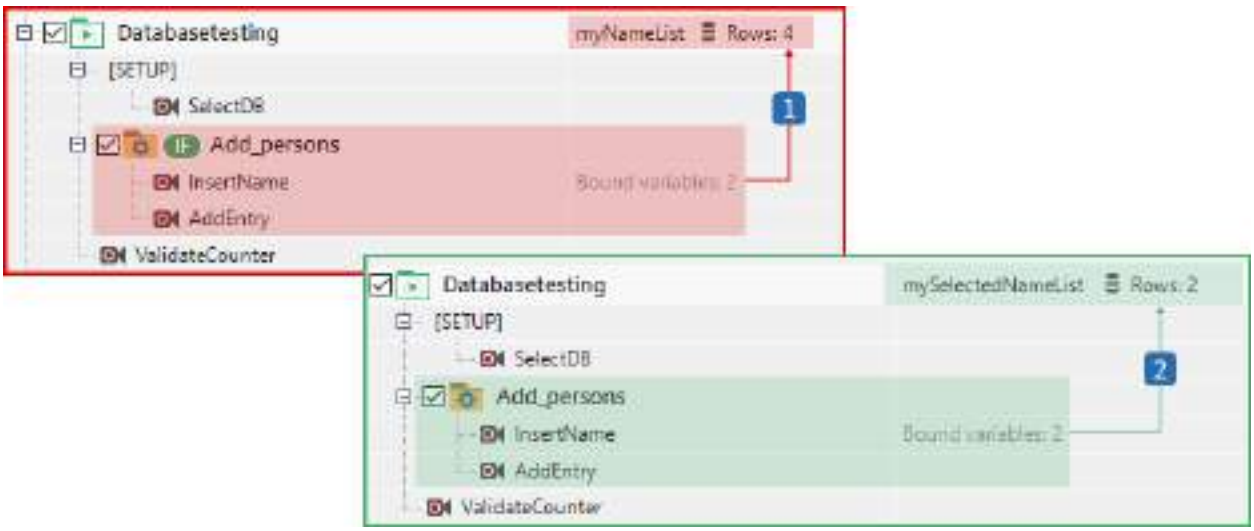
2 Avoid assigning the data source to test containers farther away, like these two

Think twice before using conditions

Sometimes, conditions are necessary, but often, they **can be avoided with a different test container structure.**

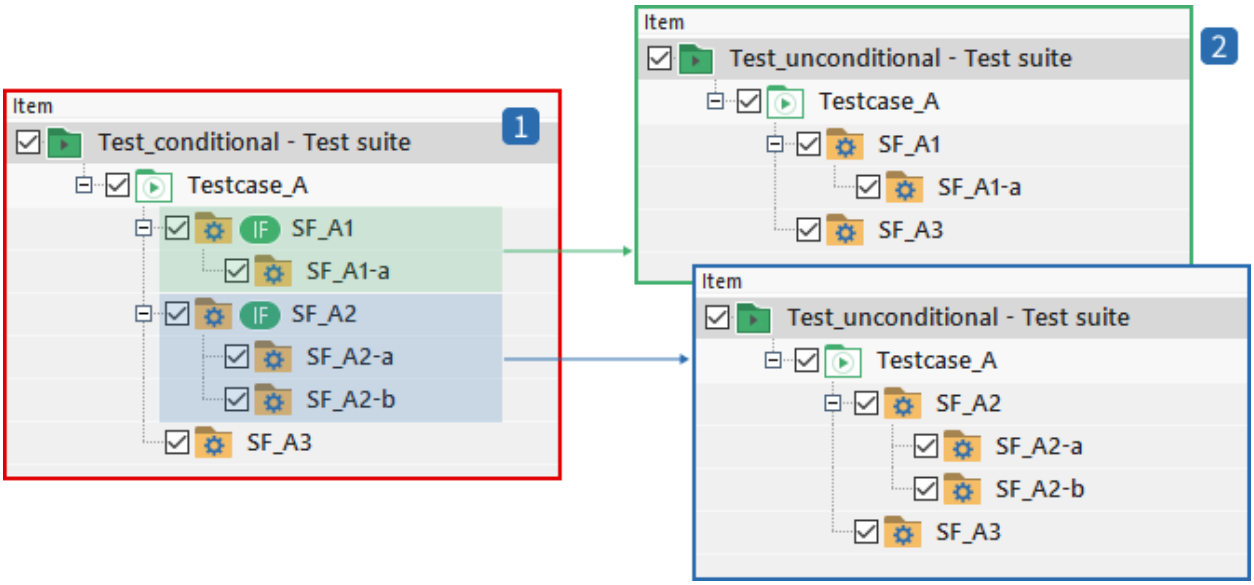
Conditions increase complexity, make maintainability worse, and are often an issue in collaboration. This is especially true when conditions extend over several test container levels. **At worst, conditions should extend over no more than 2 nested test container levels.**

Here are two examples:



1 The test container only executes a part of the data source based on a condition

2 It's better to remove the condition and adjust the data source so it only contains the data required



1 Here, we have a test case where descendant structures are executed only if conditions are fulfilled

2 We can replace this by splitting up the test case into two separate ones and replicating the structure with test containers that are always executed

Solutions to common problems

This topic is an information container where solutions to problems or test challenges are addressed which have not yet been described in detail in the Ranorex documentation, user guide, release notes, or eLearning modules. Usually, this chapter hosts upcoming problems primarily addressed within the forum or through support queries. This chapter is your first address when not finding information about your problem or question before contacting the forum or Ranorex support.

Technology limitation warning

Technology limitation warnings usually have their origin in an incomplete instrumentation. We recommend following the below-listed link before contacting the Ranorex support team.

Further reading

Technology limitation warnings are addressed in > Interfaces & connectivity > Technology instrumentation > ⇢ General troubleshooting.

How does Ranorex identify UI-elements?

Ranorex tracks, identifies and stores UI-elements in a two-step process. Basically, UI-elements are referenced by an inner representation called repository items. Each repository item abstracts the corresponding UI-element with a role and optional capabilities. The position within the GUI is described by a special RanoreXPath specifier which uniquely identifies each UI-element.

Further reading

We recommend start reading about UI-elements in the fundamental chapter > Ranorex Studio fundamentals > Repository > ⇢ Introduction.

If you are interested in more advanced information according to this topic, please refer to > Ranorex Studio advanced > UI-elements > ⇢ Introduction.

Is it required to use RanoreXPath for test automation?

No. It's also possible to search for UI-elements or forms using a number of different `Find` methods to search and/or filter for UI-elements.

> **Further reading**
>
> The user guide chapter 'Code examples' provides the necessary information > Hands-on application topics > ⇢ Code examples.

Besides this, we provide a fundamental and easy-to-understand introduction to RanoreXPath with lots of syntax examples for you to use, apply and raise your understanding. Once you are familiar with RanoreXPath, you will find it more easily to perform your UI-elements tracking challenges.

> **Further reading**
>
> The concept of RanoreXPath is introduced and explained in detail in > Ranorex Studio advanced > RanoreXPath > ⇢ Introduction.

Does Ranorex support data-driven testing?

Yes – data-driven testing is an advanced key concept of Ranorex. Ranorex supports data connectors for simple data tables, CSV-files, Excel files and SQL database files.

> **Further reading**
>
> Data-driven testing is introduced and explained in > Ranorex Studio advanced > Data-driven testing > ⇢ Introduction.

How can I speed up my Excel-based data connector?

Especially with large files, the performance of Microsoft's default file format for excel spreadsheets is getting poor. This weakness might also affect your data-driven test execution. For performance improvements we recommend to use the **binary file format (**xlsb**)** instead of the default one (xlsx). Simply save your Excel spreadsheet with the extension "xlsb" and assign it to your excel data source.

Further reading

More information on this topic can be found in > Ranorex Studio advanced > Data-driven testing > ⇢ Data & data management (#Excel data connector).

What to do if items can't be found during Ranorex test execution?

Further reading

More information on this topic can be found in > Hands-on application topics > Best practices > ⇢ Fix 'element not found' error.

Is it possible to extend recordings with user-specified code actions?

You can easily extend standard recordings with user specific code actions by converting existing action items or by adding a new user code action item to a recording.

Further reading

More information on this topic can be found in > Ranorex Studio fundamentals > Actions > ⇢ User code actions.

What is the difference between Adapter and Element?

In Ranorex, UI-elements are mapped against an inner representation which is called a repository item. UI-elements are described by roles, capabilities, attributes & values and the RanoreXPath specification. These important topics are addressed in corresponding chapters:

Further reading

If you want yourself get familiar with this important topic, we recommend start reading > Ranorex Studio advanced > UI-elements > ⇢ Introduction.

The advanced concept of RanoreXPath is introduced and explained in > Ranorex Studio advanced > RanoreXPath > ⇢ Introduction.

Note

Formerly, the term **adapter** (aka **adapter-type**) was used to describe the internal representation type of a UI-element. This definition has now been replaced by a more consistent specification distinguishing between roles, capabilities, attributes, and values.

The term was also used as part of the RanoreXPath specification which has now been replaced by the term '***role***'.

Is it possible to trigger Ranorex tests from an existing test or build environment?

The result of a Ranorex test automation project is always an executable file. The generated `*.exe` can easily be started from other environments supporting command line execution.

Further reading

Running tests without Ranorex Studio at command level is introduced and explained in > Ranorex Studio fundamentals > Test suite > ⇢ Running tests without Ranorex Studio.

Can I run my tests on machines where I am not allowed to install Ranorex?

Yes, it is possible to run automated tests on runtime machines without Ranorex Installation.

Further reading

The topic is addressed in > Interfaces & connectivity > ⇢ XCOPY deployment.

Can I use Ranorex libraries within Visual Studio?

That's one of the big advantages of using Ranorex. You're able to use your existing development environment to develop Ranorex based test automation code. Additionally, the code generated by the Ranorex Recorder or Ranorex Repository can easily be integrated into your Visual Studio projects.

Further reading

The topic is addressed in > Interfaces & connectivity > ⇢ Visual Studio integration.

What shall I do with unexpected dialogs and popup windows during test automation?

Ranorex provides the dedicated class `PopupWatcher` to watch for and to handle popup windows. By using this class, not only simply click actions to close popup dialogs can be called. Even more complex scenarios can be handled in custom callback routines being called at the time the popup appears.

Further reading

Read more about how to handle popup dialogs with Ranorex in > Hands-on application topics > ⇢ Code examples.

Is it possible to test Silverlight applications with Ranorex?

Yes, it is. Simply ensure that your Silverlight application does not run-in **window-less-mode**, i.e. set the `Windowless` property of the Silverlight HTML object to false. Find more information about window-less mode on the following site: msdn.microsoft.com

Is it possible to automate a webpage without moving the mouse pointer?

Yes, it is. Simply use the `PerformClick` instead of the normal `Click` method when working with web adapters like `DivTag`, `Input` or `Link`.

Further reading

This topic is addressed in > Ranorex Studio fundamentals > Actions > ⇢ Invoking actions.

What are the system requirements for developing and running Ranorex tests?

The following link to our online documentation shows what is needed to develop or to simply run Ranorex tests.

Further reading

Ranorex system requirements are listed in > Ranorex Studio system details > ⇢ System requirements.

Are there known incompatibilities with other software?

In general, there are no known incompatibilities. However, some antivirus or security software blocks certain Ranorex functionality. Consequently, if you experience problems with your automation and are running antivirus or security software, we recommend temporarily switching that software off for a test run.

Best practices introduction

Within this chapter, we focus on the detailed introduction and description of best practice methods which cannot be assigned to one single topic of Ranorex information structure but address a process, a guideline, or a method for creating, maintaining and performing more professional and efficient test cases and test projects. It represents a cross-disciplinary collection of well-known practices and methods.

Creating a Ranorex snapshot

A **Ranorex Snapshot** is a file representation of the user interface (UI) structure of an application under test (AUT) at a particular point in time. A Ranorex Snapshot captures all interface elements, their hierarchy, values, etc.

It can be saved as a single file. A Ranorex Snapshot file can be created and viewed with Ranorex Spy. The file extension is `.rxsnp`.

Typically, Ranorex Snapshots are used to share detailed information about the UI of an application with the Ranorex support and technical sales team.

Further reading

Creating a Ranorex snapshot is part of the Ranorex Spy knowledge. Therefore, it is introduced and explained in > Ranorex Studio advanced > Ranorex Spy > ⇢ Snapshot files.

Available screencasts:

You may also want to have a look at the available screencasts to this important topic.

Screencast 'Create a snapshot file'

See how to save (parts) of a repository to a snapshot file (`.rxsnp`) for communication with the Ranorex support team. Simply click here: Create a snapshot.

Screencast 'Create a snapshot of a context menu (approach 1 of 2) '

See how to create a snapshot of hidden UI-elements such as menu items, drop-down list items, combo box items, and others by means of instant tracking. Simply click here: Create a snapshot of a context menu (1 of 2).

