TOSCA Automation Tool Training

accenture technology

OBJECTIVES

After completing this lesson, you should be able to do the following:

- Identify the application of TOSCA Test Suite (TOSCA Commander) in automated testing.
- Identify and define the different components of TOSCA.
- Create Modules, Test Cases and Execution List using the TOSCA Commander.

COURSE OUTLINE

Day 1:

1. Installation and activation
2. Introduction to TOSCA Test Suite
3. TOSCA COMMANDER
4. WORKSPACE
5. Module
6. Xscan
7. Test Case
8. Table Steering
9. Control
10. Action Mode
11. Random Values
12. Dynamic Dates

Day 2:

1. Verify
2. Buffer
3. Calculations
4. Library
5. Rescan, Value Range, Module Merge
6. WaitOn
7. Self Defined TCP
8. Business Parameters

Day 3:

1. Dynamic Comparison {XB[]}
2. Explicit Name, Resolve Reference, TBOX Set Buffer, ResultCount
3. Execution List
4. Dynamic Test Case Generation
   - Microsoft Excel as a source of Data
5. Conditional Statements and Loops
6. Exam

INTRODUCTION - PRODUCT OVERVIEW

Tosca Testsuite is a software developed by Tricentis which allows automated, functional software to be tested.

The core of Tricentis Tosca Testsuite is **Tosca Commander™**, which enables easy creation, management, execution and analysis of TestCases.

Tosca XScan and **Tosca Wizard** scan the screens and their input fields and saves the information as modules in Tosca Commander™.

These modules contain technical information which is used to identify and steer screen items.

INTRODUCTION - BUSINESS DYNAMIC STEERING

Business dynamic steering: the concept behind TOSCA Commander is a model-driven approach to make "the entire test, and not just the input data, dynamic".

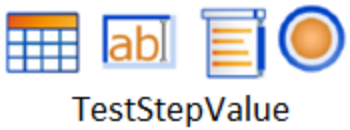
Test cases are built by dragging and dropping modules and entering validation values and actions. The dynamization of the test is supposed to enable a business-based description of manual and automated test cases so test cases can be designed, specified, automated and maintained by non-technical users (SMEs).

INTRODUCTION - TOSCA: COMPONENTS

TOSCA Test Suite (Components)

- **Requirements**
- Requirements are both the starting and the reference point in software development
- The test results are projected onto the **Requirements** in order to map the coverage from a test point of view.
- **TestCase-Design**
- Used for test data management.
- **TestCase**s
- Series of TestSteps that run from a defined starting point to a defined ending point.

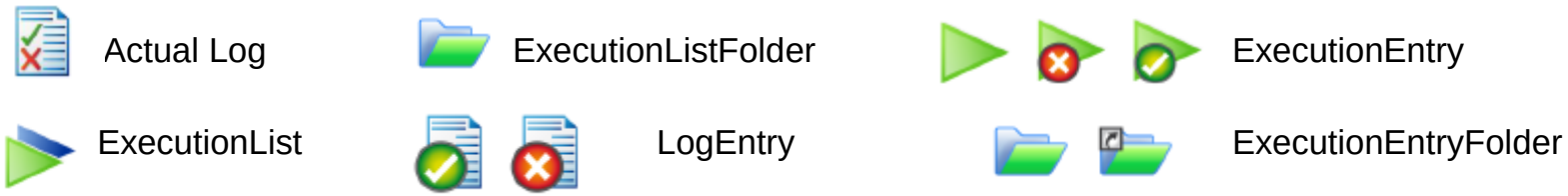
Icon	Description
	TestCase folder
	TestCase
	TestStep, XTestStep



INTRODUCTION - TOSCA: COMPONENTS

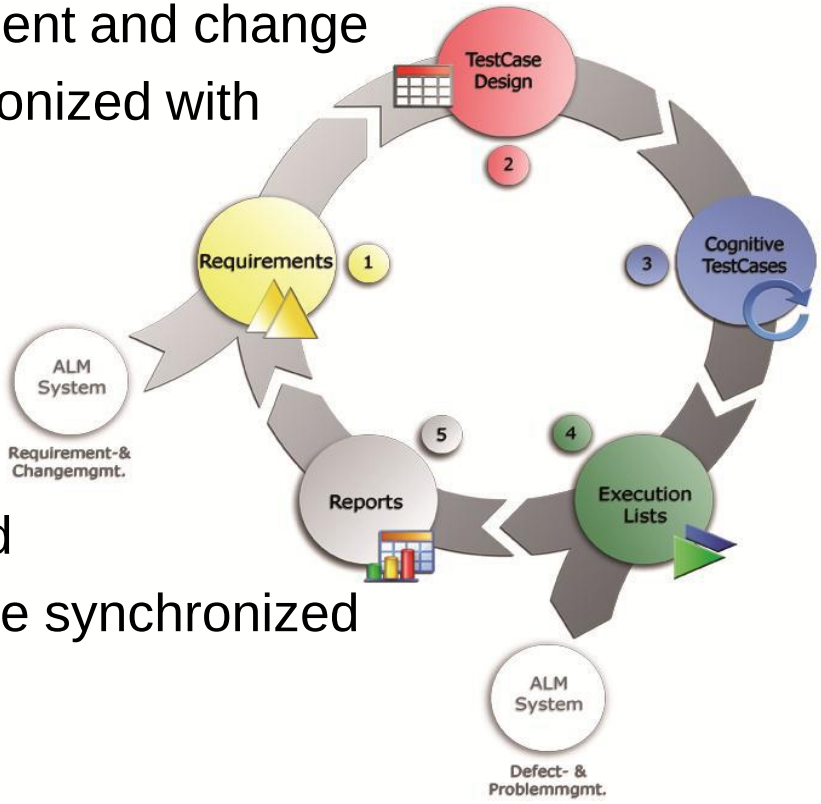
- Modules
- contain technical definitions of screens or screen areas and their technical names as well as the technical identification for the controls.

- ExecutionLists
- In this section, the actual test execution is performed.
- TestCases are combined into **ExecutionLists** and can be structured by using **ExecutionList folders** or **ExecutionEntry folders**.



INTRODUCTION - TOSCA TESTSUITE

- **Requirements**
  - Interface for requirements management and change management tools to be synchronized with *TOSCA Requirements*
- **Testcase Design**
- **Cognitive TestCase**
- **Execution Lists**
  - Interface for defect management and problem management tools to be synchronized with test results.
- **Reporting**

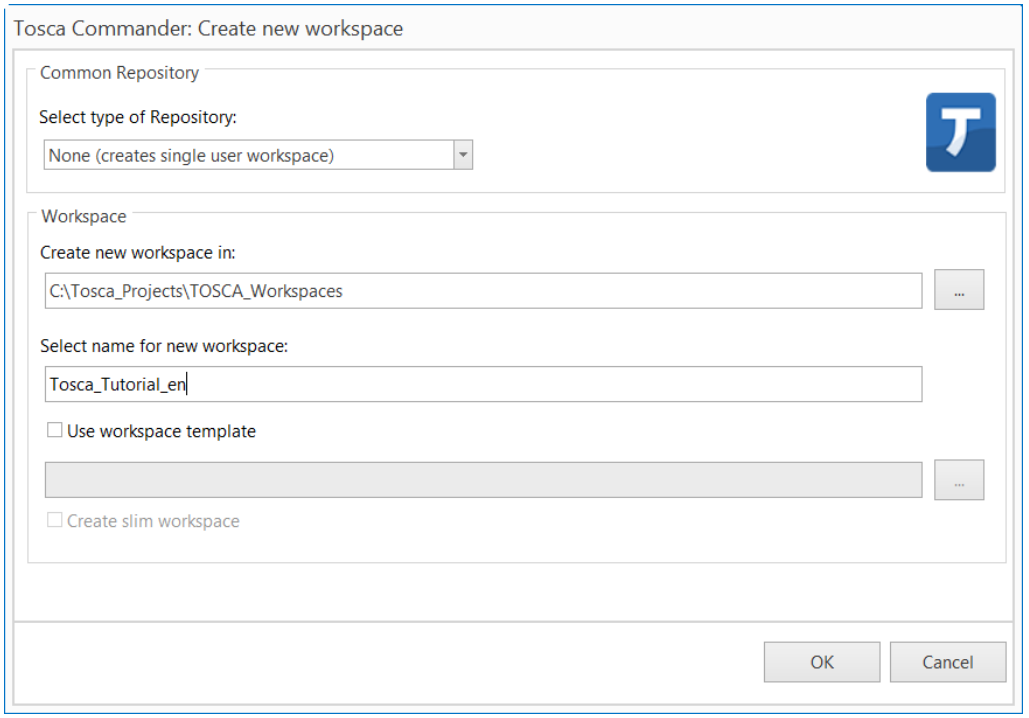


TOSCA COMMANDER

TOSCA COMMANDER - WORKSPACE

Creating Workspace

- Select **Project > New** from the Tosca Commander menu to open the **Tosca Commander: Create new workspace** window.
- Enter the name of your workspace into the **Select name for new workspace** field



TOSCA COMMANDER - MODULE

Creating Module

- The first step in test automation using *TOSCA Testsuite* is the mapping of functional elements of the system under test into Modules.
- In Tosca we refer to the creation of Modules as **Scanning**.
- Tosca analyzes the screen contents and searches for steerable items. You can then select the items you need and create a **Module** in Tosca.
- The Module contains all information required for steering the relevant items.