Screencast 'Create a snapshot of a context menu (approach 2 of 2) '

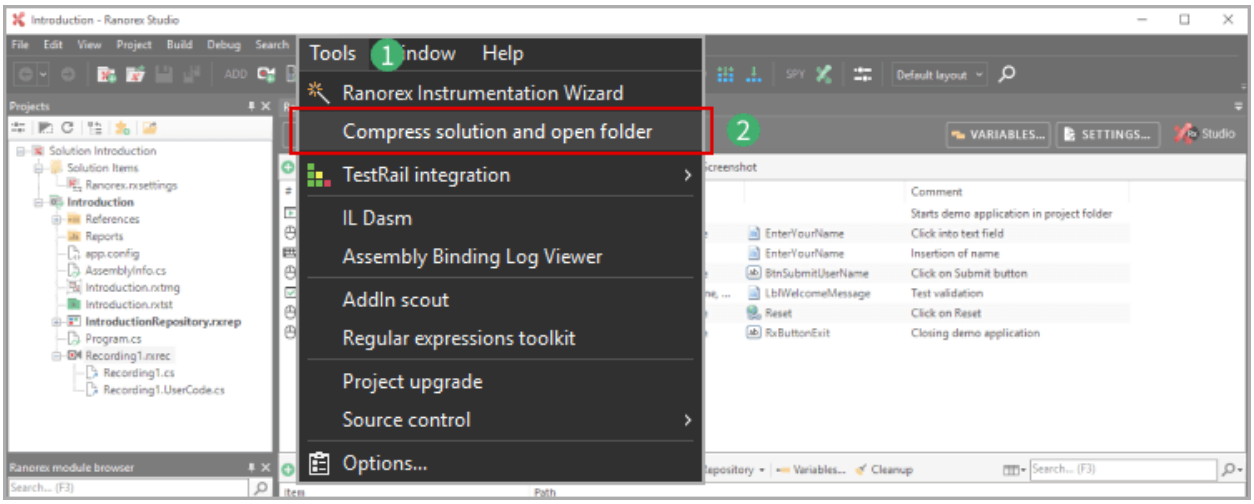
See how to create a snapshot of hidden UI-elements such as menu items, drop-down list items, combo box items, and others by means of the track function. Simply click here: Create a snapshot of a context menu (2 of 2).

Creating a compressed Ranorex solution

A compressed Ranorex solution is a file archive including all relevant files of a solution. This solution is created by a built-in compression function which works similar to any other compression software.

Create a compressed Ranorex solution

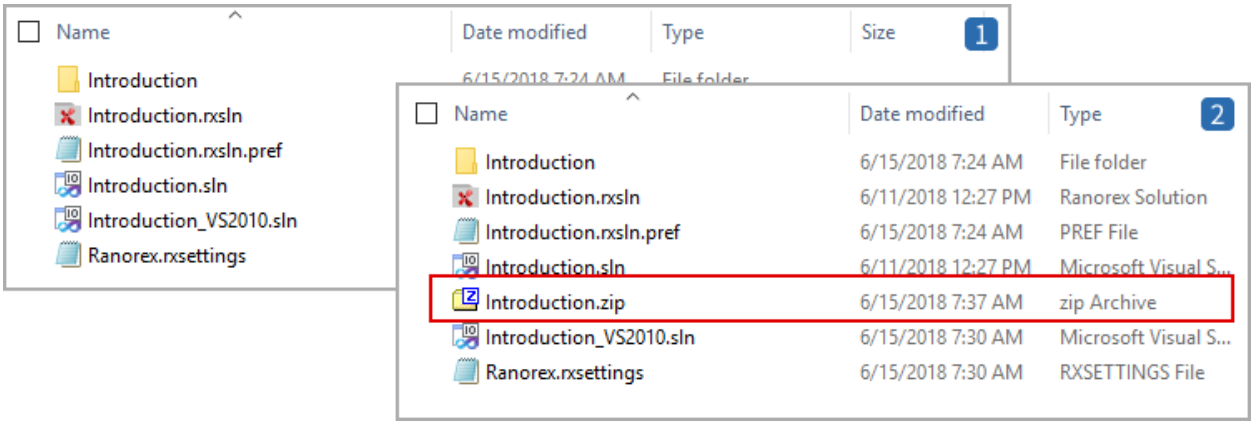
Creating a compressed solution is simple – follow the below instruction.



1. Loaded a solution, open the **Tools menu**
2. Click **Compress solution and open folder**

Result(s)

The compressed file will be created in the solution folder.



1. Project file folder **before** the creation of a compressed Ranorex solution
2. Project file folder **with the compressed Ranorex solution** (`Introduction.zip`)

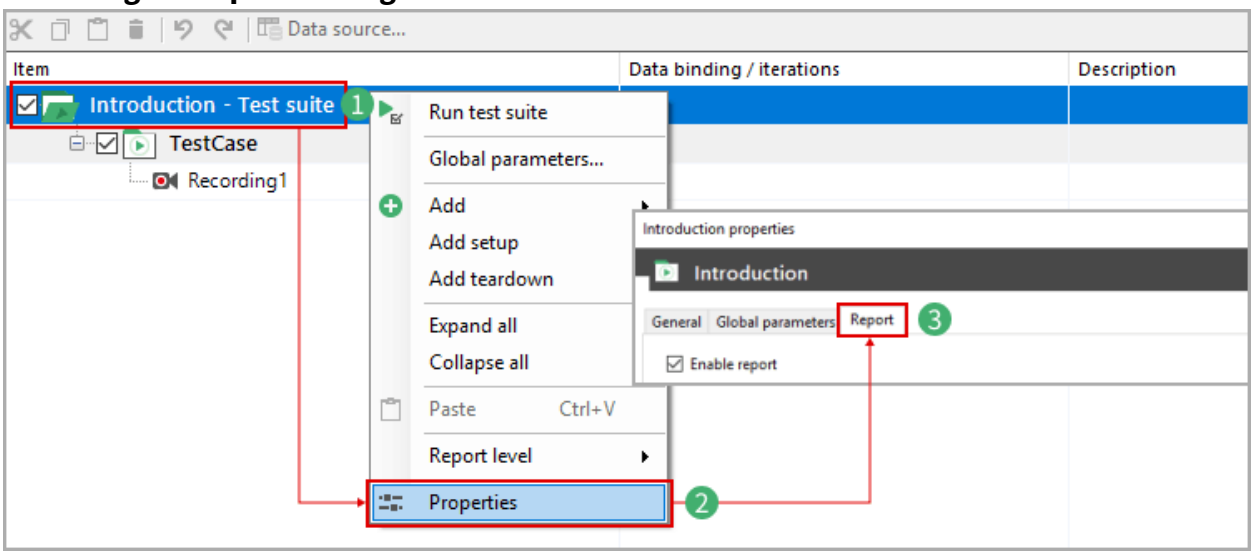
Creating a compressed Ranorex report

A compressed Ranorex Report is one zipped file including all files that belong to a single report. These are the report file itself (`*.rxlog`), all images and further relevant files. The file extension is '`.rxzlog`'.

Automatically create compressed Ranorex report

Ranorex can be configured to create a compressed copy of every report which is created. This is done in the report settings of the settings & configuration area. Here is how to.

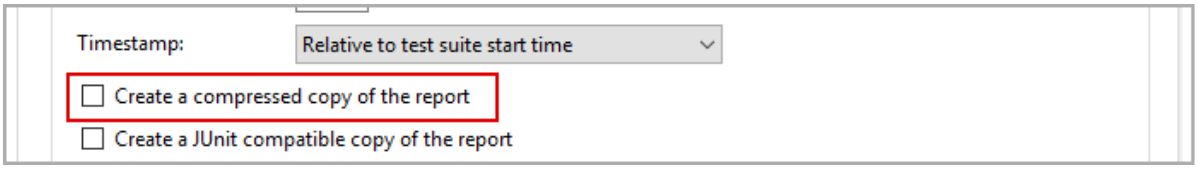
Accessing the report settings



1. Select the **test suite** and open the context menu
2. Click **Properties** in the context menu
3. In the settings window, click the **Report** register tab

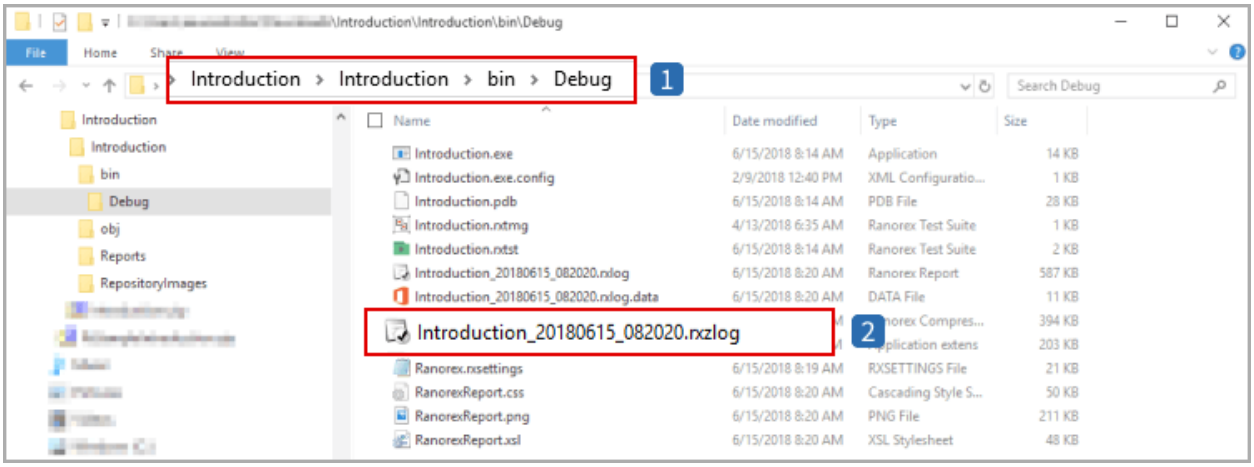
Enabling compressed report copy

Check the corresponding checkbox in the settings dialog to enable automatic creation of a compressed report copy.



Result(s):

The compressed Ranorex report file is saved in the output folder of the project file structure.



1 ***Destination folder**:*

- The output folder of the solution is `binDebug`
- In this folder, the compressed report copy is saved to

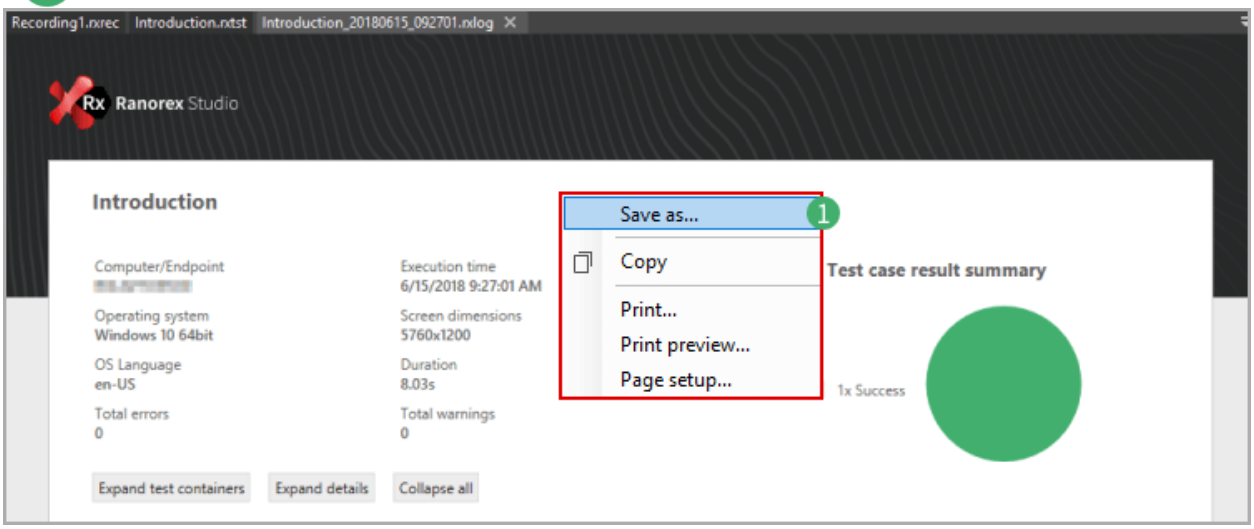
2 ***Compressed report copy***

- See the compressed report copy with the file ending '`.rxzlog`'
- The naming conventions are the same as for regular reports

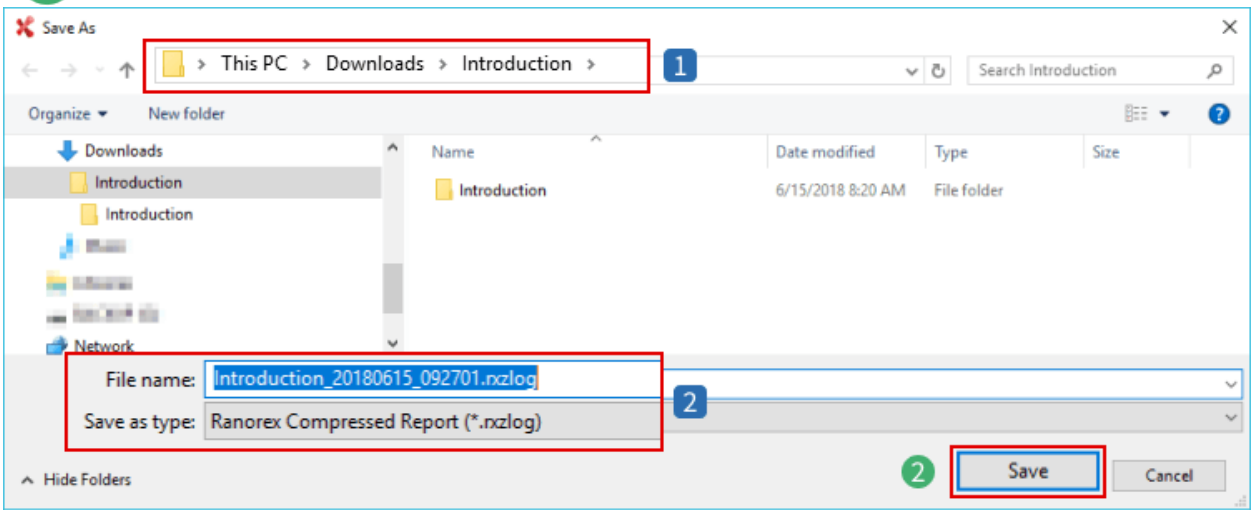
Manually create compressed Ranorex report

Any created report can be saved as compressed Ranorex report. This is how to do.

1 Open the **context menu** in a created report in Ranorex Studio and click **Save as…**



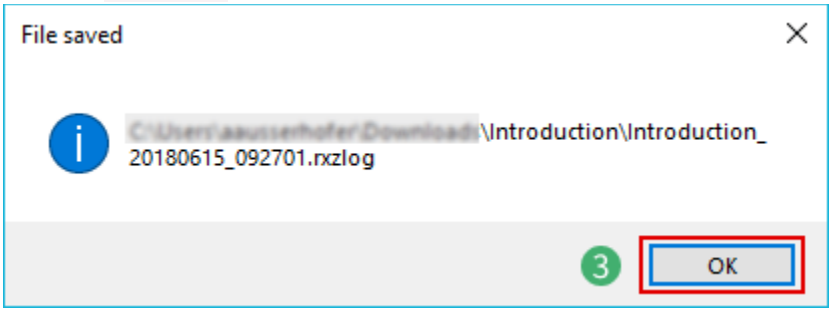
2 Select a destination folder, optionally change file name and click **Save**



Important to know:

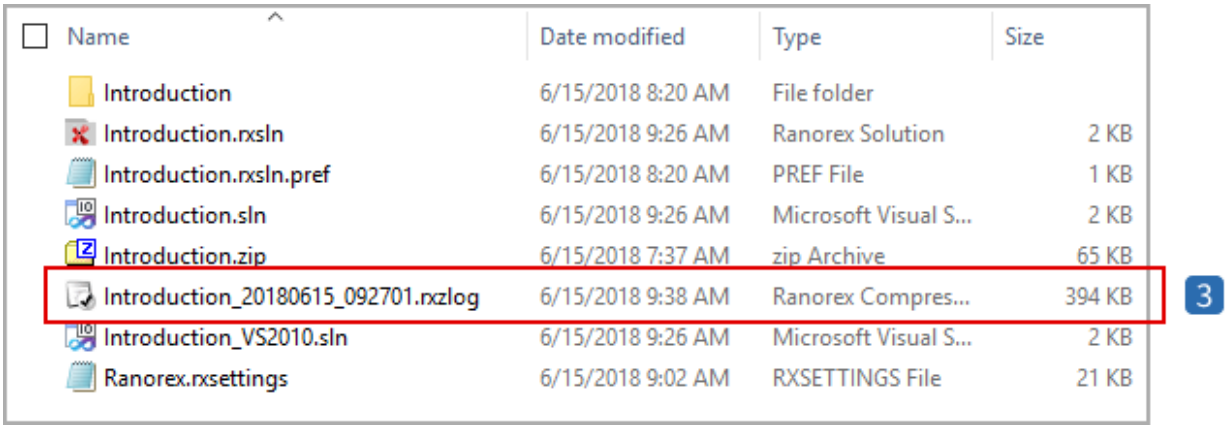
1 See the **default destination folder** which is the current root solution folder

2 See the **default file name** for the compressed report and the according file type `.rxzlog`



3 Click **OK** in the information window confirming the creation of the compressed report

Result(s):



3 See the **compressed report file** in the designated destination folder

Add a solution settings file to a solution

Ranorex distinguishes between user settings and solution settings. While user settings are saved with the Ranorex software, solution settings are stored in a corresponding settings file with the solution.

Usually, a solution settings file is automatically created with every solution. In some situations, it will be necessary to replace this settings file, or – if it is not available – to simply add a solution settings file to a solution.

Further reading

The concept of user settings and solution settings and the method of adding a solution settings file to a solution are introduced and explained in detail in > Ranorex Studio system details > Settings & configuration > ⇢ Introduction.

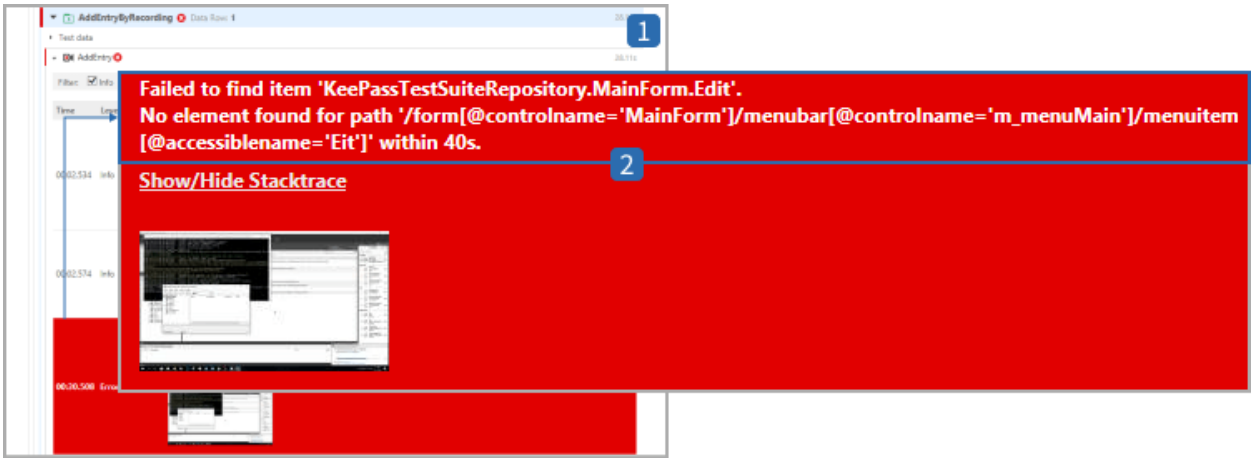
Fix ‘element not found’ error

The “***element not found***” error (i.e. **ENF**) is one of the most frequent errors that can cause a test to fail. Unfortunately, it’s often not easy to find out what caused the error, especially if you’re not that experienced in Ranorex Studio yet.

This is where this guide comes in. It will help you identify the most common causes of the error and resolve them so your test will execute successfully.

Analyze report

When you encounter the ENF error, report analysis is the first thing to do.



1 See the **report** identifying a **test failure**

2 **ENF error** stating that a UI-element could not be found

> **Note**
>
> ENF errors are often rooted in the individual characteristics of an application under test. This means that this guide can only speak in very general terms about solving such errors.

Screenshot inspection:

The report provides several screenshots to make it easier for you to identify what went wrong. These screenshots show what happened in the last seconds up to the error.

Compare the screenshots and pay special attention to the **UI element you wanted to find.** Select corresponding to your analysis:

1. Proceed to **Step 1** if the UI-element is **NOT in the screenshot**
2. Proceed to **Step 2** if the UI-element is **IN the screenshot**

Step 1 – UI-element is not in screenshot

Solution approach:

Make sure the application under test is in a correct state.

Summary:

If the UI-element isn't there, it means that your application under test (AUT) wasn't in the correct state for the UI-element to be found within the search timeout. This can have various causes. For example, a button might not appear until a certain action is completed and this action is taking longer because the system is working slower than usual. This will cause the UI-element to be there during some runs, and not during others.

Solution:

Ensure the application under test always reaches the correct state for Ranorex Studio to find the UI-element:

- **Check** that all the actions needed to reach this state are there. For example, if a UI-element is only reachable through a context menu, make sure there is an action that opens this context menu beforehand.
- **Add** validations to verify that the window, page, etc. that contains the UI-element appears during test execution.
- **Increase** the search timeout on the repository item or **add** a WaitFor action in the action table to compensate for slowdowns of the application under test and other possible delays.

Further reading

The concept of test validation is introduced and explained in > Ranorex Studio fundamentals > Test validation > ⇢ Introduction.

The search timeout of repository items can be set in the repository properties in > Ranorex Studio fundamentals > Repository > ⇢ Managing repository items.

Hint

If the UI-element is still not found, proceed to Step 2.

Step 2 – UI-element is in screenshot

Solution approach:

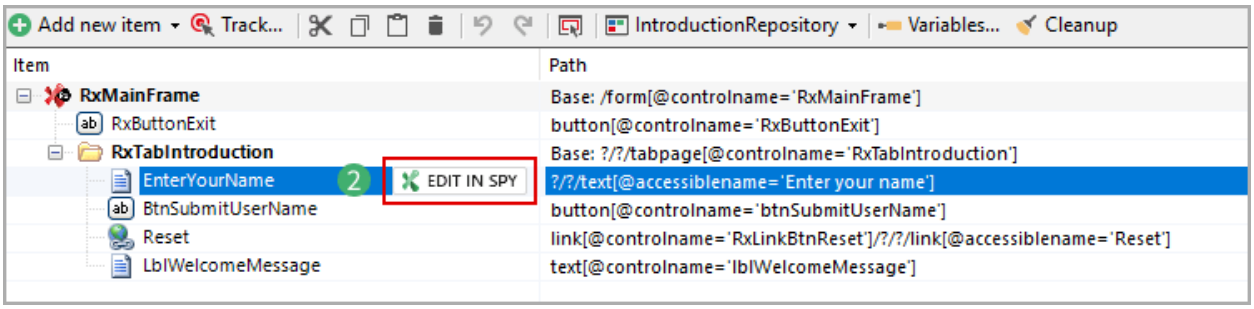
Check the RanoreXPath specification of the UI-element.

Summary:

If the UI-element is in the screenshot, it means that your application under test reached the correct state for the UI-element to be found within the timeout. However, Ranorex Studio still couldn't find the UI-element, so something else must be wrong. The most likely explanation is a **faulty RanoreXPath**. This can have various reasons, such as changes to the GUI, dynamic parts, multilingual GUIs, or even simple typos.

Open the UI-element in Spy

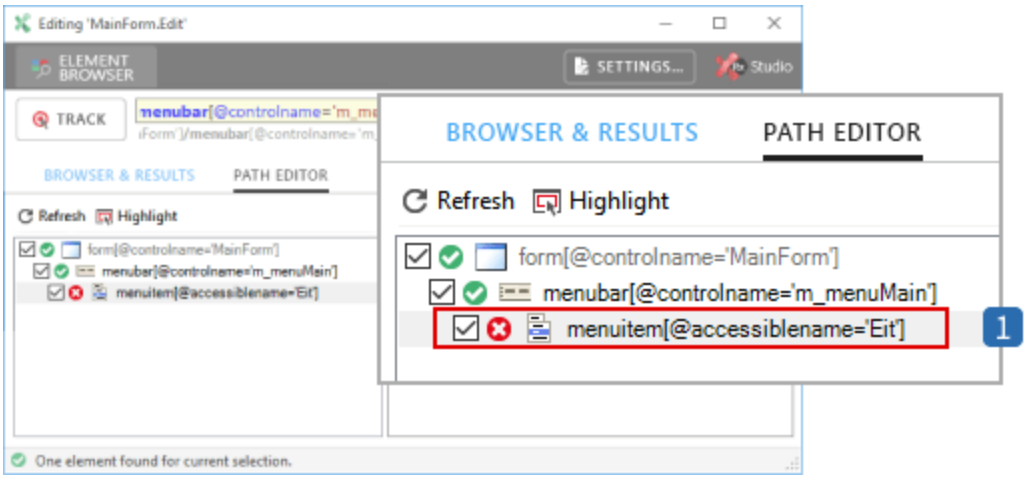
To be able to check and edit the validity of the RanoreXPath specification it is necessary to open the UI-element in Spy.



1. Ensure the **application under test** is in the same state when the ENF error occurred
2. Select the repository item and click **EDIT IN SPY** to open it in Ranorex Spy

Identify faulty RanoreXPath elements

First of all, the highest faulty level in the UI-element tree browser needs to be identified for a more detailed error analysis.



1 Identify the **highest faulty tree level**

Check for RanoreXPath syntax

Check the predicate of faulty the RanoreXPath specification for possible errors:

- Check attributes, values, and operators
- Look for typos, incomplete specifications and wrong values

> **Further reading**
>
> The advanced concept of RanoreXPath is introduced and explained in > Ranorex Studio advanced > RanoreXPath > ⇢ Introduction.

Check variables

- Does the RanoreXPath specification contain variables?
- Check if variables are correct and if they are bound to data sources
- Check if the referenced values are correct

> **Further reading**
>
> The concept of variables & parameter is introduced and explained in > Ranorex Studio advanced > Variables & parameter > ⇢ Introduction.

Re-track UI-element(s)

Perhaps the attributes and variables are correct, but the path structure itself is faulty, resulting in the UI-element not being found. A good way to solve this problem is by re-tracking the UI-element.

Further reading

The tracking of UI-elements is explained in > Ranorex Studio advanced > Tracking UI-elements > ⇢ Introduction.

Contact Ranorex support team

If you tried everything to solve the problem, but it still remains – your challenge seems more complex. Therefore, make a snapshot of your solution and contact the Ranorex support team by means of the contact form at `/support-query/`.

Further reading

If you need to know how to create a Ranorex snapshot file, refer to > Ranorex Studio advanced > Ranorex Spy > ⇢ Snapshot files.

Ranorex code examples

This chapter contains a collection of code examples which explain how to use the Ranorex API to write code modules and extend recording modules with user-specific code.