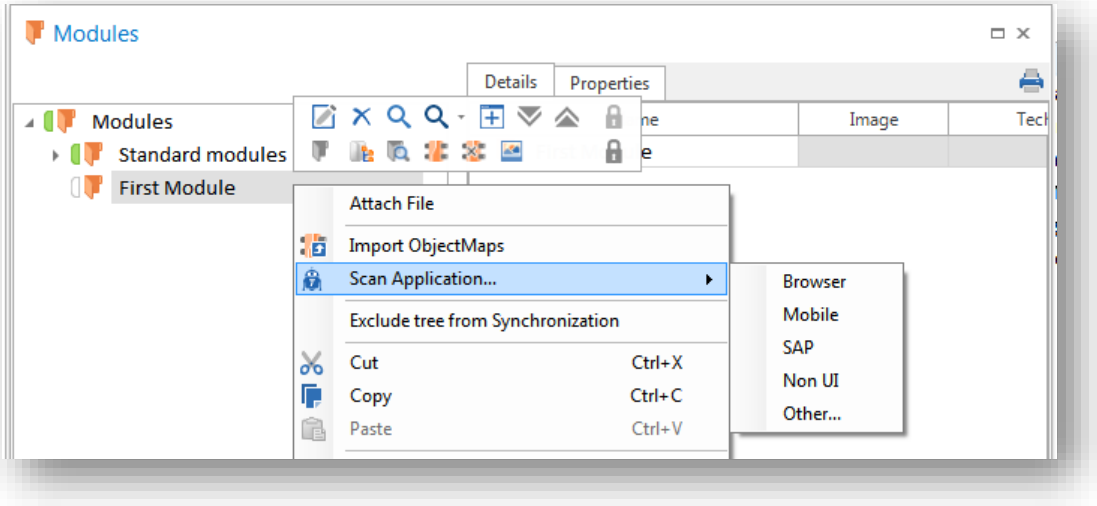
TOSCA COMMANDER - MODULE

Xscan

1.Right-click on the Module folder **Modules** and select the option **Create Folder** from the mini toolbar.

2.Assign a relevant name to the new Module folder.

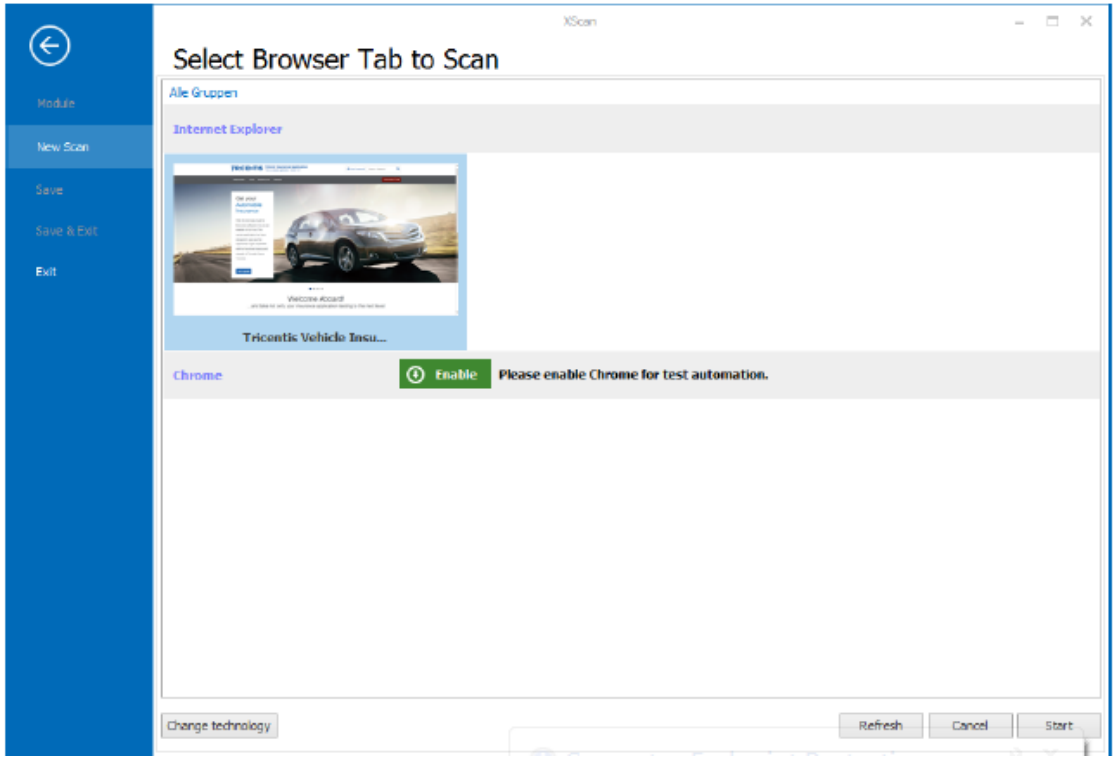
3.Right-click on the Module folder created and select the option **Scan Application -> Desktop** from the context menu.



TOSCA COMMANDER - MODULE

4.Tosca **XScan** loads all browsers and tabs that are currently open.

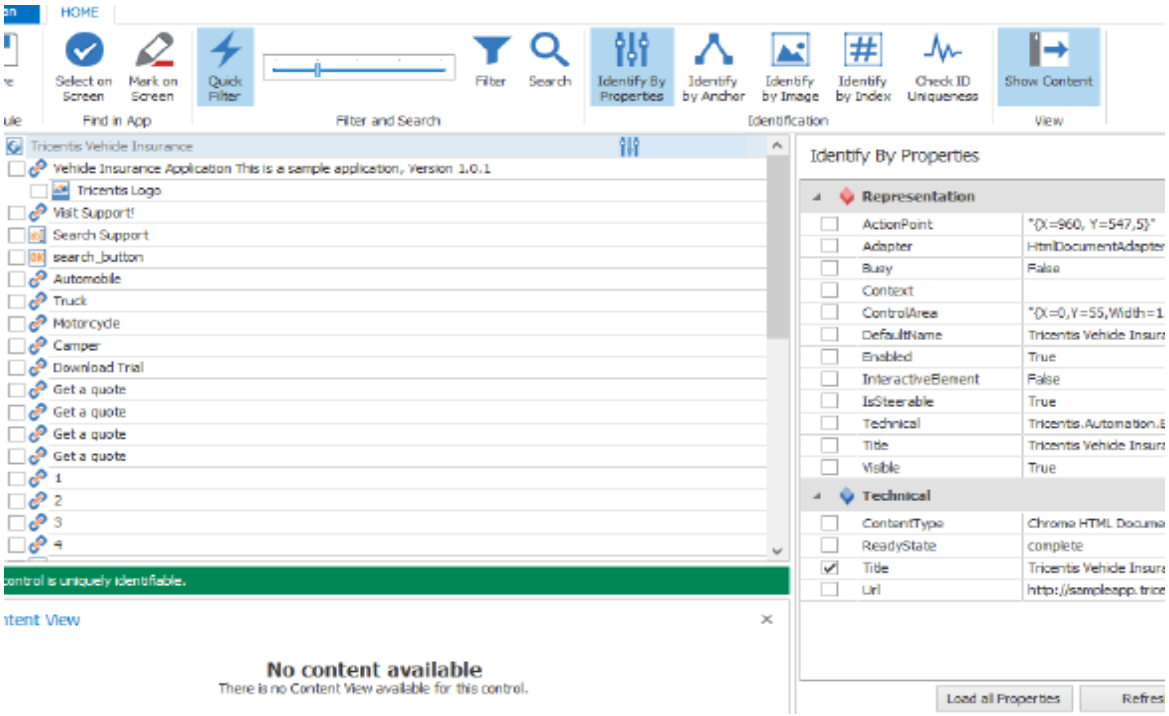
5.Click on the browser tab which shows the Tricentis sample application and



TOSCA COMMANDER - MODULE

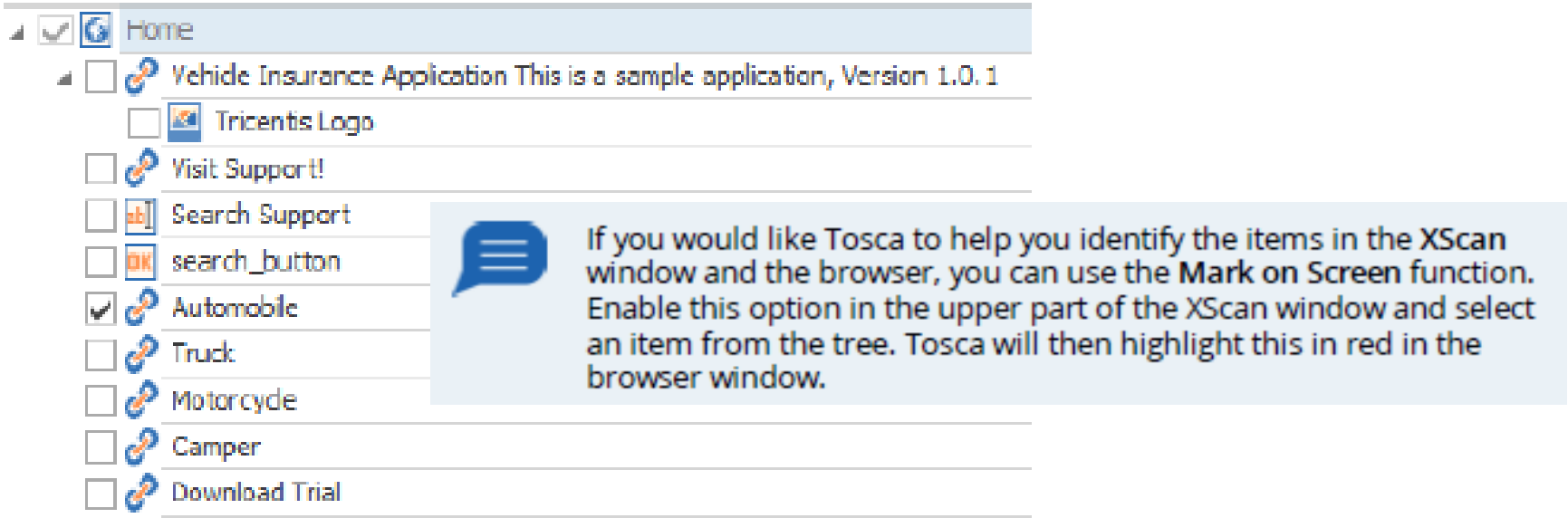
6.The **HOME** window now displays all items which were found during the scan process.