Using repository in code

C#

```
[TestModule("D451F1D1-C347-4B58-939F-F6187642EB56", ModuleType.UserCode, 1)]
public class UsingRepository : ITestModule
{
 // Repository object to access UI elements
 MyFirstTestProjectRepository repo = MyFirstTestProjectRepository.Instance;

 /// <summary>
 /// Constructs a new instance.
 /// </summary>
 public UsingRepository()
 {
 // Do not delete - a parameterless constructor is required!
 }

 void ITestModule.Run()
 {
 Mouse.DefaultMoveTime = 300;
 Keyboard.DefaultKeyPressTime = 100;
 Delay.SpeedFactor = 1.0;

 // Using Ranorex.Form adapter represented by 'MyApp'
 // 'MyApp' is used as a folder within the repository;
```

```
// the 'Self' property returns an object of type Ranorex.Form

// Activates application
repo.MyApp.Self.Activate();
// Log 'Active' state
Report.Info(repo.MyApp.Self.Active.ToString());
// Maximize, Minimize and Restore
repo.MyApp.Self.Maximize();
repo.MyApp.Self.Minimize();
repo.MyApp.Self.Restore();
// Closes application
repo.MyApp.Self.Close();



// Using Ranorex.Button adapter represented by 'ButtonAdd'
// Read and log value of 'Text' attribute
Report.Info(repo.MyApp.Buttons.ButtonAdd.Text);

// Press button
repo.MyApp.Buttons.ButtonAdd.Press();
// Read and log value of 'Enabled' attribute
Report.Info(repo.MyApp.Buttons.ButtonAdd.Enabled.ToString());

// Using Ranorex.RadioButton adapter
// represented by 'GenderOption'

// Select radio button
repo.MyApp.GenderOption.Select();
```

```
// Accessing listitems of Ranorex.List adapter
// represented by 'CategoryList'
IList<ranorex.listitem> listItems = repo.MyApp.CategoryList.Items;
foreach ( Ranorex.ListItem item in listItems )
{
Report.Info(item.Text + " is member of CategoryList");
}

// Using Ranorex.MenuItem to open 'File' menu
repo.MyApp.MenuItemFile.Select();
// Selecting sub menu item 'Connect'
repo.FileMenu.Connect.Select();
// Read and log 'Enabled' state of menu item 'Connect'
Report.Info(repo.FileMenu.Connect.Enabled.ToString());
}
}
```

VB.NET

```
Public Class UsingRepository
 Implements ITestModule
 ' Repository object to access UI elements
 Private repo As MyFirstTestProjectRepository = MyFirstTestProjectRepository.Inst
ance

 ''' <summary>
 ''' Constructs a new instance.
 ''' </summary>
 ' Do not delete - a parameterless constructor is required!
 Public Sub New()
```

```
End Sub


''' <summary>
''' Performs the playback of actions in this module.
''' </summary>
''' <remarks>You should not call this method directly, instead pass the module
''' instance to the <see cref="TestModuleRunner.Run(ITestModule)"> method
''' that will in turn invoke this method.</see></remarks>
Private Sub ITestModule_Run() Implements ITestModule.Run
Mouse.DefaultMoveTime = 300
Keyboard.DefaultKeyPressTime = 100
Delay.SpeedFactor = 1.0
' Using Ranorex.Form adapter represented by 'MyApp'
' 'MyApp' is used as a folder within the repository;
' the 'Self' property returns a Ranorex.Form object

' Activates application
repo.MyApp.Self.Activate()
' Log 'Active' state
Report.Info(repo.MyApp.Self.Active.ToString())
' Maximize, Minimize and Restore
repo.MyApp.Self.Maximize()
repo.MyApp.Self.Minimize()
repo.MyApp.Self.Restore()
' Closes application
repo.MyApp.Self.Close()



' Using Ranorex.Button adapter represented by ButtonAdd'
```

```
' Read and log value of 'Text' attribute
Report.Info(repo.MyApp.Buttons.ButtonAdd.Text)

' Press button
repo.MyApp.Buttons.ButtonAdd.Press()
' Read and log value of 'Enabled' attribute
Report.Info(repo.MyApp.Buttons.ButtonAdd.Enabled.ToString())

' Using Ranorex.RadioButton adapter
' represented by 'GenderOption'

' Select radio button
repo.MyApp.GenderOption.[Select]()
' Accessing listitems of Ranorex.List adapter
' represented by 'CategoryList'
Dim listItems As IList(Of Ranorex.ListItem) = repo.MyApp.CategoryList.Items
For Each item As Ranorex.ListItem In listItems
Report.Info(item.Text & " is member of CategoryList")
Next

' Using Ranorex.MenuItem to open 'File' menu
repo.MyApp.MenuItemFile.[Select]()
' Selecting sub menu item 'Connect'
repo.FileMenu.Connect.[Select]()
' Read and log 'Enabled' state of menu item 'Connect'
Report.Info(repo.FileMenu.Connect.Enabled.ToString())
End Sub
End Class
```

Wait for UI elements using repository

Each item and each folder type provides an additional object item declared with ‘Info’. It is used to access item related attributes without accessing the UI element directly in order to prevent Ranorex from throwing exceptions. The info object is mainly used to check whether an item or a folder path is valid or not. In combination with the timeout set for the item, you can use it to wait for UI elements like dialogs, text values and web content.

C#

```
// Use the 'Info' object to check existence of the
// 'SaveDialog' item; Method 'Exists' uses the timeout
// specified for the 'SaveDialog' in the repository
Report.Info("Exists = " + repo.SaveDialog.SelfInfo.Exists().ToString());

// Use the 'Info' object to check existence of the
// 'TextOnline' item which uses the following RXPath:
// statusbar/text[@accessiblename='Online']
// This way you can wait with the timeout specified for
// the item within the repository for the text 'Online'
bool statusTextConnected = repo.MyApp.TextOnlineInfo.Exists();

// Using 'Info' objects for validation
// Throws a Ranorex.ValidationException if validation
// fails. Automatically reports success or failed message
// to log file
Validate.Exists(repo.SaveDialog.ButtonOKInfo);

// Validates the existence of the repository item,
// but does not throw any exception
Validate.Exists(repo.SaveDialog.ButtonOKInfo,"Check Object '{0}'",false);
```

- VB.NET

```
' Use the 'Info' object to check existence of the
```

```
' 'SaveDialog' item; Method 'Exists' uses the timeout
' specified for the 'SaveDialog' in the repository
Report.Info("Exists = " & repo.SaveDialog.SelfInfo.Exists().ToString())

' Use the 'Info' object to check existence of the
' 'TextOnline' item which uses the following RXPath:
' statusbar/text[@accessiblename='Online']
' This way you can wait with the timeout specified for
' the item within the repository for the text 'Online'
Dim statusTextConnected As Boolean = repo.MyApp.TextOnlineInfo.Exists()

' Using 'Info' objects for validation
' Throws a Ranorex.ValidationException if validation
' fails. Automatically reports success or failed message
' to log file
Validate.Exists(repo.SaveDialog.ButtonOKInfo)

' Validates the existence of the repository item,
' but does not throw any exception
Validate.Exists(repo.SaveDialog.ButtonOKInfo, "Check Object '{0}'", False)
```

Create adapters to access more properties and methods

If you analyze the VIP Database application form with Ranorex Spy, you see that that the application window provides three adapters (Form, Control and NativeWindow). The Ranorex.Form adapter with all attributes and methods is directly available using the repository's application folder 'MyApp'. If you want to access properties like 'ProcessName' or invoke methods exposed by a .NET WinForms control you need to convert the Form adapter to NativeWindow or Control. As you can see in the code section below the '…Info' object is used to create the desired adapter.

Note

In order to access properties or methods exposed by a WinForms control you need to know their names. If you're not familiar with the control's API ask the developer of your application for assistance.

C#

```
// Creating adapter of type 'NativeWindow' using the "...Info" object

Ranorex.NativeWindow nativeWnd = repo.MyApp.SelfInfo.CreateAdapter<Ranorex.NativeWindow>(false);

// ... and read value of attribute 'ProcessName'

Report.Info("Process name of VIP Database: " + nativeWnd.ProcessName);


// Using Control Adapter to access properties and methods of

// .NET WinForms control

Ranorex.Control winFormsControl = repo.MyApp.SelfInfo.CreateAdapter<Ranorex.Control>(false);

// Set background color of VIP application to Color.Black using the

// exposed property 'BackColor'

winFormsControl.SetPropertyValue("BackColor",Color.Black);


// Report screenshot after changing the background color

Report.Screenshot(repo.MyApp.Self);

// Closes VIP Database by invoking the 'Close' method

// exposed by the System.Windows.Forms.Form class

winFormsControl.InvokeMethod("Close");
```

- VB.NET

```
' Creating adapter of type 'NativeWindow' using the "...Info" object

Dim nativeWnd As Ranorex.NativeWindow = repo.MyApp.SelfInfo.CreateAdapter(Of Ranorex.NativeWindow)(False)
```

```
' ... and read value of attribute 'ProcessName'
Report.Info("Process name of VIP Database: " & nativeWnd.ProcessName)

' Using Control Adapter to access properties and methods of
' .NET WinForms control
Dim winFormsControl As Ranorex.Control = repo.MyApp.SelfInfo.CreateAdapter(Of Ranorex.Control)(False)
' Set background color of VIP application to Color.Black using the
' exposed property 'BackColor'
winFormsControl.SetPropertyValue("BackColor", Color.Black)

' Report screenshot after changing the background color
Report.Screenshot(repo.MyApp.Self)
' Closes VIP Database by invoking the 'Close' method
' exposed by the System.Windows.Forms.Form class
winFormsControl.InvokeMethod("Close")
```

Create a list of adapters from a repository element

If multiple elements match a RanoreXPath of a single repository item use the CreateAdapters method to create a list of adapters. Learn more about how to create repository items delivering multiple elements here: ⇢ Adding Repository Items

C#

```
// Create a list of adapters using the "Info" object
IList<Ranorex.Button> buttonList =
 repo.MyApp.EnabledButtonsInfo.CreateAdapters<Ranorex.Button>();
```

```
// Move the mouse pointer to each button of the list
// and add a screenshot for each to the report
foreach (Ranorex.Button button in buttonList )
{
 button.MoveTo();
 Report.Screenshot(button);
}
```

VB.NET

```
' Create a list of adapters using the "Info" object
Dim buttonList As IList(Of Ranorex.Button) =
 repo.MyApp.EnabledButtonsInfo.CreateAdapters(Of Ranorex.Button)()

' Move the mouse pointer to each button of the list
' and add a screenshot for each to the report
For Each button As Ranorex.Button In buttonList
 button.MoveTo()
 Report.Screenshot(button)
Next
```

Using the Validate class

The Ranorex.Validate class is used to check values from UI elements, but it can also be used to simply compare non UI related objects in code. In comparison to a simple IF statement the methods provided automatically log fail or success messages to the report.

Note

Each method provided by the Validate class allows to suppress exceptions thrown in case of a failed validation. The code snippet above uses only a few validation methods. For further and more detailed explanation of the Ranorex.Validate class see the API documentation.

C#

```
// Validate for Existence


// Using 'Info' objects for validation
// Throws a Ranorex.ValidationException if validation
// fails. Automatically reports success or failed message
// to log file
Validate.Exists(repo.SaveDialog.ButtonOKInfo);


// Validates the existence of the repository item,
// but does not throw any exception
bool exists = Validate.Exists(repo.SaveDialog.ButtonOKInfo,"Check Object '{0}'",f
alse);


// Check whether an application form exists using a RanoreXPath
Validate.Exists("/form[@controlname='formVipApplication']");


// Check whether an application does not exists
// for the time specified as timeout for the given repository item
Validate.NotExists(repo.MyApp.SelfInfo);
 // Validate 'Enabled' attribute of button 'Delete'
Validate.Attribute(repo.MyApp.Buttons.ButtonDeleteInfo,"Enabled",false);
```

VB.NET

```
' Validate for Existence


' Using 'Info' objects for validation
' Throws a Ranorex.ValidationException if validation
' fails. Automatically reports success or failed message
' to log file
Validate.Exists(repo.SaveDialog.ButtonOKInfo)


' Validates the existence of the repository item,
' but does not throw any exception
Dim exists As Boolean = Validate.Exists(repo.SaveDialog.ButtonOKInfo,"Check Objec
t '{0}'",false)


' Check whether an application form exists using a RanoreXPath
Validate.Exists("/form[@controlname='formVipApplication']")


' Check whether an application does not exists
' for the time specified as timeout for the given repository item
Validate.NotExists(repo.MyApp.SelfInfo)


' Validate 'Enabled' attribute of button 'Delete'
Validate.Attribute(repo.MyApp.Buttons.ButtonDeleteInfo,"Enabled",false)
```

Forcing a test case to fail using the Validate class

As described above you can use the Validate class to compare values from UI and non UI related objects bringing a test to fail or success. Additional to this, you can force a test case to fail without comparing using the Validate class.

C#

```
Ranorex.Cell cellObject = null;
// Try to find a cell object
bool found=false;
found = Host.Local.TryFindSingle<Ranorex.Cell>("/form//table/row/cell[3]", 2000,
out cellObject);
// If the expressions does not return an object
// call Validate.Fail and the test case fails
if (!found) Validate.Fail("RanoreXPath with no return");
```

VB.NET

```
Dim cellObject As Ranorex.Cell = Nothing
' Try to find a cell object
Dim found As Boolean = False
found = Host.Local.TryFindSingle(Of Ranorex.Cell)("/form//table/row/cell[3]", 200
0, cellObject)
' If the expressions does not return an object
' call Validate.Fail and the test case fails
If Not found Then
 Validate.Fail("RanoreXPath with no return")
End If
```

Forcing a test case to fail using exceptions

Ranorex uses exception handling to determine whether a test case run failed or succeeded. As long as no exception is thrown by any of the Ranorex methods (e.g Ranorex.Validate method or use of not valid repository item) the test run will be successful.

If you want to prevent Ranorex from throwing exceptions but at the same time decide on your own whether a test case fails or not, you need to throw Ranorex exceptions programmatically.

C#

```
Ranorex.Cell cellObject = null;

// Try to find a cell object using two

// different RanoreXPath expressions

bool found=false;

found = Host.Local.TryFindSingle<Ranorex.Cell>("/form//table/row/cell[3]", 2000,
out cellObject);

found = found || Host.Local.TryFindSingle<Ranorex.cell>("/form//table/row/cell[4]
", 2000, out cellObject);

// If none of the expressions returns an object

// throw new 'ElementNotFoundException' and test case fails

if (!found)

{

 throw new Ranorex.ElementNotFoundException("Both RanoreXPath with no return", nu
ll);

}

else

{

 // If the value of attribute 'Text' does not equal to the expected value

 // throw new 'ValidationException' to break the test case

 if ( cellObject.Text == "MyExpectedTextValue" )

 {

 Report.Success("User Specific Validation","Text validation of cell object succee
ded");

 }

 else

 {

 throw new Ranorex.ValidationException("Text validation of cell object succeeded
failed");

 }

}
```

```
</ranorex.cell></ranorex.cell>
```

VB.NET

```
Dim cellObject As Ranorex.Cell = Nothing
' Try to find a cell object using two
' different RanoreXPath expressions
Dim found As Boolean = Host.Local.TryFindSingle(Of Ranorex.Cell)("/form//table/row/cell[3]", 2000, cellObject)
found = found OrElse Host.Local.TryFindSingle(Of Ranorex.Cell)("/form//table/row/cell[4]", 2000, cellObject)
' If none of the expressions returns an object
' throw new 'ElementNotFoundException' and test case fails
If Not found Then
 Throw New Ranorex.ElementNotFoundException("Both RanoreXPath with no return", Nothing)
Else
 ' If the value of attribute 'Text' does not equal to the expected value
 ' throw new 'ValidationException' to break the test case
 If cellObject.Text = "MyExpectedTextValue" Then
 Report.Success("User Specific Validation", "Text validation of cell object succeeded")
 Else
 Throw New Ranorex.ValidationException("Text validation of cell object succeeded failed")
 End If
End If
```

Set up automation speed

You can optionally specify and change the automation speed at any time in the code. The code generated by a recording uses the same properties to define replay speed as used within

a code module. A newly created code module already contains three lines of code specifying the automation speed in the ‘Run’ method.

C#

```
void ITestModule.Run()
{
 Mouse.DefaultMoveTime = 300;
 Keyboard.DefaultKeyPressTime = 100;
 Delay.SpeedFactor = 1.0;
}
```

VB.NET

```
Private Sub ITestModule_Run() Implements ITestModule.Run
 Mouse.DefaultMoveTime = 300
 Keyboard.DefaultKeyPressTime = 100
 Delay.SpeedFactor = 1.0
End Sub
```

Accessing test case and test suite context

Sometimes it’s necessary to forward values read from the UI in a code module or recording to the module executed next in the scope of a test case. The example code shown below uses the module variables varDialogTextA (Code Module A) and varDialogTextB (Code Module B) which are both bound to a parameter specified within the test case’s data binding dialog to transfer data within a test case.

C#

```
// ----------------- Code Block used by Code Module A -----------------

// Click 'Save' button to open 'SaveDialog'
repo.MyApp.Buttons.ButtonSave.Click();
// Read text message shown with 'SaveDialog'
```

```
// and assign it to the variable 'varDialogTextA' bound to a test case parameter
varDialogTextA = repo.SaveDialog.TextMessage.TextValue;
// -------- Code Block used by User Code Action of recording B --------
// Read value of module variable 'varDialogTextB' in other code module
// or recording module using a user code action
Report.Info(varDialogTextB);
// Get the current data context and log
// the current row index of a data driven run
Report.Info(TestCase.Current.DataContext.CurrentRowIndex.ToString());
```

VB.NET

```
' ----------------- Code Block used by Code Module A -----------------

' Click 'Save' button to open 'SaveDialog'
repo.MyApp.Buttons.ButtonSave.Click()
' Read text message shown with 'SaveDialog'
' and assign it to the variable 'varDialogTextA' bound to a test case parameter
varDialogTextA = repo.SaveDialog.TextMessage.TextValue

' -------- Code Block used by User Code Action of recording B --------

' Read value of module variable 'varDialogTextB' in other code module
' or recording module using a user code action
Report.Info(varDialogTextB)

' Get the current data context and log
' the current row index of a data driven run
```

```
Report.Info(TestCase.Current.DataContext.CurrentRowIndex.ToString())
```

Advanced code examples

You can also use RanoreXPath expressions directly in code in order to search for items using different 'Find' methods offered by the API. Start searching for elements directly at the root level using 'Host.Local' or reuse existing adapters like repository items to start searching from there.

C#

```
// Create Ranorex.Button adapter using 'FindSingle' method
// from Host.Local (starting at root node) with absolute RanoreXPath
// Note: ElementNotFound exception is thrown if item is not found within
// the specified timeout of 2000 ms.
Ranorex.Button addButtonVar1 = Host.Local.FindSingle<Ranorex.Button>("/form[@controlname='formVipApplication']/button[@controlname='btAdd']",2000);
addButtonVar1.MoveTo();

// Alternatively you can use the 'TryFindSingle' method to prevent
// a test case from failing because of an exception thrown if
// the element is not found within the specified timeout of 2000 ms
Ranorex.Button addButtonVar2 = null;
bool found = Host.Local.TryFindSingle<Ranorex.Button>("/form[@controlname='formVipApplication']/button[@controlname='btAdd']", 2000, out addButtonVar2);
// Move mouse pointer to button
addButtonVar2.MoveTo();

// Request a list of buttons shown from the VIP Database application
// and create a screenshot for each button in the report file
IList<Ranorex.Button> buttonList = Host.Local.Find<Ranorex.Button>("/form[@controlname='formVipApplication']/button",500);
foreach (Ranorex.Button bt in buttonList)
```

```
{
 Report.Screenshot(bt);
}


// Using find methods in combination with existing Ranorex repository items
Ranorex.Button btAdd = repo.MyApp.Self.FindSingle<Ranorex.Button>("button[@contro
lname='btAdd']",2000);
```

VB.NET

```
' Create Ranorex.Button adapter using 'FindSingle' method
' from Host.Local (starting at root node) with absolute RanoreXPath
' Note: ElementNotFound exception is thrown if item is not found within
' the specified timeout of 2000 ms.
Dim addButtonVar1 As Ranorex.Button = Host.Local.FindSingle(Of Ranorex.Button)("/
form[@controlname='formVipApplication']/button[@controlname='btAdd']", 2000)
addButtonVar1.MoveTo()


' Alternatively you can use 'TryFindSingle' method to prevent
' a test case from failing because of an exception thrown if
' the element is not found within the specified timeout of 2000 ms
Dim addButtonVar2 As Ranorex.Button = Nothing
Dim found As Boolean = Host.Local.TryFindSingle(Of Ranorex.Button)("/form[@contro
lname='formVipApplication']/button[@controlname='btAdd']", 2000, addButtonVar2)
' Move mouse pointer to button
addButtonVar2.MoveTo()


' Request a list of buttons from the VIP Database application
' and create a screenshot for each button in the report file
Dim buttonList As IList(Of Ranorex.Button) = Host.Local.Find(Of Ranorex.Button)("
/form[@controlname='formVipApplication']/button", 500)
```

```
For Each bt As Ranorex.Button In buttonList
 Report.Screenshot(bt)
Next

' Using find methods in combination with existing Ranorex repository items
Dim btAdd As Ranorex.Button = repo.MyApp.Self.FindSingle(Of Ranorex.Button)("butt
on[@controlname='btAdd']", 2000)
```

How to do image-based automation

If Ranorex is not able to clearly identify some of your GUI elements, it may be helpful to automate them using the implicit image search mechanism of the 'Click' method.

C#

```
// Create bitmap to search for
// within application form and
// click it
Bitmap bmp = Ranorex.Imaging.Load(
 @"....Green Sea Turtle Small.bmp");
// Performs a right click on the image found
// within the application window based on
// the image location
myRepo.WinFormsApp.Self.Click(MouseButtons.Right,bmp);

// You can also search for images that are slightly different to the
// loaded image by specifying the minimum Similarity for a match (95% = 0.95).
myRepo.WinFormsApp.Self.Click(new Location(bmp, new Imaging.FindOptions(0.95)));

// OR Set the default Similarity value for all following image operations
Imaging.FindOptions.Default.Similarity = 0.95;
```

```
myRepo.WinFormsApp.Self.Click(bmp);


Report.Success("Image found and clicked successfully");
```

VB.NET

```
' Create bitmap to search for
' within application form and
' click it
Dim bmp As Bitmap = Ranorex.Imaging.Load("....Green Sea Turtle Small.bmp")
' Performs a right click on the image found
' within the application window based
' on the image location
myRepo.WinFormsApp.Self.Click(MouseButtons.Right, bmp)

' You can also search for images that are slightly different to the
' loaded image by specifying the minimum Similarity for a match (95% = 0.95).
myRepo.WinFormsApp.Self.Click(new Location(bmp, new Imaging.FindOptions(0.95)))

' OR Set the default Similarity value for all following image operations
Imaging.FindOptions.Default.Similarity = 0.95
myRepo.WinFormsApp.Self.Click(bmp)

Report.Success("Image displayed successfully")
```

How to find and compare images

To compare an image simply search for it within an existing Ranorex element using the ‘Contains’ method.

> **Hint**
>
> Both examples load an uncompressed file (BMP or PNG format) in order to carry out a one-to-one comparison. Use the FindOptions class to configure similarity, preprocessing and other search settings.

C#

```
// Create bitmap
Bitmap bmp = Ranorex.Imaging.Load(
 @"....Green Sea Turtle Small.bmp");

// Search for it within the application window
if (Ranorex.Imaging.Contains(myRepo.WinFormsApp.Self,bmp) == true)
{
 Report.Success("Image found within WinForms application");
}
```

VB.NET

```
' Create bitmap
Dim bmp As Bitmap = Ranorex.Imaging.Load("....Green Sea Turtle Small.bmp")
' Search for it within the application window
If Imaging.Contains(myRepo.WinFormsApp.Self,bmp Then
 Report.Success("Image found within WinForms application")
End If
```

Handling unexpected dialogs

A common problem in UI testing is the appearance of an unexpected dialog – e.g. the Update-Check dialog in KeePass.

To overcome this issue, you can use the PopupWatcher class. Using this class you can add watches for each dialog which might pop up during test execution. In these watches you can

specify a RanoreXPath or a Repository item the PopupWatcher should keep watching for, as well as a method which should be triggered or a repository item which should be clicked.

C#

```
void ITestModule.Run()
{

 // Create PopupWatcher
 PopupWatcher myPopupWatcher = new PopupWatcher();

 // Add a Watch using a RanoreXPath and triggering the Method CloseUpdateCheckDialog
 myPopupWatcher.Watch("/form[@controlname='UpdateCheckForm']/button[@controlname='m_btnClose']", CloseUpdateCheckDialog);

 // Add a Watch using the info object of a button and triggering the Method CloseUpdateCheckDialog
 // myPopupWatcher.Watch(repo.UpdateCheckDialog.btCloseInfo, CloseUpdateCheckDialog);

 // Add a Watch using the info object of the dialog and the info object of the button to click
 // myPopupWatcher.WatchAndClick(repo.UpdateCheckDialog.SelfInfo, repo.UpdateCheckDialog.btCloseInfo);

 // Add a Watch using a repository folder object and the info object of the button to click
 // myPopupWatcher.WatchAndClick(repo.UpdateCheckDialog, repo.UpdateCheckDialog.btCloseInfo);

 // Start PopupWatcher
 myPopupWatcher.Start();
```

```
}


public static void CloseUpdateCheckDialog(Ranorex.Core.Repository.RepoItemInfo myInfo, Ranorex.Core.Element myElement)
{
 myElement.As<Ranorex.Button>().Click();
}


public static void CloseUpdateCheckDialog(Ranorex.Core.RxPath myPath, Ranorex.Core.Element myElement)
{
 myElement.As<Ranorex.Button>().Click();
}
```

VB.NET

```
Private Sub ITestModule_Run() Implements ITestModule.Run


 ' Create PopupWatcher
 Dim myPopupWatcher As New PopupWatcher()


 ' Add a Watch using a RanoreXPath and triggering the Method CloseUpdateCheckDialog
 myPopupWatcher.Watch("/form[@controlname='UpdateCheckForm']/button[@controlname='m_btnClose']", AddressOf CloseUpdateCheckDialog)


 ' Add a Watch using the info object of a button and triggering the Method CloseUpdateCheckDialog
 ' myPopupWatcher.Watch(repo.UpdateCheckDialog.btCloseInfo, CloseUpdateCheckDialog);
```

```
 ' Add a Watch using the info object of the dialog and the info object of the but
ton to click

 ' myPopupWatcher.WatchAndClick(repo.UpdateCheckDialog.SelfInfo, repo.UpdateCheck
Dialog.btCloseInfo);


 ' Add a Watch using a repository folder object and the info object of the button
 to click

 ' myPopupWatcher.WatchAndClick(repo.UpdateCheckDialog, repo.UpdateCheckDialog.bt
CloseInfo);


 ' Start PopupWatcher

 myPopupWatcher.Start()


End Sub


Public Shared Sub CloseUpdateCheckDialog(myInfo As Ranorex.Core.Repository.RepoIt
emInfo, myElement As Ranorex.Core.Element)

 myElement.[As](Of Ranorex.Button)().Click()

End Sub


Public Shared Sub CloseUpdateCheckDialog(myPath As Ranorex.Core.RxPath, myElement
 As Ranorex.Core.Element)

 myElement.[As](Of Ranorex.Button)().Click()

End Sub
```

Creating and using a WebDriver (incl. configuration) in code

C#

```
// Create endpoint management factory
```

```
var fac = new RemoteEndpointFactory();


// Use existing endpoints
var existing = fac.GetAll();
Report.Log(ReportLevel.Info, string.Format("Endpoints: {0}", existing.Count()));


foreach (var e in existing)
{
 Report.Log(ReportLevel.Info, string.Format("Name: {0}", e.DisplayName));
}


// Get WebDriver endpoints form the existing ones (no equivalent for mobile curre
ntly)
var webDriverEndpoints = fac.GetAllWebDriverEndpoints();
Report.Log(ReportLevel.Info, string.Format("There are '{0}' WebDriver endpoints",
 webDriverEndpoints.Count()));


// Create user process endpoint
var ep = fac.CreateTransientWebDriverEndpoint(new WebDriverEndpointInfo("tempEp",
 "http://localhost:4444/wd/hub"));


var cfg = WebDriverConfiguration.FromJson("{}");
cfg.Name = "MyConfig1";
cfg.Description = "Code sample Config";


ep.ActivateConfiguration(cfg);


ep.ConnectAsync()
 .ContinueWith(_ => ep.MakeCurrentHostAsync())
 .Wait();
```

```
var b = ep.StartBrowser("firefox", "http://www.ranorex.com");
```

Advanced validation – clipboard

This sample shows how the current content of the clipboard can be validated against a given reference value.