7.Although a large amount of items and functions will be shown in the **XScan** window, most of them are not relevant for this example. Search for the items that you need for your Module.



TOSCA COMMANDER - MODULE

8. Enable the checkbox in front of the items that needs to be included in the test case

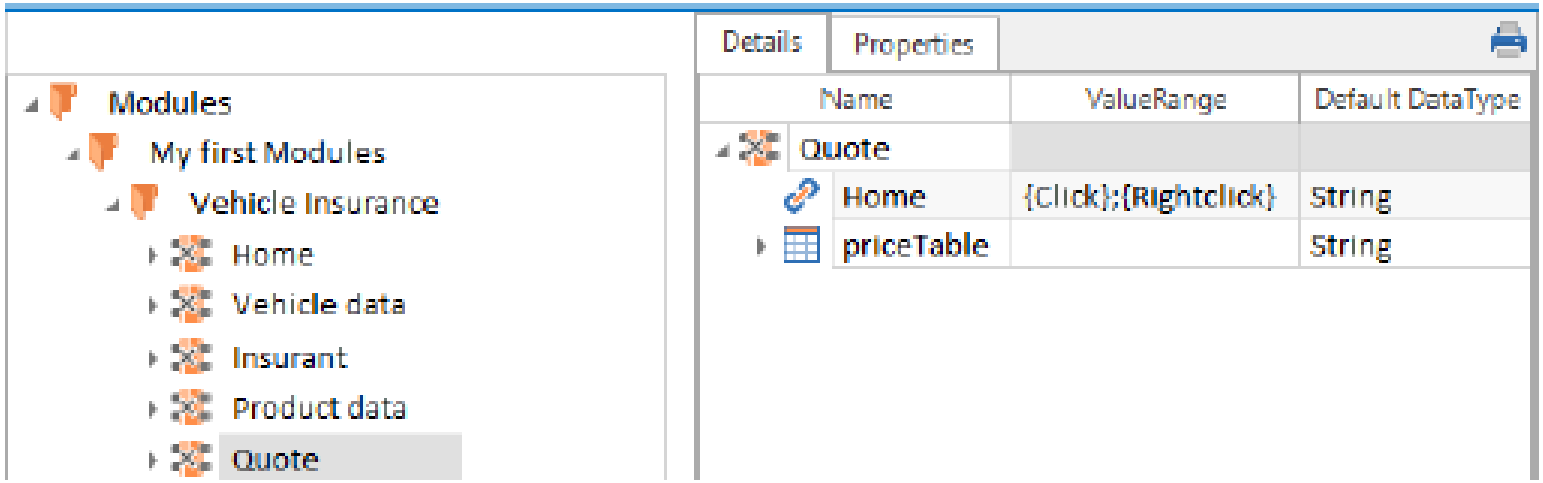


TOSCA COMMANDER - MODULE

9.Click on **Save** in the **XScan** window and close the window.

10.Tosca will now create a Module in the Module folder. The item selected (in this example: **Automobile**) will be shown in the Module as ModuleAttribute.

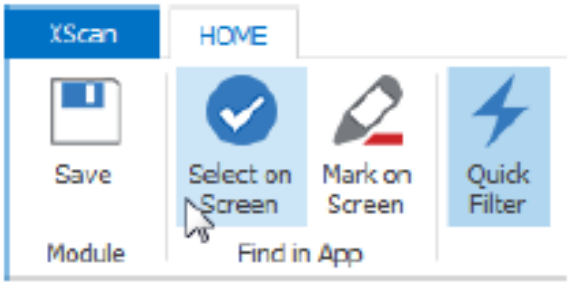
11.The Module holds any information that is required to click the link in the sample application.



TOSCA COMMANDER - MODULE

Xscan Using Select on Screen

Enabling the option **Select on Screen** in Tosca Xscan will help you select the needed items with ease.

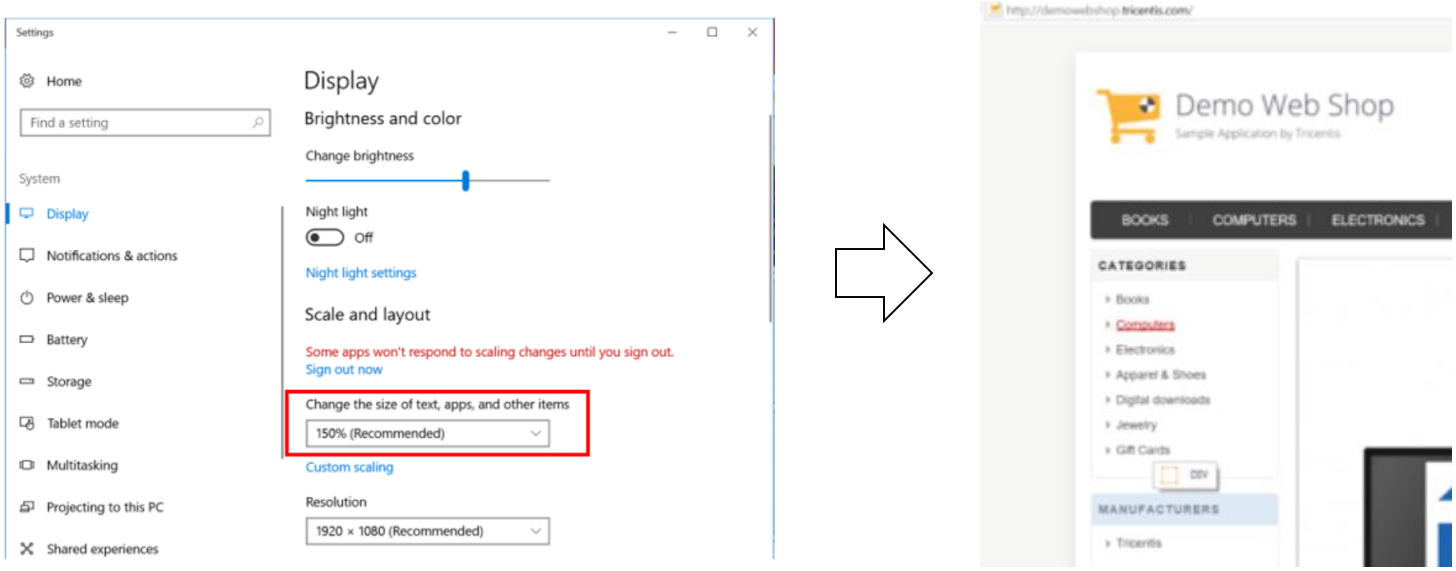


The **Select your controls on the screen** window will open upon clicking the **Select on Screen**

TOSCA COMMANDER - MODULE

Note: When using **Select on Screen**, Tosca assumes that the screen view is on **100% Zoom**.

If your using a mobile device or laptop with Auto Zoom/Resize feature, you should disable it or else the position of the object being selected would be inaccurate.



In this example, the ‘Manufacturers’ DIV is being highlighted even though the pointer is on the ‘Computers’ link. This is because the Display Setting is configured to resize the Zoom to 150%

RECOMMENDED LEARNING MATERIAL

In addition to this workbook it is necessary to use the following material to complete the exercises successfully.

Sample Web Shop application

- A sample Web Shop application is used for most of the exercises. Please use the link below

  to start the sample Web Shop application

  http://demowebshop.tricentis.com/

❑ **The training should be taken using Internet Explorer only.**

EXERCISES

Exercise: GETTING STARTED

Introduction: System Under Test (SUT)

Objective:

Identify the main steps of the order process within the sample Web Shop application.

Why is this important?

The scenario for this training is that we are responsible for testing the Web Shop, our SUT. To understand how to test the SUT we must have a good understanding of how it works.

Instructions

1. Open the Tricentis Demo Web Shop within Internet Explorer: **http://demowebshop.tricentis.com**

2. Register in the Web Shop using the following information:

User data:

· Gender: ..... < your gender >

· First name: ..... < your first name >

· Last name: ..... < your last name >

· E-mail address: ..... Your email, replace the .com, .at, etc. with .test

· Password: ..... Tosca1234!

· Confirm password: ..... Tosca1234!

Example Email address:…….Firstname.lastname@yourdomain.test

EXERCISES

3. User address:

· First name: ..... < your first name >

· Last name: ..... < your last name >

· E-mail address: ..... Email address used above

· Country: ..... Austria

· City: ..... Vienna

· Address 1: ..... Vienna street

· Zip:..... 1234

· Phone: 00 11 22 33 44 55

4. Navigate through the Web Shop by purchasing some items. Please use the following credit card details.

Credit card details:

· Select Credit Card: ..... Visa

· Cardholder name: ..... Barbara Gordon

· Card number: 4485564059489345

· Expiration date Month: 04

· Expiration date Year: 2020

· Card code: 123

Hints: Making a rough flow chart of the steps taken through the SUT will help you to understand the structure of the TestCases to come.

EXERCISES

Introduction: Tricentis Tosca

Objective: Create a single User workspace and import a Subset into a workspace.

Why is this important?

Each project will have its own workspace. Subsets allow you to import and export parts of a project, for example the Tricentis Standard Modules.

Instructions:

1. Download the Base Subset: AutomationSpecialistI_BaseSubset.tce from the Learning Management System.

2. Create a new single user workspace using the Base Subset as your template. Name this workspace **"AS1Training"**.

EXERCISES

Exercise 1 Module Creation

Exercise 1a - Create a Module Identify by Property

Objective: Use XScan to create a Module and use the technical properties of the Module to uniquely identify it.