C#

```
public void Validate_ClipboardContentEqual(string compareText, string customLogMe
ssage)
{
 // check if content of clipboard is text
 if (System.Windows.Forms.Clipboard.ContainsText())
 {
 // Get text from clipboard
 string clipboardtext = System.Windows.Forms.Clipboard.GetText();


 // prepare log message if the log-parameter is empty
 if (string.IsNullOrEmpty(clipboardtext))
 {
 customLogMessage = "Comparing content of clipboard with given text: ({0} vs. {1}
)"; ;
 }


 // Validate if given text (compareText) is equal to current text in clipboard
 Ranorex.Validate.AreEqual(clipboardtext, compareText, customLogMessage);
 }
 else
 {
 throw new Ranorex.RanorexException("There is not text on the clipboard that can
be compared");
```

```
 }
}
```

VB.NET

```
Public Sub Validate_ClipboardContentEqual(compareText As String, customLogMessage
 As String)
 ' check if content of clipboard is text
 If System.Windows.Forms.Clipboard.ContainsText() Then
 ' Get text from clipboard
 Dim clipboardtext As String = System.Windows.Forms.Clipboard.GetText()


 ' prepare log message if the log-parameter is empty
 If String.IsNullOrEmpty(clipboardtext) Then
 customLogMessage = "Comparing content of clipboard with given text: ({0} vs. {1}
)"



 End If


 ' Validate if given text (compareText) is equal to current text in clipboard
 Ranorex.Validate.AreEqual(clipboardtext, compareText, customLogMessage)
 Else
 Throw New Ranorex.RanorexException("There is not text on the clipboard that can
be compared")
 End If
End Sub
```

Advanced validation – entire table

This sample shows how the content of a whole table (UI control) can be validated at once. Therefore, the table has to be passed as a parameter. Additionally, the filename of a Ranorex Snapshot needs to be provided which will act as a reference and defines the expected content of the table.

> **Note**
>
> The referencing snapshot has to be created before calling this advanced validation function. Simply select the table in the tree view in Ranorex Spy, right-click and choose “Save as Snapshot” in the context menu. More information on creating snapshots can be found here: ⇢ Creating Ranorex Snapshots.

C#

```
public void Validate_TableContentEqual(Ranorex.Adapter repoItem, string filename_
ReferenceTableSnapshot, string customLogMessageOverall, string customLogMessageDe
tail)
{


 // check if snapshot file exists
 const string fileNotExists = "The given file does not exist: {0}";
 if (!System.IO.File.Exists(filename_ReferenceTableSnapshot))
 {
 throw new Ranorex.ValidationException(string.Format(fileNotExists, filename_Refe
renceTableSnapshot));
 }


 ElementSnapshot snap = null;
 try
 {
 snap = Ranorex.Core.ElementSnapshot.CreateFromFile (filename_ReferenceTableSnaps
hot); // ElementSnapshot.CreateFromFile is available starting with Ranorex 5.4.2
```

```
 }
 catch
 {
 throw new Ranorex.ValidationException("Snapshot could not be loaded from file");

 }

 // restore table from snapshot
 Ranorex.Table refTable;
 try
 {
 refTable = snap.Element;
 }
 catch
 {
 throw new Ranorex.ValidationException("Table could not be created from snapshot"
);
 }
 var tableAdapter = repoItem.As <Ranorex.Table>();
 if (tableAdapter==null)
 {
 throw new Ranorex.ValidationException("Repo-item could not be accessed");
 }

 // check if rowcount is identical
 if (tableAdapter.Rows.Count != refTable.Rows.Count)
 {
 throw new Ranorex.ValidationException(String.Format ("Tables do not have same nu
mber of rows ({0} vs. {1})", tableAdapter.Rows.Count, refTable.Rows.Count));
 }
```

```
// run through table-rows
for (int iRow = 0; iRow <= tableAdapter.Rows.Count - 1; iRow++)
{
int cellCountCur = tableAdapter.Rows[iRow].Cells.Count;
int cellCountRef = refTable.Rows[iRow].Cells.Count;

// check if number of cells is identical in current row
if (cellCountCur != cellCountRef)
{
throw new Ranorex.ValidationException(String.Format("Table-
Rows do not have same number of cells ({0} vs. {1})", cellCountCur, cellCountRef)
);
}

// run through cells in current row
for (int iCol = 0; iCol <= cellCountCur - 1; iCol++)
{
string aCurText = tableAdapter.Rows[iRow].Cells[iCol].As<Ranorex.Cell>().Text;
string aRefText = refTable.Rows[iRow].Cells[iCol].As<Ranorex.Cell>().Text;

string validationMessage = string.Empty;
if (string.IsNullOrEmpty(customLogMessageDetail))
{
validationMessage = String.Format ("Comparing content of cell ({2}/{3}) (found:'
{0}', expected: '{1}')", aCurText, aRefText, iRow,iCol);
}
else
{
validationMessage = customLogMessageDetail;
```

```
 }


 // validate whether current text and expected text are identical

 Ranorex.Validate.AreEqual (aCurText, aRefText, validationMessage);

 }

 }

 // Log overall success

 if (string.IsNullOrEmpty(customLogMessageOverall))

 customLogMessageOverall = "Successfully completed content-
validation of table with provided snapshot of table (reference)";

 Ranorex.Report.Log (ReportLevel.Success, customLogMessageOverall);

}
```

VB.NET

```
public Sub Validate_TableContentEqual(repoItem As Ranorex.Adapter, filename_Refer
enceTableSnapshot As String, customLogMessageOverall As String, customLogMessageD
etail As String)


 ' check if snapshot file exists

 Const fileNotExists As String = "The given file does not exist: {0}"

 If Not System.IO.File.Exists(filename_ReferenceTableSnapshot) Then

 Throw New Ranorex.ValidationException(String.Format(fileNotExists, filename_Refe
renceTableSnapshot))

 End If


 Dim snap As ElementSnapshot = Nothing

 Try

 snap = Ranorex.Core.ElementSnapshot.CreateFromFile(filename_ReferenceTableSnapsh
ot) 'ElementSnapshot.CreateFromFile is available starting with Ranorex 5.4.2

 Catch
```

```
 Throw New Ranorex.ValidationException("Snapshot could not be loaded from file")
 End Try


 ' restore table from snapshot
 Dim refTable As Ranorex.Table
 Try
 refTable = snap.Element
 Catch
 Throw New Ranorex.ValidationException("Table could not be created from snapshot"
)
 End Try
 Dim tableAdapter = repoItem.[As](Of Ranorex.Table)()
 If tableAdapter Is Nothing Then
 Throw New Ranorex.ValidationException("Repo-item could not be accessed")
 End If


 ' check if rowcount is identical
 If tableAdapter.Rows.Count <> refTable.Rows.Count Then
 Throw New Ranorex.ValidationException([String].Format("Tables do not have same n
umber of rows ({0} vs. {1})", tableAdapter.Rows.Count, refTable.Rows.Count))
 End If


 ' run through table-rows
 For iRow As Integer = 0 To tableAdapter.Rows.Count - 1
 Dim cellCountCur As Integer = tableAdapter.Rows(iRow).Cells.Count
 Dim cellCountRef As Integer = refTable.Rows(iRow).Cells.Count


 ' check if number of cells is identical in current row
 If cellCountCur <> cellCountRef Then
```

```
 Throw New Ranorex.ValidationException([String].Format("Table-
Rows do not have same number of cells ({0} vs. {1})", cellCountCur, cellCountRef)
)
 End If


 ' run through cells in current row
 For iCol As Integer = 0 To cellCountCur - 1
 Dim aCurText As String = tableAdapter.Rows(iRow).Cells(iCol).[As](Of Ranorex.Cel
l)().Text
 Dim aRefText As String = refTable.Rows(iRow).Cells(iCol).[As](Of Ranorex.Cell)()
.Text


 Dim validationMessage As String = String.Empty
 If String.IsNullOrEmpty(customLogMessageDetail) Then
 validationMessage = [String].Format("Comparing content of cell ({2}/{3}) (found:
'{0}', expected: '{1}')", aCurText, aRefText, iRow, iCol)
 Else
 validationMessage = customLogMessageDetail
 End If


 ' validate whether current text and expected text are identical
 Ranorex.Validate.AreEqual(aCurText, aRefText, validationMessage)
 Next
 Next
 ' Log overall success
 If String.IsNullOrEmpty(customLogMessageOverall) Then
 customLogMessageOverall = "Successfully completed content-
validation of table with provided snapshot of table (reference)"
 End If
 Ranorex.Report.Log(ReportLevel.Success, customLogMessageOverall)
End Sub
```

Advanced validation – entire table in web

This sample shows how the content of a whole web table (HTML tag table) can be validated at once. Therefore, the table has to be passed as a parameter. Additionally, the filename of a Ranorex Snapshot needs to be provided which will act as a reference and defines the expected content of the web table.

> **Note**
>
> The referencing snapshot has to be created before calling this advanced validation function. Simply select the table in the tree view in Ranorex Spy, right-click and choose “Save as Snapshot” in the context menu. More information on creating snapshots can be found here: ⇢ Creating Ranorex Snapshots.

C#

```
public void Validate_WebTableContentEqual(Ranorex.Adapter repoItem, string filena
me_ReferenceTableSnapshot, string customLogMessageOverall, string customLogMessag
eDetail)
{
 // check if snapshot file exists
 Ranorex.Validate.IsTrue (System.IO.File.Exists (filename_ReferenceTableSnapshot)
, string.Format("Checking existence of snapshot file: {0}",filename_ReferenceTabl
eSnapshot ));

 ElementSnapshot snap;
 try
 {
 snap = Ranorex.Core.ElementSnapshot.CreateFromFile (filename_ReferenceTableSnaps
hot); // ElementSnapshot.CreateFromFile is available starting with Ranorex 5.4.2
 }
 catch
```

```
 {
 throw new Ranorex.ValidationException ("Snapshot could not be loaded from file")
;
 }


 // restore table from snapshot
 Ranorex.TableTag refTable;
 try
 {
 refTable = snap.Element;
 }
 catch (Exception e)
 {
 throw new Ranorex.ValidationException("Table could not be created from snapshot.
", e);
 }


 Ranorex.TableTag tableAdapter = repoItem.As <Ranorex.TableTag>();
 if (tableAdapter==null)
 {
 throw new Ranorex.ValidationException("Repo-item could not be accessed.");
 }


 IList<TrTag> actTableRows = tableAdapter.FindDescendants<TrTag>();
 int rowCntAct = actTableRows.Count;
 IList<TrTag> refTableRows = refTable.FindDescendants<TrTag>();
 int rowCntRef = refTableRows.Count;


 // check if rowcount is identical
```

```
 Ranorex.Validate.AreEqual (rowCntAct, rowCntRef, "Comparing number of rows ({0} vs. {1})");


 // run through table-rows

 for (int iRow = 0; iRow <= actTableRows.Count - 1; iRow++)

 {

 IList<Ranorex.WebElement> cellsInActRow = actTableRows[iRow].FindChildren<Ranorex.WebElement>();

 IList<Ranorex.WebElement> cellsInRefRow = refTableRows[iRow].FindChildren<Ranorex.WebElement>();


 // check if number of cells is identical in current row

 Ranorex.Validate.AreEqual (cellsInActRow.Count, cellsInRefRow.Count, "Comparing number of cells in current row ({0} vs. {1})");


 // run through cells in current row

 for (int iCol = 0; iCol <= cellsInActRow.Count - 1; iCol++)

 {

 string aCurText = new WebElement(cellsInActRow[iCol]).InnerText;

 string aRefText = new WebElement(cellsInRefRow[iCol]).InnerText;


 string validationMessage = string.Empty;

 if (string.IsNullOrEmpty(customLogMessageDetail))

 {

 validationMessage = String.Format ("Comparing content of cell ({2}/{3}) (found:'{0}', expected: '{1}')", aCurText, aRefText, iRow,iCol);

 }

 else

 {

 validationMessage = customLogMessageDetail;

 }
```

```
// validate whether current text and expected text are identical
Ranorex.Validate.AreEqual (aCurText, aRefText, validationMessage);
}
}

// Log overall success
if (string.IsNullOrEmpty(customLogMessageOverall))
customLogMessageOverall = "Successfully completed content-validation of web-table with provided snapshot of web-table (reference)";
Ranorex.Report.Log (ReportLevel.Success, customLogMessageOverall);
}
```

VB.NET

```
Public Sub Validate_WebTableContentEqual(repoItem As Ranorex.Adapter, filename_ReferenceTableSnapshot As String, customLogMessageOverall As String, customLogMessageDetail As String)
' check if snapshot file exists
Ranorex.Validate.IsTrue(System.IO.File.Exists(filename_ReferenceTableSnapshot), String.Format("Checking existence of snapshot file: {0}", filename_ReferenceTableSnapshot))

Dim snap As ElementSnapshot
Try
' ElementSnapshot.CreateFromFile is available starting with Ranorex 5.4.2
snap = Ranorex.Core.ElementSnapshot.CreateFromFile(filename_ReferenceTableSnapshot)
Catch
Throw New Ranorex.ValidationException("Snapshot could not be loaded from file")
End Try
```

```
 ' restore table from snapshot

 Dim refTable As Ranorex.TableTag

 Try

 refTable = snap.Element

 Catch e As Exception

 Throw New Ranorex.ValidationException("Table could not be created from snapshot.
", e)

 End Try


 Dim tableAdapter As Ranorex.TableTag = repoItem.[As](Of Ranorex.TableTag)()

 If tableAdapter Is Nothing Then

 Throw New Ranorex.ValidationException("Repo-item could not be accessed.")

 End If


 Dim actTableRows As IList(Of TrTag) = tableAdapter.FindDescendants(Of TrTag)()

 Dim rowCntAct As Integer = actTableRows.Count

 Dim refTableRows As IList(Of TrTag) = refTable.FindDescendants(Of TrTag)()

 Dim rowCntRef As Integer = refTableRows.Count


 ' check if rowcount is identical

 Ranorex.Validate.AreEqual(rowCntAct, rowCntRef, "Comparing number of rows ({0} v
s. {1})")


 ' run through table-rows

 For iRow As Integer = 0 To actTableRows.Count - 1

 Dim cellsInActRow As IList(Of Ranorex.WebElement) = actTableRows(iRow).FindChild
ren(Of Ranorex.WebElement)()

 Dim cellsInRefRow As IList(Of Ranorex.WebElement) = refTableRows(iRow).FindChild
ren(Of Ranorex.WebElement)()
```

```
 ' check if number of cells is identical in current row
 Ranorex.Validate.AreEqual(cellsInActRow.Count, cellsInRefRow.Count, "Comparing n
umber of cells in current row ({0} vs. {1})")


 ' run through cells in current row
 For iCol As Integer = 0 To cellsInActRow.Count - 1
 Dim aCurText As String = New WebElement(cellsInActRow(iCol)).InnerText
 Dim aRefText As String = New WebElement(cellsInRefRow(iCol)).InnerText


 Dim validationMessage As String = String.Empty
 If String.IsNullOrEmpty(customLogMessageDetail) Then
 validationMessage = [String].Format("Comparing content of cell ({2}/{3}) (found:
'{0}', expected: '{1}')", aCurText, aRefText, iRow, iCol)
 Else
 validationMessage = customLogMessageDetail
 End If


 ' validate whether current text and expected text are identical
 Ranorex.Validate.AreEqual(aCurText, aRefText, validationMessage)
 Next
 Next


 ' Log overall success
 If String.IsNullOrEmpty(customLogMessageOverall) Then
 customLogMessageOverall = "Successfully completed content-validation of web-
table with provided snapshot of web-table (reference)"
 End If
 Ranorex.Report.Log(ReportLevel.Success, customLogMessageOverall)
End Sub
```

Advanced validation – file (text-based)

This sample shows how the content of a text-based file can be validated. The content of the file will be compared with the content of a reference-file. Therefore, the filename of the file to validate needs to be provided as well as the filename of a reference-file.

C#

```
public void Validate_FileTextEqual(string filePath_Expected, string filePath_Current, string customLogMessage)
{
 // prepare log messages
 const string fileNotFoundMessage = "File not found for comparison in Validate_FileContentEqual: {0}";
 const string logMessage = "Comparing content of files ({0} vs. {1})";
 if (string.IsNullOrEmpty(customLogMessage))
 {
 customLogMessage = string.Format(logMessage, filePath_Expected, filePath_Current);
 }

 // check if file exists
 if (!System.IO.File.Exists(filePath_Current))
 {
 throw new Ranorex.RanorexException(string.Format(fileNotFoundMessage, filePath_Current));
 }

 // check if referencing file exists
 if (!System.IO.File.Exists(filePath_Expected))
 {
 throw new Ranorex.RanorexException(string.Format(fileNotFoundMessage, filePath_Expected));
```

```
 }


 // check if filenames are identical
 if (filePath_Expected.Equals(filePath_Current))
 {
 Ranorex.Validate.IsTrue(true, customLogMessage);
 }
 else
 {
 string current = System.IO.File.ReadAllText(filePath_Current);
 string expected = System.IO.File.ReadAllText(filePath_Expected);
 // validate whether expected value equals to current value
 Ranorex.Validate.AreEqual(current, expected, customLogMessage);
 }
}
```

VB.NET

```
Public Sub Validate_FileTextEqual(filePath_Expected As String, filePath_Current A
s String, customLogMessage As String)
 ' prepare log messages
 Const fileNotFoundMessage As String = "File not found for comparison in Validate
_FileContentEqual: {0}"
 Const logMessage As String = "Comparing content of files ({0} vs. {1})"
 If String.IsNullOrEmpty(customLogMessage) Then
 customLogMessage = String.Format(logMessage, filePath_Expected, filePath_Current
)
 End If
```

```
 ' check if file exists
 If Not System.IO.File.Exists(filePath_Current) Then
 Throw New Ranorex.RanorexException(String.Format(fileNotFoundMessage, filePath_C
urrent))
 End If


 ' check if referencing file exists
 If Not System.IO.File.Exists(filePath_Expected) Then
 Throw New Ranorex.RanorexException(String.Format(fileNotFoundMessage, filePath_E
xpected))
 End If


 ' check if filenames are identical
 If filePath_Expected.Equals(filePath_Current) Then
 Ranorex.Validate.IsTrue(True, customLogMessage)
 Else
 Dim current As String = System.IO.File.ReadAllText(filePath_Current)
 Dim expected As String = System.IO.File.ReadAllText(filePath_Expected)
 ' validate whether expected value equals to current value
 Ranorex.Validate.AreEqual(current, expected, customLogMessage)
 End If
End Sub
Advanced validation - file
```

Advanced validation – file (not-text-based, binary)

This sample shows how a non text-based file (binary file) can be validated. The content of the file will be compared with the content of a reference-file. Therefore the filename of the file to validate needs to be provided as well as the filename of a reference-file.

C#

```
public void Validate_FileBinaryContentEqual(string filePath_Expected, string file
Path_Current, string customLogMessage)
{
 // prepare log messages
 const string fileNotFoundMessage = "File not found for comparison in Validate_Fi
leContentEqual: {0}";
 const string logMessage = "Comparing content of files ({0} vs. {1})";
 if (string.IsNullOrEmpty(customLogMessage))
 {
 customLogMessage = string.Format(logMessage, filePath_Expected, filePath_Current
);
 }

 // check if file exists
 if (!System.IO.File.Exists(filePath_Current))
 {
 throw new Ranorex.ValidationException (string.Format(fileNotFoundMessage, filePa
th_Current));
 }

 // check if referencing file exists
 if (!System.IO.File.Exists(filePath_Expected))
 {
 throw new Ranorex.ValidationException(string.Format(fileNotFoundMessage, filePat
h_Expected));
 }

 // check if filenames are identical
```

```
 if (filePath_Expected.Equals(filePath_Current))
 {
 Ranorex.Validate.IsTrue(true, customLogMessage);
 }
 else
 {
 using (var file1 = new System.IO.FileStream(filePath_Current, System.IO.FileMode
.Open))
 using (var file2 = new System.IO.FileStream(filePath_Expected, System.IO.FileMod
e.Open))
 // Check whether files are equal
 Ranorex.Validate.IsTrue(StreamEquals(file1, file2), customLogMessage);
 }
}


// compare file-streams
private static bool StreamEquals(System.IO.Stream stream1, System.IO.Stream strea
m2)
{
 const int bufferSize = 2048;
 byte[] buffer1 = new byte[bufferSize];
 byte[] buffer2 = new byte[bufferSize];
 while (true)
 {
 int count1 = stream1.Read(buffer1, 0, bufferSize);
 int count2 = stream2.Read(buffer2, 0, bufferSize);

 if (count1 != count2)
 return false;
```

```
 if (count1 == 0)

 return true;


 for (int i = 0; i < count1; i++)

 {

 if (buffer1[i] != buffer2[i])

 {

 return false;

 }

 }

 }

}
```

VB.NET

```
Public Sub Validate_FileBinaryContentEqual(filePath_Expected As String, filePath_
Current As String, customLogMessage As String)

 ' prepare log messages

 Const fileNotFoundMessage As String = "File not found for comparison in Validate
_FileContentEqual: {0}"

 Const logMessage As String = "Comparing content of files ({0} vs. {1})"

 If String.IsNullOrEmpty(customLogMessage) Then

 customLogMessage = String.Format(logMessage, filePath_Expected, filePath_Current
)

 End If


 ' check if file exists

 If Not System.IO.File.Exists(filePath_Current) Then

 Throw New Ranorex.ValidationException(String.Format(fileNotFoundMessage, filePat
h_Current))

 End If
```

```
 ' check if referencing file exists
 If Not System.IO.File.Exists(filePath_Expected) Then
 Throw New Ranorex.ValidationException(String.Format(fileNotFoundMessage, filePath_Expected))
 End If



 ' check if filenames are identical
 If filePath_Expected.Equals(filePath_Current) Then
 Ranorex.Validate.IsTrue(True, customLogMessage)
 Else
 Using file1 = New System.IO.FileStream(filePath_Current, System.IO.FileMode.Open)
 Using file2 = New System.IO.FileStream(filePath_Expected, System.IO.FileMode.Open)
 ' Check whether files are equal
 Ranorex.Validate.IsTrue(StreamEquals(file1, file2), customLogMessage)
 End Using
 End Using
 End If
End Sub


' compare file-streams
Private Shared Function StreamEquals(stream1 As System.IO.Stream, stream2 As System.IO.Stream) As Boolean
 Const bufferSize As Integer = 2048
 Dim buffer1 As Byte() = New Byte(bufferSize - 1) {}
 Dim buffer2 As Byte() = New Byte(bufferSize - 1) {}
 While True
```

```
 Dim count1 As Integer = stream1.Read(buffer1, 0, bufferSize)
 Dim count2 As Integer = stream2.Read(buffer2, 0, bufferSize)


 If count1 <> count2 Then
 Return False
 End If


 If count1 = 0 Then
 Return True
 End If


 For i As Integer = 0 To count1 - 1
 If buffer1(i) <> buffer2(i) Then
 Return False
 End If
 Next
 End While
End Function
```

Advanced validation – database (single field)

This sample shows how a database-field can be validated against a given text value. To access the database, a valid connection string needs to be provided as well as a valid SQL Statement. Please see the MSDN for more information on the OLE database connection.

> **Note**
>
> For performance reasons, do not use this method in a series. If you want to access an entire table and/or SQL results, use the method after this one.