Why is this important?

Modules store the technical information and therefore are the foundation of all test automation in Tosca.

Instructions

1.Navigate to the Modules section.

2.Navigate to the Module folder: **Webshop >> Customer >> 1a Log in Page.**

3.Open the Web Shop in Internet Explorer to the “User Log in” page.

***Remember:** Tricentis Tosca can be used with a variety of Browsers, however the training is designed for use with Internet Explorer. You may experience issues using other browsers.

4.Use XScan to scan the login page. Select:

“Email” text box

“Password” text box

“Remember Me?” checkbox link

“Log in” button **“Forgot password?”**

5.Rename any controls to match their purpose, e.g. **“True”** to **“Remember Me?”**

6.Within XScan, rename the Module to **“Log in Page”**.

7.Save the Module.

TOSCA DEFAULT VIEW

- **Navigation pane** – left hand side, to keep an overview of your work.

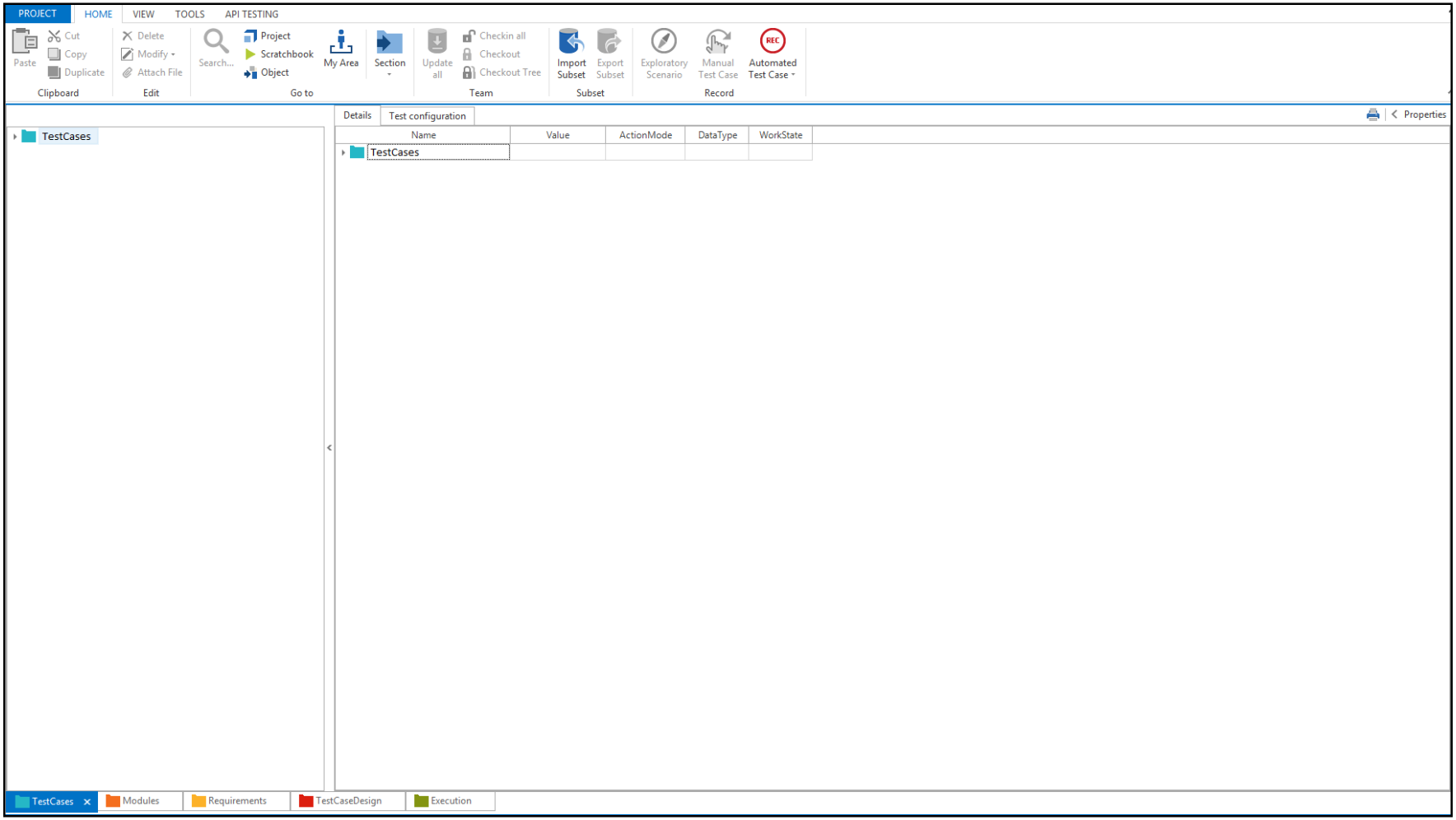
- **Working pane** – which you will focus on individual item.

  – Column header at the top of **Working pane**, were columns can be shown or hidden.

- **Menu** – is on the top made of RIBBONS, its many tab allow you to access a corresponding function.

- **Windows Tab** – is at the bottom, which are use to switch between different section.

TOSCA DEFAULT VIEW



TOSCA COMMANDER - TEST CASE

A TestCase in TOSCA Commander™

- Consists of m **TestSteps**, which contain n **TestStepValues**
- Describes the functional test process
- Is processed step by step from top to bottom
- Will always be linked to precise values
- Loops and If statements are possible

TOSCA COMMANDER - TEST CASE

- **TestCaseFolder** 
- TestCase folders are used to structure and manage TestCases.
- **TestCase** 
- TestCases detail a specific test sequence through the use of TestSteps. Their purpose is to test one or several values and characteristics of the system under test.
- **TestStepFolder** 
- A TestStep folder serves to manage individual TestSteps. A TestStep folder is used to give a clear overview of a TestCase's structure.
- **TestStep** 
- TestSteps define the sequence in which the test is carried out.
- Manual TestSteps and automated TestSteps differ from one another and are thus represented with different icons.
- An automated TestStep is the physical representation of a module.

EXERCISES

Exercise 2: TestCase Structure and Predefined Test Configuration Parameter

Objective: Create a logical TestCase structure within Tosca and set a predefined Test Configuration Parameter (TCP).

Why is this important?

A logical folder structure within your TestCase makes your work easier to create and maintain. Test configuration parameters are used to apply specific values to various TestCases; defined for various object types. A common TCP is to set the browser that Tosca is using to test.

Instructions

1. Navigate to the TestCases section.

2. Create a TestCase folder named **"TestCase 1"**. Within this new folder, create a folder named **"2TestCase Structure and TCP"**.

3. Within the TestCase 1 subfolder, create a TestCase named **"Shipping Costs"**.

4. Create the following TestStep folders within the TestCase you just created:

"Precondition"

"Order Product"

"Start Checkout"

"Checkout Process"

"Verification of Prices"

"Verification of Success

"Postcondition"

5. On the TestCase level create a Test Configuration Parameter named **"Browser"** with the value "InternetExplorer".

6. On the "Details" tab on the TestCase level, change the TestCase WorkState from **"PLANNED"** to **"IN_WORK"**.

EXERCISES

Hints:

- The folder structure follows the path taken through the application to order a product.
- Use the shortcut **“Ctrl+N, Ctrl+F”** to create a new folder.
- Use **“Ctrl+,”** to create a folder on the same level.
- Folder structures might change during the process of TestCase creation. folders might need to be added or deleted – this is normal

TOSCA COMMANDER - TEST CASE

- **TestStepValue** 
  - TestStepValues contain the actual information required for steering the system under test and can vary in form.
  - The TestStepValue icons are grayed out until you define actions for these values.

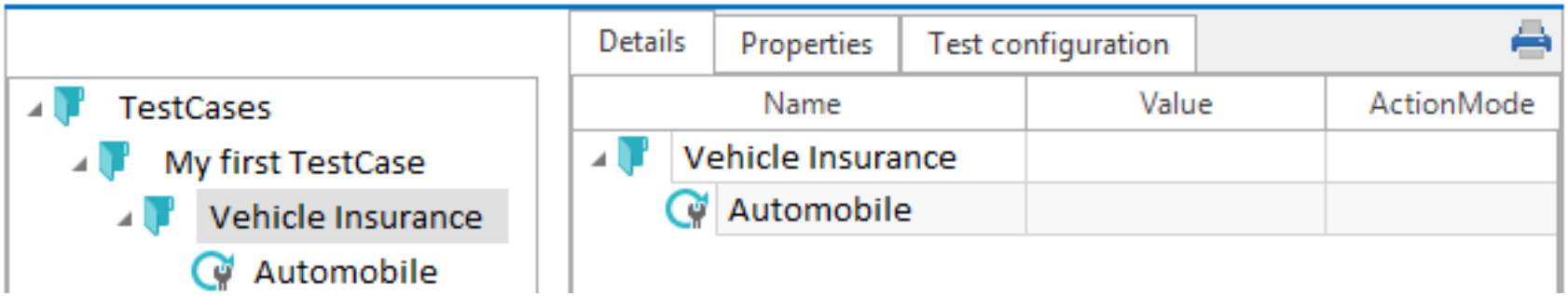
TestStepValues which are grayed out due to lacking values can either be hidden from view or shown via the F9 key.

TOSCA COMMANDER - TEST CASE

Creating TestCase

➢Right-click on the TestCase folder Vehicle Insurance and select Create TestCase from the mini toolbar.

➢Assign the name (e.g. **Automobile**) to the new TestCase**.**

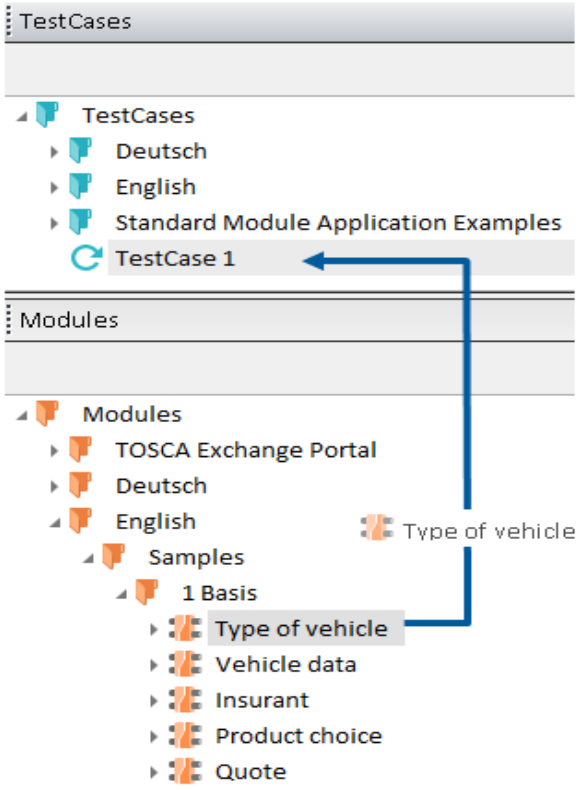


➢**Note:** Test cases should be named like in the test focus of a certain test case e.g. StdDriver_NoAccident_Older21

TOSCA COMMANDER - TEST CASE

Creating Automated Test Steps

- To create automated test steps, switch to the Modules window, drag the **Home** Module onto your **Automobile** TestCase and drop it there.



Note:

TestStep Name - A test step should be named after the function of the test step e.g. “Confirm entry“. It wouldn’t be the best practice to name the test step after the module e.g. “Transfer mask“.

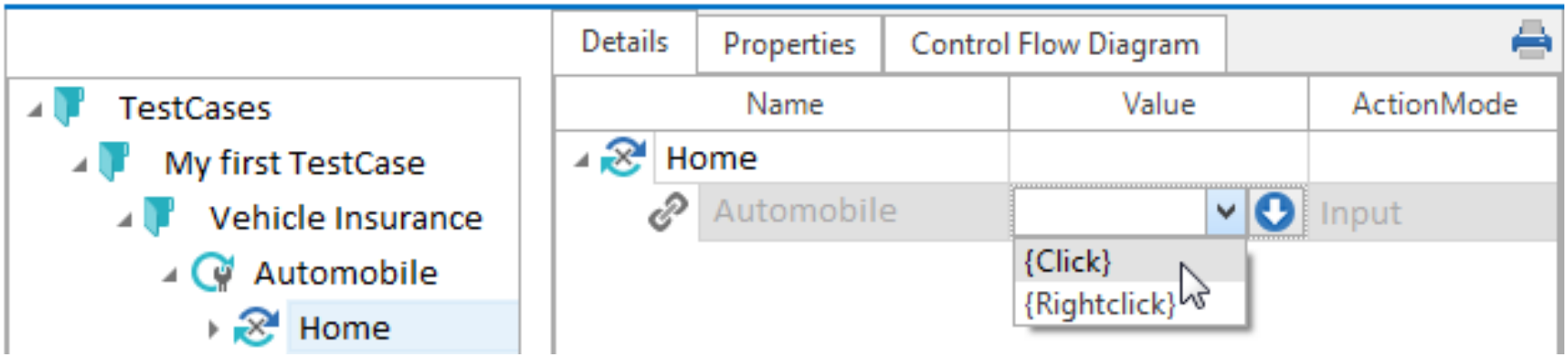
TOSCA COMMANDER - TEST CASE

Test Case Properties

Name - The name of the TestStep. Corresponds to the name of the control from the module definition.

Value - The action is performed using this value depending on the selected ActionMode.

ActionMode - ActionModes are used for steering the test object and they define how the value in the **Value** field has to be used in order to steer the control. In order to improve visibility,



TOSCA COMMANDER - SCRATCHBOOK

- The **Scratchbook** is a temporary aid for carrying out TestSteps during the process of creating a TestCase.
- Assigning TestSteps and/or sequences of TestSteps to the scratchbook can be done by means of:
  - Drag-and-drop
  - The command **Run in Scratchbook**
- Using the mouse and the keyboard during the execution of a test can influence its result.
- The execution results for each TestStep are displayed separately in the Scratchbook.
- The execution results will **NOT** be saved.

EXERCISES

Exercise 3a: Open and Close the Application

Objective: Use Tricentis Standard Modules to open and close an Html application.

Why is this important?

Opening the SUT is the recommended first step in the automation process. Closing it is the recommended final step.

Instructions

1. Duplicate TestCase folder **“2 TestCase Structure and TCP”** and name it **“3a Open and Close the Application“**.

2. View both the Module section and the TestCases section together on your screen. Navigate to the Modules section.

3. Navigate to: Tricentis Standard Modules >> TBox XEngines >> Html >> OpenUrl.

4. Drag the Module “OpenUrl” and drop it into the TestStep folder **“Precondition”**. Rename this TestStep **“Open Web Shop”**.

5. Navigate to Tricentis Standard Modules >> TBox Automation Tools >> Basic Windows Operations >>TBox Window Operation. Drag & drop this Module into the TestStep folder **“Postcondition”** and rename the TestStep **“Close Web Shop”**.

6. Navigate to the **“Open Web Shop”** TestStep and in the value column next to ”Url” enter the following link: **http://demowebshop.tricentis.com**

7. Navigate to **“Close Web Shop”** TestStep and in the value column next to **”Caption”** enter the text: **“Demo*”** and set the ActionMode to Select.

EXERCISES

8. In the Value column next to **“Operation”** choose **“Close”** from the drop down menu.

9. Ensure that you have closed all Demo Web Shop browser windows then right click on the TestCaseand select **“Run in ScratchBook”**, then save your work.

Hint:

- Use the window docking function to customize your workspace.

Exercise 3b: Create TestSteps

Objective: Fill the TestCase with the necessary TestSteps to automate the Web Shop.

Why is this important?

Modules create the TestSteps for an automated TestCase.

Instructions

1. Duplicate TestCase folder **“3a: Open and Close the Application” and name it “3b Create TestSteps”**.

2. Using the table below, add Modules from the Modules Section into the correct TestStep folders. You can do this using drag and drop. You can also use the **“Add TestStep”** function to find the Modules: Click on the TestStep folder where you wish to add the Module. Click **“Ctrl+T”**, and start typing the Module name. A list of Modules which match the search criteria will appear. Select the Module you wish to use by clicking on it and rename it according to the table.

EXERCISES

Path to Module	Module Name	Add Module into TestStep Folder	Rename the TestStep
Created in Ex. 3a	OpenUrl	Precondition	Open Web Shop
Webshop >> Navigation elements on all pages	Top Menu	Precondition	Navigate to Log in Page
Webshop >> Customer >> 1a Log in Page	Log in Page	Precondition	Log in
WebShop >> Products >> 1b Product Choice Tabs	Product Choice tabs	Order Product	Navigate to Apparel and Shoes
Webshop >> Products	Apparel & Shoes Product Selection	Order Product	Navigate to Blue Jeans
Webshop >> Products >> Products Category Apparel and Shoes	Blue Jeans	Order Product	Order Blue Jeans
Webshop >> Navigation elements on all pages	Top Menu	Start Checkout	Navigate to Shopping Cart
Webshop >> Check out process	Shopping Cart	Start Checkout	Shopping Cart Procedures
Webshop >> Check out process	Billing Address	Checkout Process	Billing Address Continue
Webshop >> Check out process	Shipping Address	Checkout Process	Shipping Address Continue
Webshop >> Check out process	Shipping Method	Checkout Process	Shipping Method Ground
Webshop >> Check out process	Payment Method	Checkout Process	Payment Method Credit Card
Webshop >> Check out process	Payment Information Credit Card	Checkout Process	Payment Information Credit Card
Webshop >> Check out process	Confirm Order	Verification of Prices	Verification of Prices
Webshop >> Check out process	Confirm Order	Confirmation	Confirm the Order
Webshop >> Order confirmation and details	Order Successful	Verification of Success	Verify the Order Success
Webshop >> Order confirmation and details	Order Successful	Postcondition	Continue

EXERCISES

Path to Module	Module Name	Add Module into TestStep Folder	Rename the TestStep
Webshop >> Navigation elements on all pages	Top Menu	Postcondition	Log Out
Created in Ex 3a	TBox Window Operation	Postcondition	Close Web Shop

TABLE STEERING

- The deployment of table steering is dependent on the engine (or technology) being used.
- Table steering facilitates the stable, business-relevant controlling of tables, even if the actual positions of the table cells change.
- The input parameters (TestStepSubvalues) of table steering are visible as soon as the ActionMode DoNothing is changed.
- Input parameters:
  - Action: Entry of a specific user action
  - Column: Entry specifying which column should be steered
  - Row: Entry specifying which row should be steered this info can be numeric* or a string
  - Value: A value, which refers to a particular cell

*numerical syntax: #<column or row number>

CONTROL

- A **control** in a system under test has a number of distinct **properties** at runtime.
- The current properties of the control are alphabetically listed in the **Control Properties** tab in **TOSCA Wizard**.
- The following control properties are available, irrespective of the control:
  - enabled
  - exists
  - visible
- Controls and their properties can be verified at the time of execution.