C#

```
public void Validate_DatabaseFieldWithQuery(string OLEConnectionString, string SQ
LQuery, string expectedValue, string customLogMessage)
{

 // check is connection string is empty
 if (string.IsNullOrEmpty(OLEConnectionString))
 {
 throw new Ranorex.RanorexException("ConnectionString is empty");
 }

 // check if SQL statement is empty
 if (SQLQuery.Trim().Equals(string.Empty))
 {
 throw new Ranorex.RanorexException("SQLQuery is empty");
 }

 // establish connection to database
 using (System.Data.OleDb.OleDbConnection connection = new System.Data.OleDb.OleD
bConnection(@OLEConnectionString))
 {
 connection.Open();

 System.Data.OleDb.OleDbCommand command = null;
 System.Data.OleDb.OleDbDataReader SQLReader = null;
 try
 {
 // set SQL statement and execute search
 command = new System.Data.OleDb.OleDbCommand(SQLQuery, connection);
```

```
SQLReader = command.ExecuteReader();


SQLReader.Read();
 // check if there is a result
if (SQLReader.FieldCount > 0)
{
 // retrieve single result from SQL database
var actualValue = SQLReader.GetString(0);
// prepare log message
if (customLogMessage.Trim().Equals(string.Empty))
{
customLogMessage = "Actual value = '{0}', expected value = '{1}' (database query
= " + SQLQuery + ")";
}
 // compare retrieved value with expected value
Ranorex.Validate.AreEqual(actualValue, expectedValue, customLogMessage);
}
else
{
throw new Ranorex.RanorexException(string.Format("SQL statement did not return a
ny results: {0}", SQLQuery));
}
}
finally
{
command.Dispose();
SQLReader.Dispose();
}
}
}
```

VB.NET

```
Public Sub Validate_DatabaseFieldWithQuery(OLEConnectionString As String, SQLQuer
y As String, expectedValue As String, customLogMessage As String)

' check is connection string is empty
If String.IsNullOrEmpty(OLEConnectionString) Then
Throw New Ranorex.RanorexException("ConnectionString is empty")
End If

' check if SQL statement is empty
If SQLQuery.Trim().Equals(String.Empty) Then
Throw New Ranorex.RanorexException("SQLQuery is empty")
End If

' establish connection to database
Using connection As New System.Data.OleDb.OleDbConnection(OLEConnectionString)
connection.Open()

Dim command As System.Data.OleDb.OleDbCommand = Nothing
Dim SQLReader As System.Data.OleDb.OleDbDataReader = Nothing
Try
' set SQL statement and execute search
command = New System.Data.OleDb.OleDbCommand(SQLQuery, connection)
SQLReader = command.ExecuteReader()

SQLReader.Read()

' check if there is a result
If SQLReader.FieldCount > 0 Then
```

```
' retrieve single result from SQL database
Dim actualValue = SQLReader.GetString(0)

' prepare log message
If customLogMessage.Trim().Equals(String.Empty) Then
customLogMessage = "Actual value = '{0}', expected value = '{1}' (database query
= " & SQLQuery & ")"
End If

' compare retrieved value with expected value
Ranorex.Validate.AreEqual(actualValue, expectedValue, customLogMessage)
Else
Throw New Ranorex.RanorexException(String.Format("SQL statement did not return a
ny results: {0}", SQLQuery))
End If
Finally
command.Dispose()
SQLReader.Dispose()
End Try
End Using
End Sub
```

Advanced validation – database (entire table)

This sample shows how the content of a database can be validated. Using it you can check against an entire table, a database view or against every result of an SQL statement. Thus a valid SQL statement as well as a valid connection string has to be provided to allow access to the database. Please see the MSDN for more information on establishing an OLE database connection.

Additionally, the filename of a csv-file needs to be passed in as a parameter. This file is acts as a reference and should contain the expected values. If the values in the file are separated differently than using the semicolon (“;”), please use the parameter “csvSeparator” to pass it in.

> **Note**
>
> The reference file (csv-file) needs to be created in forefront of calling this method. Every text editor can be used for creating this reference-file.

C#

```
 public void Validate_DatabaseTableWithQuery(string OLEConnectionString, string S
QLQuery, string referenceCSVTable, string csvSeparator, string customLogMessageOv
erall, string customLogMessageDetail)
{


 // check if reference file exists
 if (!System.IO.File.Exists(referenceCSVTable))
 {
 throw new Ranorex.RanorexException(string.Format("File does not exist: {0}", ref
erenceCSVTable));
 }


 // check if connection string is empty
 if (OLEConnectionString.Trim().Equals(string.Empty))
 {
 throw new Ranorex.RanorexException("ConnectionString is empty");
 }


 // check if SQL statement is empty
 if (SQLQuery.Trim().Equals(string.Empty))
```

```
{
throw new Ranorex.RanorexException("SQLQuery is empty");
}

// prepare separator for csv file (if not passed in)
char separator;
if (string.IsNullOrEmpty(csvSeparator))
{
separator = ';';
}
else
{
separator = csvSeparator[0];
}

// establish database connection
using (System.Data.OleDb.OleDbConnection connection = new System.Data.OleDb.OleDbConnection(@OLEConnectionString))
{
connection.Open();

System.Data.OleDb.OleDbCommand command = null;
System.Data.OleDb.OleDbDataReader SQLReader = null;
System.IO.StreamReader CSVReader = null;

try
{
// set SQL statement and execute search
command = new System.Data.OleDb.OleDbCommand(SQLQuery, connection);
```

```
 SQLReader = command.ExecuteReader();


 // open csv file (reference file)
 CSVReader = new System.IO.StreamReader(System.IO.File.OpenRead(@referenceCSVTabl
e));


 // run through every line in file
 int iRow = 0;
 while ((SQLReader.Read()) && (!CSVReader.EndOfStream))
 {


 // read line from csv file
 var line = CSVReader.ReadLine();


 // split line into array of values
 var values = line.Split(separator);


 // check if number of values equals (in current row of csv file and in SQL resul
t)
 if (values.Length != SQLReader.FieldCount)
 {
 throw new Ranorex.RanorexException(string.Format("Number of fields in SQL query
and reference-
file does not match ({0} vs. {1})", values.Length + 1, SQLReader.FieldCount));
 }



 // run through every field in SQL result
 for (int iFields = 0; iFields <= SQLReader.FieldCount - 1; iFields++)
 {
```

```
 var expectedValue = values[iFields];
 var actualValue = SQLReader[iFields].ToString();

 // prepare log message
 string validationMessage = string.Empty;
 if (string.IsNullOrEmpty(customLogMessageDetail))
 {
 validationMessage = String.Format("Comparing content of cell ({2}/{3}) (found:'{
0}', expected: '{1}')", actualValue, expectedValue, iRow, iFields);
 }
 else
 {
 validationMessage = customLogMessageDetail;
 }
 // validate if actual value and expected value are equal
 Ranorex.Validate.AreEqual(actualValue, expectedValue, customLogMessageDetail);

 }
 iRow++;
 }
 if (string.IsNullOrEmpty(customLogMessageOverall))
 {
 customLogMessageOverall = "Successfully validated SQL Table with given SQL-
Statement against content of given CSV file";
 }
 // Log success
 Ranorex.Report.Log(Ranorex.ReportLevel.Success, customLogMessageOverall);

 }
```

```
 finally
 {
 command.Dispose();
 SQLReader.Dispose();
 CSVReader.Dispose();
 }
 }
}
```

VB.NET

```
Public Sub Validate_DatabaseTableWithQuery(OLEConnectionString As String, SQLQuer
y As String, referenceCSVTable As String, csvSeparator As String, customLogMessag
eOverall As String, customLogMessageDetail As String)


 ' check if reference file exists
 If Not System.IO.File.Exists(referenceCSVTable) Then
 Throw New Ranorex.RanorexException(String.Format("File does not exist: {0}", ref
erenceCSVTable))
 End If


 ' check if connection string is empty
 If OLEConnectionString.Trim().Equals(String.Empty) Then
 Throw New Ranorex.RanorexException("ConnectionString is empty")
 End If


 ' check if SQL statement is empty
 If SQLQuery.Trim().Equals(String.Empty) Then
 Throw New Ranorex.RanorexException("SQLQuery is empty")
 End If
```

```
' prepare separator for csv file (if not passed in)
Dim separator As Char
If String.IsNullOrEmpty(csvSeparator) Then
separator = ";"C
Else
separator = csvSeparator(0)
End If

' establish database connection
Using connection As New System.Data.OleDb.OleDbConnection(OLEConnectionString)
connection.Open()

Dim command As System.Data.OleDb.OleDbCommand = Nothing
Dim SQLReader As System.Data.OleDb.OleDbDataReader = Nothing
Dim CSVReader As System.IO.StreamReader = Nothing

Try
' set SQL statement and execute search
command = New System.Data.OleDb.OleDbCommand(SQLQuery, connection)
SQLReader = command.ExecuteReader()

' open csv file (reference file)
CSVReader = New System.IO.StreamReader(System.IO.File.OpenRead(referenceCSVTable
))

' run through every line in file
Dim iRow As Integer = 0
While (SQLReader.Read()) AndAlso (Not CSVReader.EndOfStream)
```

```
 ' read line from csv file

 Dim line = CSVReader.ReadLine()


 ' split line into array of values

 Dim values = line.Split(separator)


 ' check if number of values equals (in current row of csv file and in SQL result
)

 If values.Length <> SQLReader.FieldCount Then

 Throw New Ranorex.RanorexException(String.Format("Number of fields in SQL query
and reference-
file does not match ({0} vs. {1})", values.Length + 1, SQLReader.FieldCount))

 End If




 ' run through every field in SQL result

 For iFields As Integer = 0 To SQLReader.FieldCount - 1


 Dim expectedValue = values(iFields)

 Dim actualValue = SQLReader(iFields).ToString()


 ' prepare log message

 Dim validationMessage As String = String.Empty

 If String.IsNullOrEmpty(customLogMessageDetail) Then

 validationMessage = [String].Format("Comparing content of cell ({2}/{3}) (found:
'{0}', expected: '{1}')", actualValue, expectedValue, iRow, iFields)

 Else

 validationMessage = customLogMessageDetail

 End If

 ' validate if actual value and expected value are equal
```

```
 Ranorex.Validate.AreEqual(actualValue, expectedValue, customLogMessageDetail)
 Next
 iRow += 1
 End While
 If String.IsNullOrEmpty(customLogMessageOverall) Then
 customLogMessageOverall = "Successfully validated SQL Table with given SQL-
Statement against content of given CSV file"
 End If
 ' Log success

 Ranorex.Report.Log(Ranorex.ReportLevel.Success, customLogMessageOverall)
 Finally
 command.Dispose()
 SQLReader.Dispose()
 CSVReader.Dispose()
 End Try
 End Using
End Sub
```

Advanced validation – XML code

This sample shows how XML code can be validated with Ranorex. The method below awaits the XML code snippet as a parameter and an XPath expression in the parameter 'node'. This XPath expression selects the node to validate. Please provide the expected value as well in the parameter 'expectedValue'.

Use case and example: This method might be useful to validate the result of a web service in XML format. Assuming an online library provides a web service allowing to gather detailed information of books when submitting a unique ISBN. The result from the web service is in XML format and contains information like author, title, year of publication and more of the found book. The XML result (see sample XML code further below) can now be validated using the code method with the following call:

C#

```
// call method to validate xml code
Validate_XMLResponse(xml, "books/book/@title", "Ranorex Test Automation Guide", "
Validating result of web service request ({0} vs. {1})");
```

VB.NET

```
' call method to validate xml code
Validate_XMLResponse(xml, "books/book/@title", "Ranorex Test Automation Guide", "
Validating result of web service request ({0} vs. {1})")
```

Note

This method can also be called from the recorder's action table. Therefore the usercode file needs to provide this method (directly or derived from a base class).

C#

```
public void Validate_XMLResponse(string xmlContent, string node, string expectedV
alue, string customLogMessage)
{
 string actualValue = string.Empty;

 // check if xml content is empty
 if (string.IsNullOrEmpty(xmlContent))
 {
 throw new Ranorex.ValidationException ("Parameter 'xmlContent' is empty");
```

```
}

// check if node (XPath) is empty
if (string.IsNullOrEmpty(node))
{
throw new Ranorex.ValidationException("Parameter 'node' is empty");
}

// check if expected value is empty
if (string.IsNullOrEmpty(expectedValue))
{
throw new Ranorex.ValidationException("Parameter 'expectedValue' is empty");
}

System.Xml.XmlDocument document = new System.Xml.XmlDocument();
try
{
// load XML from text in xml file
document.LoadXml(xmlContent);

//select the node with given argument (XPath)
System.Xml.XmlNode desiredNode = document.SelectSingleNode(node);

if (desiredNode != null)
{
actualValue = desiredNode.InnerText;
}
else
{
```

```
 throw new Ranorex.ValidationException(string.Format("No node found with XPath '{
0}'!", node));
 }


 }
 catch (System.Xml.XmlException)
 {
 throw new Ranorex.ValidationException("Unable to load valid XML string!");
 }



 // prepare log file
 if (string.IsNullOrEmpty(customLogMessage))
 {
 customLogMessage = "Comparing XML response ({0} vs. {1})";
 }

 // validate if expected value and actual value are equal
 Ranorex.Validate.AreEqual(expectedValue, actualValue, customLogMessage);
}
```

VB.NET

```
Public Sub Validate_XMLResponse(xmlContent As String, node As String, expectedVal
ue As String, customLogMessage As String)
 Dim actualValue As String = String.Empty


 ' check if xml content is empty
 If String.IsNullOrEmpty(xmlContent) Then
```

```
Throw New Ranorex.ValidationException("Parameter 'xmlContent' is empty")
End If

' check if node (XPath) is empty
If String.IsNullOrEmpty(node) Then
Throw New Ranorex.ValidationException("Parameter 'node' is empty")
End If

' check if expected value is empty
If String.IsNullOrEmpty(expectedValue) Then
Throw New Ranorex.ValidationException("Parameter 'expectedValue' is empty")
End If

Dim document As New System.Xml.XmlDocument()
Try
' load XML from text in xml file
document.LoadXml(xmlContent)

'select the node with given argument (XPath)
Dim desiredNode As System.Xml.XmlNode = document.SelectSingleNode(node)

If desiredNode IsNot Nothing Then
actualValue = desiredNode.InnerText
Else
Throw New Ranorex.ValidationException(String.Format("No node found with XPath '{0}'!", node))

End If
Catch generatedExceptionName As System.Xml.XmlException
```

```
 Throw New Ranorex.ValidationException("Unable to load valid XML string!")
 End Try



 ' prepare log file
 If String.IsNullOrEmpty(customLogMessage) Then
 customLogMessage = "Comparing XML response ({0} vs. {1})"
 End If

 ' validate if expected value and actual value are equal
 Ranorex.Validate.AreEqual(expectedValue, actualValue, customLogMessage)
End Sub
```

Sample XML code

This sample XML code allows you to implement the sample from above.

XML

```
<books>
 <book title="Ranorex Test Automation Guide" author="Ranorex" year="2014">
 </book>
```