  Syntax:

  <ControlPropertyName> <Operator> <Value>

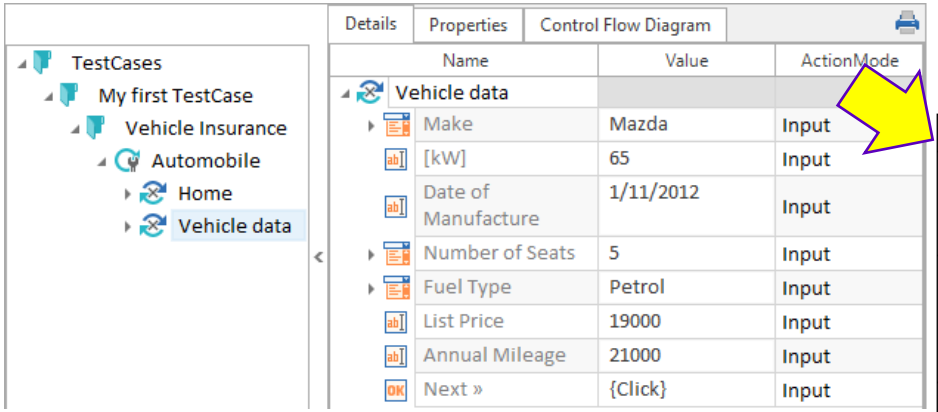
  e.g.: .exists=true
- Acceptable operators are: "=" “!=" ">"

ACTIONMODE

- The **ActionMode** determines how to process the entry in the *value* field for each individual TestStepValue.

- If an action or a value should be executed or passed on to the system under test, the action mode **input** must be used.
  - It is used to steer individual controls:
    - Enter, selecting a value
    - Operating a button etc.

ACTION MODES



ActionMode	Color	Description
DoNothing	white	With the ActionMode **DoNothing** no action is performed. The content of the field **Value** has no influence. The test object specified by the Module is steered, but not the control.
Input	white	**Input** is used for steering individual controls. This includes the input of a value, the operation of a control panel or a **ComboBox**, etc. The specific input mode is determined by the field **Value**.
Verify	green	Allows the value of a control provided by the application to be verified as specified in the field **Value**. Control properties can also be verified.
Buffer	yellow	If input values are repeatedly needed they can be saved with the ActionMode **Buffer**.
Output	blue	The value provided by the application to the control of the TestStep is read and stored in the XML ResultSet. These values are available in Tosca Commander™ ▪ if the TestCase was executed in an ExecutionList. ▪ in the column **Used Value** in the details view of the ExecutionList.
WaitOn	orange	The ActionMode **WaitOn** interrupts the execution of the TestCase until the respective control has the value or property that has been specified in the **Value** field.

RANDOM VALUES

Random Numbers:

- Syntax:

  {RND[<Number of Digits>]}

  A number between 1 and 32000 can be entered as the number of digits.

In this example, a random twelve-digit number is input:

Name	Value	ActionMode	DataType
Example			
Prefix	01	Input	String
Number	{RND[12]}	Input	String

- {RND[12][99]} will give a random number from 12 to 99.

RANDOM VALUES

Random Text

- Syntax:
- {RANDOMTEXT[<Number of characters>]}

In this example, a random text with 12 characters is input:

Name	Value	ActionMode	DataType
Insurant			
First Name	{RANDOMTEXT[12]}	Input	String

Random text

EXERCISES

Exercise 4a: TestStepValues

Objective: Enter test data into the TestSteps to steer the SUT.

Why is this important?

At this point the information for the test case must be entered: what to click, enter etc. into the SUT. Entering this information allows Tosca to steer and automate the test process.

Instructions

1. Duplicate TestCase folder **“3b Create TestSteps”** and name it **“4a TestStepValues”**.

2. Enter the following Values for the TestStepValues as per the table below:

TestStep Name	TestStep Value or Attribute	Value
TestStep folder Precondition		
Open Web Shop	Url	http://demowebshop.tricentis.com
Navigate to Log in Page	Log in	X
Log in	Email	Registered Email Address
	Password	Registered Password DataType Password
	Log in	X

EXERCISES

TestStep folder Order Product		
Navigate to Apparel and Shoes	Product Categories	Apparel & Shoes
Navigate to Blue Jeans	Blue Jeans	X
Order Blue Jeans	Quantity	25
	Add to cart	X
TestStep folder Start Checkout		
Navigate to Shopping Cart	Shopping cart	X
Shopping Cart Procedures	Terms of Service	True
	Checkout	X

EXERCISES

TestStep folder Checkout Process		
Billing address Continue	Continue	X
Shipping address Continue	Continue	X
Shipping method Ground	Shipping Methods	Ground
	Continue	X
Payment method Credit Card	Payment Methods	Credit card
	Continue	X
Payment information Credit Card	Select credit card	Visa
	Cardholder name	Barbara Gordon
	Card number	4485564059489345
	Expiration date | Month	04

EXERCISES

TestStep Name	TestStep Value or Attribute	Value
Payment info Credit Card (cont.)	Expiration date | Year	2020
	Card code	123
	Continue	X
TestStep folder Verification of Prices		
Verification of Prices	**LATER EXERCISE**	
TestStep folder Confirmation		
Confirm the order	Confirm	X
TestStep folder Verification of Success		
Verify the Order Success	**LATER EXERCISE**	
TestStep folder Postcondition		
Continue	Continue	X
Log out	Log out	X
Close Web Shop	Caption	Demo*
	Operation	Close

Hints

- F9 expands and contracts the detail that can be seen, as it hides all values with no actions specified.
- Each TestStep steers actions within the Web Shop. Some TestSteps require multiple actions, e.g. enter Log in details and click enter.
- Some TestStepValues have Values that can be entered via a drop down menu.

EXERCISES

Exercise 4b: Random Values

Objective: Enter Values to enter random text and numbers into the SUT.

Why is this important?

TestCases will be run repeatedly over long periods of time. Entering Random texts means that there is no need to create text for text fields that do not need to be verified, saving time.

Instructions

1. Duplicate TestCase folder **“4a TestStepValues”** 4a and name it **“4b Random Values”**.

2. Within the TestStep folder **“Checkout Process”** navigate to TestStep **“Payment information Credit Card”**

enter the correct syntax to:

- Generate a random 10 character length text in **“cardholder name”**
- Generate a random 3 digit number between 100 and 999 in the **“card code”** field

Hints:

- Right click in the Value field containing the syntax and select Translate value to test if your equation generates the required result.

DYNAMIC DATES

Date Expressions:

{DATE[]} → Date today

{DATE[][<*add or subtract*>][<*format*>]}

Dynamic date expressions are specified in curly brackets.

Expression	Description	Example
{DATE} or {DATUM}	Full date	30.12.2013
{DAY} or {TAG}	Current day	30
{MONTH} or {MONAT}	Current month	12
{YEAR} or {JAHR}	Current year (two-digit)	13
{RNDDATE[first year, last year]}	Random date within the specified year, e.g. {RNDDATE[2002, 2013]}. The result is transferred as a complete date.	08.10.2006
{TMSTMP}	Time stamp: current day, current month, current time and random year (accuracy in miliseconds)	30.12.2743 15:03:02.300007
{MONTHFIRST} or {MONATSERSTER}	First day of the current month as a complete date	01.12.2013
{MONTHLAST} or {MONATSLETZTER}	Last day of the current month as a complete date	31.12.2013
{QUARTERFIRST}	First day of the current quarter as a complete date	01.10.2013
{TRIMESTERFIRST}	First day of the current trimester as a complete date	01.09.2013
{HYEARFIRST}	First day of the current half year as a complete date	01.07.2013

DYNAMIC DATES

Adding or subtracting increments

Syntax: `[<Operator><Number><Unit>]`

Operator: `+` or `-`

Number: Number which is added or subtracted

Unit:

- `D` or `T` for days
- `W` or `A` for working days
- `M` for months
- `Y` or `J` for years

Units must be entered sequentially in descending order, e.g. `J`, `M`, `T`.

The current date in this example is 15/04/2010. In the following table, dynamic date values are created using this date as a starting point. The option **English (Great Britain)** in the **Format** tab must be selected in the Microsoft Windows® System settings under **Start->Control Panel->Regional and Language Options**.

Syntax	Result
{Ldate}	Thursday, 15 April 2010
{Date+1M+1D}	16/05/2010
{LDay}	Thursday
{LMonthfirst+35D}	Thursday, 06 May 2010
{LMonth+1M}	May
{Date[15/04/2010]+3D}	18/04/2010
{LMonthfirst[15/04/2010]+3D}	Sunday, 04 April 2010
{Quarterfirst[{Date[15/04/2010]}]+3M}	01/07/2010 (The first day of the next quarter)
{Trimesterfirst[{Date[15/04/2010]}]+4M}	01/05/2010 (The first day of the next trimester)
{HYearfirst[{Date[15/04/2010]}]+6M}	01/07/2010 (The first day of the next half year)

EXERCISES

Exercise 5: Dynamic dates

Objective: Set Tosca to automatically enter dynamic dates in the **“Payment info”** Credit card TestStep.

Why is this important?

TestCases will be run repeatedly over long periods of time. Setting dynamic dates means there is less maintenance due to checking/amending dates.

Instructions

1. Duplicate TestCase folder **“4b Random Values”** and name it **“5 Dynamic Dates”**.

2. Within the TestStep **“Payment information Credit Card”** enter the correct syntax to:

- Generate a dynamic date in the Expiration date | Month field. The date should be plus 4months, displayed as only a 2-digit month
- Generate a dynamic date in the Expiration date | Year field. The date should be plus 3 years, displayed as only a 4-digit year

Hints:

➢Pay close attention to closing any bracket that you open.

ACTIONMODE VERIFY

- **The ActionMode Verify is used to check values and characteristics in a system under test.**
  - The test instruction is entered into the **value** field of the **TestStepValue**, according to the specification.
- **The type of verification is determined by a logical operator**
  - **==** equals (standard, no entry is needed)
  - **!=** does not equal (specification of the logical operator is required)
  - When verifying a numerical value, both **<** and **>** are possible operators
  - When verifying a string, the verification always goes from **left** to right
  - Verification in tables is conducted using TOSCA‘s table steering.
- **Verification in tables is conducted using TOSCA‘s table steering.**

ACTIONMODE VERIFY

The syntax .enabled=true is used to verify whether the **Next** button is either enabled or available in the Tosca HTML sample application.

Name	Value	ActionMode
Type of vehicle		
Type of Vehicle	Truck over 1t (payload)	Input
Next	.enabled=true	Verify

The following control properties are available by default for verify operations:

Name	Description
.enabled	This verifies whether a control is available and enabled
.exists	Verifies whether a control exists
.value	The current value of a control is verified. This control property is used if no other property is explicitly specified. **Example:** It does not make any difference if either .value=ABC or ABC is specified in the **Value** column. They are both interpreted the same way.

The following control properties are available for list box

Name	Description
.contains	This verifies whether a specific content exists in a ComboBox or a ListView.
.list	The entire content of a ComboBox or ListView is verified. The entered values are case-sensitive, and they are verified according to the specified sequence. The content of the entire list is verified. Individual values are specified separated by semicolons ;.

ACTIONMODE BUFFER

- The ActionMode Buffer is used to save variable values from the system under test to a **local buffer (variable storage memory)**.
  - **Stored values** can be used at any stage during test execution :
    - {B[name of saved variable]}
  - Stored values can be viewed in the menu item:
    - **Tools->Settings-> Engine->Buffer**.

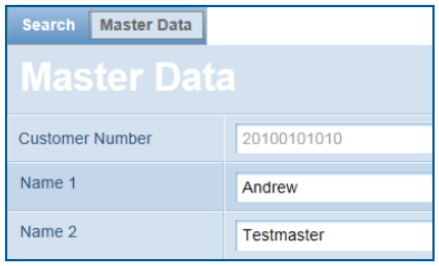


Illustration : TRICENTIS Bank - GUI

Illustration : TRICENTIS Bank - database

1 Save variable in local buffer

Name	Value	ActionMode	DataType
Master Data			
Customer Number	<name of variable>	Buffer	String
Name 1		DoNothing	*String

2 Use saved variable from local buffer

Name	Value	ActionMode	DataT
Database			
Customer Number	{B[name of variable]}	Input	String
Name 1		DoNothing	*String

ACTIONMODE BUFFER – TABLE STEERING

Search	Master Data	Conditions	Transactions

Master Data

Account Number	10000000007
Customer Number	20100101010
Name 1	Andrew
Name 2	Testmaster
Customer Type	Private Customer

1 Save variable in local buffer

Details | Properties

Name	Value	ActionMode	DataType
TestStep Buffer			
Roottable		Buffer	String
Action	<name of variable>		
Column	<specified column>		
Row	<specified row>		
Value			*

2 Use saved variable from local buffer

Details | Properties

Name	Value	ActionMode	DataType
TestStep Verify			
Roottable		Verify	String
Action			*
Column	<specified column>		
Row	<specified row>		
Value	{B[name of variable]}		

Illustration : TRICENTIS Bank - database

CALCULATIONS

Tosca allows calculations to be performed with the expressions **MATH**.

- Syntax:
- {MATH[<Operand 1><Operator><Operand 2>..<Operator><Operand n>]}

The following operators are supported:

Operator	Description
+, -, *, /	basic arithmetic operations
%	Modulo operation
==	equals
!=	not equal
&&	and operation for two items
||	or operation for two items
<	less than
>	greater than
<=	less or equal
>=	greater than or equal to
&, |, ^, <<, >>,	Bit-wise operators: and, or, xor, left shift, right shift
!, ~	Unary operators: not, bitwise not

EXERCISES

Exercise 6a: Buffer

Objective: Set a Buffer to save information within Tosca.

Why is this important?

The Buffer allows you to temporarily store information in Tosca, which can used later on in the TestCase to verify values, identify Modules etc..

Instructions

1. Duplicate TestCase folder **“5 Dynamic Values”** and name it **“6a Buffer”**.

2. Navigate to the TestStep **“Order Blue Jeans“** in the **“Order Product”** folder.

3. Buffer the control **“Price”** and name the buffer value **”PriceBlueJeans”**.

Hints:

- The color of the ActionMode field changes depending on the ActionMode selected, giving a visual confirmation that the correct ActionMode has been selected.
- Clear naming for a Buffer is important. When you use the Buffer value, you must duplicate the name exactly.

EXERCISES

Exercise 6b: Verify

Objective: Verify values generated in the SUT.

Why is this important?

In automated testing, it is important to check that the correct values are generated. The main goal in this TestCase is to verify that the shipping costs according to your order are calculated accurately.

Instructions

1. Duplicate TestCase folder **"6a Buffer"** and name it **"6b Verify"**.

2. Expand the TestStep **"Verification of Prices"** within the folder **"Verification of prices"**.

3. Verify that the shipping cost (when **"Ground"** is selected as a delivery method) is 10.00.

4. Run the TestCase in the ScratchBook.

Hints:

- The shipping costs are in column 2 and the shipping costs cell will be in the drop down list called **"Shipping:*"**

Exercise 6c: Verify Calculated Value

Objective: Verify a value in the SUT using a Buffer and verify the appearance of an object in response to an action in the SUT.

Why is this important?

In order to verify the Sub-Total, the previously saved Buffer must be used along with calculations using an on screen message. These steps allow the TestCase to meet all the verification requirements.

Instructions

1. Duplicate TestCase folder **"6b Verify"** and name it **"6c Verify Calculated Value"**.

2. Expand the TestStep **"Verification of Prices"** in the folder **"Verification of Prices"**.

3. Verify the order **"Sub-Total"** using:

EXERCISES

• The Buffer **“PriceBlueJeans”**

• The MATH function to multiply the Buffer by the quantity of blue jeans ordered (in this case 25)

✓ Remember that depending on the SUT you may need to change the Data Type to numeric.

4. Buffer the Sub-Total using the buffer name **“SubTotal"**.

✓ Remember - the Buffer name must exactly match the name used when it was created.

5. Verify the order **“Total”** using the MATH function: This requires adding the **“Sub-Total”** Buffer to the cost of the shipping (10.00)

6. Expand the TestStep **“Verify the Order Success”** in the **“Verification of Success”** folder. Verify that the **“Message Order successful”** message is visible.

7. Run the TestCase in the ScratchBook.

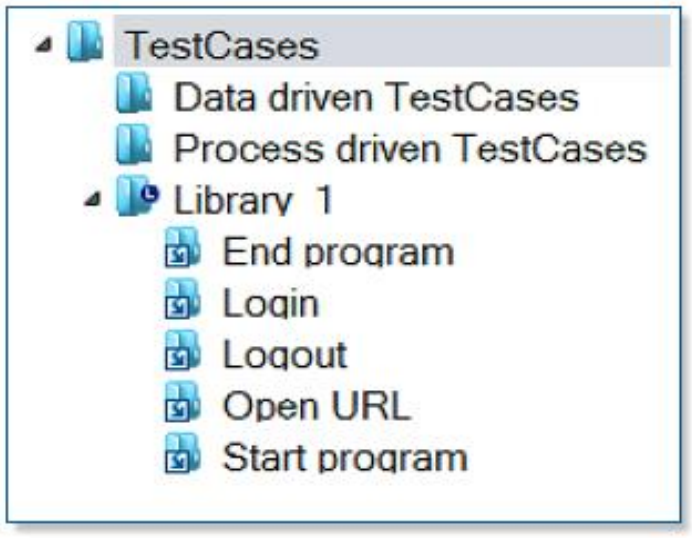
8. Set the TestCase WorkState to **“COMPLETED”**.

Hints:

➢ The shipping costs are in column 2 and the shipping costs cell will be in the drop down list called **“Shipping:*”**

TEST STEP LIBRARY

- TestSteps and TestSteps sequences, which are repeatedly used (without modification) in several TestCases, can be centrally managed in a **TestStepLibrary** in *TOSCA Commander*™.
- A TestStepLibrary is a special folder, which exclusively contains **Reusable TestStepBlocks**.
  - Symbol **TestStepLibrary**
  - Symbol **Reusable TestStepBlock**



REUSABLE TESTSTEPBLOCK | REFERENCES

- If a **Reusable TestStepBlock** is being used to generate TestSteps, a **Reference** to the Reusable TestStepBlock will be created in the TestCase.

  The symbol for a **Reference** 

- Each change within a Reusable TestStepBlock, or in a reference will be reflected in all references.

- The link between a Reusable TestStepBlock and a Reference can be broken at any time (using the command **Resolve Reference**).

62

USING LIBRARIES

Note:

If more than one test step is used in many test cases we recommend to use **reusable test step blocks** that are located in **libraries**. A typical example for a reusable test step block is a login procedure. The login will be used in many test cases and if something changes it is less effort to integrate those changes at one **centralized location**. So all usages are only linked to the library and can be changed in that library as well.

EXERCISES

TEST CASE TWO

Exercise 7a: Create a Library

Objective: Create a Library of Reusable TestStepBlocks.

Why is this important?

A Library greatly reduces the amount of work needed to create the multiple TestCases that are needed to fully automate TestCases in the SUT.

Instructions

1. On the TestCases root folder, create a TestStepLibrary.

2. Drag and drop all the folders from the TestCase in Ex. 6c into the Library folder, except: **"Verification of Prices"**.

Hints

- Library folders are sorted alphabetically.

Exercise 7b: Create Reusable TestStepBlocks

Objective: Use the Reusable TestStepBlocks to create References in a TestCase.

Why is this important?

This exercise will demonstrate how it is much more efficient to use a Library to create automated TestCases.

Instructions

1. On the TestCase root folder, create a new TestCase folder named **"TestCase 2"**.

2. Within your new folder, create a subfolder named **"7b Create Reusable TestStepBlocks"**. Create a new TestCase named **"Payment Process"**, Add the Test Configuration Parameter named **"Browser"** with the value **"InternetExplorer"**.

EXERCISES

3. Drag the following folders from the Library, and drop them into the TestCase:

- **"Precondition"**
- **"Order Product"**
- **"Start Checkout"**
- **"Confirmation"**
- **"Verification of Success"**
- **"Postcondition"**

4. Create two new TestStep folders and name them:

- **"Checkout Process"**
- **"Verification of Prices"**

❖ Reorder the folders to follow the Web Shop process flow (see TestCase 1)

5. Navigate to (Webshop >> Check out process) the Module folder named **"7b Payment Information Check Money Order"**

6. Within this folder, create a new Module of the Web Shop **"Payment Information"** page that appears when you select **"Check Money Order"** and click **"Continue"**

❖ Scan this page and add the controls:

- **"Message"** table
- **"Back"** button
- **"Continue"** button

❖ Make sure the controls are uniquely identified.

❖ Name the Module **"Payment Information Check Money Order"** and save.

EXERCISES

7. Add the following Modules into the **"Checkout Process"** TestStep folder and rename them accordingly:

- **"Billing Address"**
- **"Shipping Address"**
- **"Shipping Method"**
- **"Payment Method"**
- **"Payment Information Check Money Order"**

8. Add the **"Confirm Order"** Module into the **"Verification of Prices"** TestStep folder and rename the TestStep **"Verification of Prices"**.

9. Enter the following Values for the new TestStepValues as per the table below:

TestStep	TestStepValue	Value	ActionMode
TestStep folder Checkout Process			
Billing address continue	Continue	X	Input
Shipping address continue	Continue	X	Input
Shipping method ground	Shipping Methods	Ground	Input
	Continue	X	Input
Payment method check money order	Payment Methods	Check / Money Order	Input
	Continue	X	Input
Payment information Check Money Order	Continue	X	Input

EXERCISES

TestStep folder Verification of Prices			
Verification of Prices	Cart Total		Select
	Column: Rename to #2		Select
	Cell: Choose Sub-Total	{MATH[{B[PriceBlueJeans]}*25]}	Verify DataType: Numeric
	Cell: Choose Sub-Total	SubTotal	Buffer
	Cell: Shipping:*	10.00	Verify DataType: Numeric
	Cell: Payment method additional fee:	5.00	Verify DataType: Numeric
	Cell: Total	Use the SubTotal Buffer to verify the total cost adding any shipping / additional fees.	Verify DataType: Numeric

10. Run the TestCase in the ScratchBook.
11. Set the TestCase WorkState to **"COMPLETED"**.

RESCAN, VALUE RANGE, MODULE MERGE

Rescan and Module Merge

- Modules can be amended; they do not have to be recreated every time you wish to change them. Modules can be reused any number of times and are not specific to the TestStep. Modules are technical representations of the SUT in Tosca, so we should not have multiple Modules for the same controls. Module Merge helps solve this duplication problem.

Value Range

Details | Properties

Name	Image	TechnicalID	TypeInfoDescription	ValueRange
Payment Information...				
Select credit card				Visa;Master ...
Cardholder name				Barbara Gor...
Card number				4485564059...
Expiration date | ...				01;02;03;04;...
Expiration date | Y...				2017;2018;2...
Card code				123
Back				X
Continue				X

ACTIONMODE WAITON

- **Asynchronous process operations** occur in practically every complex test project
- Test execution must be adapted to these operations, so that test instructions are **not prematurely sent** to the system under test (which is not yet ready)
- **Dynamic synchronization** with the process speed of the system under test (SuT)
- The execution of a **TestStepValue is suspended** until the appropriate control has accepted the value or characteristic.
- Syntax:

.<ControlPropertyName> <Operator> <Value>
e.g. : .enabled=True (see control properties)

- The settings for **dynamic synchronization** can be found in the Engine settings.
  - Menu: Tools->Settings

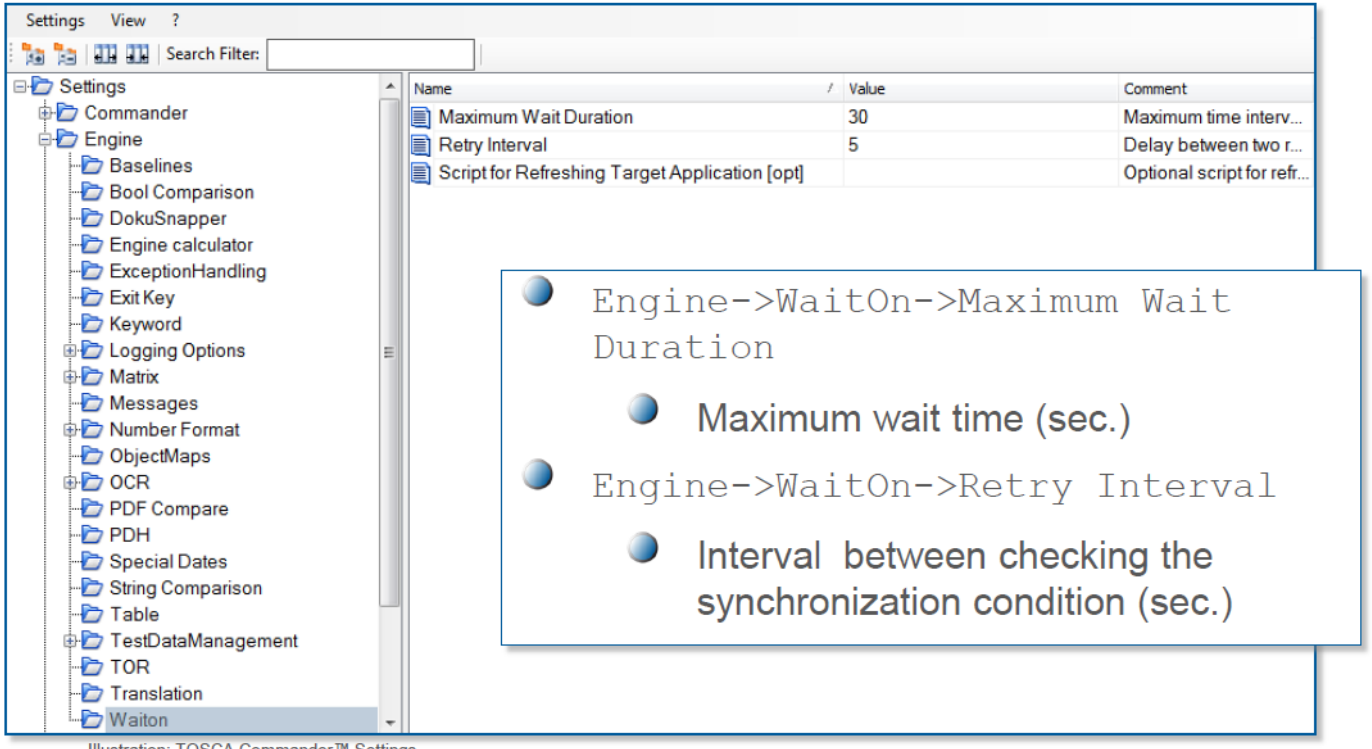


Illustration: TOSCA Commander™ Settings

EXERCISES

TEST CASE THREE

Exercise 8a: Rescan, Value Range and Module Merge

Objective: To use XScan to rescan a Module and then to merge to Modules to avoid duplication.

Why is this important?

Modules can be amended; they do not have to be recreated every time you wish to change them. Modules can be reused any number of times and are not specific to the TestStep. Modules are technical representations of the SUT in Tosca, so we should not have multiple Modules for the same controls. Module Merge helps solve this duplication problem.

Instructions

1. In the Web Shop, add items & navigate to the “Shopping cart” page; apply a discount code named:

•**AutomationDiscount2**

❖ Leave the page open as we will be scanning it momentarily.

2. Navigate to Webshop >> Check out process>>8a Rescan Shopping Cart in the Modules section.

3. Right click to Rescan the “Rescan Cart” Module which is in the 8a Rescan Shopping Cart folder.

4. Ensure the following controls are selected, in addition to those which are already in the Module:

•**“Update shopping cart”** button

•**“discountcouponcode”** textbox, rename it **“Enter discount code”**

•**“Apply coupon”** button

•**“Coupon applied message DIV”**, rename it **“Coupon applied message”**

•**“giftcardcouponcode”** text box, rename it **“Gift card coupon code”**

•**“Shopping cart cost total table”**, rename it **“Shopping cart costs”**

EXERCISES

5. Add a value range of possible codes to the discount code in the Module you just rescanned. Therange should include the following values:

•**AutomationDiscount2**

•**PercentageTotal**

•**PercentageShipping**

•**FlatTotal**

❖ Make sure to separate the values by semicolons.

6. Expand all the folders in the **"Check out process"** Module folder. You will notice that there are currently two Shopping Cart Modules (one called **"Shopping Cart"** and one called **"Rescan Cart"** in the8a Rescan Cart Module folder). **"Ctrl+Click"** to select both Modules simultaneously.

7. In the top ribbon, on the **"Modules"** tab, select Merge Modules.

8. The Target Module is the one we will keep, and we need to select the Attributes we want to bring from the Source Module into the target Module. Ensure that the Rescan Cart Module (the one we just scanned) is listed as the Target Module. If it is not, choose the option to Switch Modules.

9. Because of their identical properties, Tosca should automatically link the Attributes which exist in both Modules.

10. When you are ready, press **"Merge"**. Tosca will let you know how many Attributes have been updated, which usages have been relinked to the new Module, and deletes the Source Module. Press **"Close"**.

❖ Rename the Module from **"Rescan Cart"** back to **"Shopping Cart"** and save your work.

Hints

➢ To identify tables use Mark on Screen to locate the correct one.

➢ AutomationDiscount2 will apply a discount of 20% to the total value of the order.

EXERCISES

Exercise 8b: Create the TestCase

Objective: Use the Reusable TestStepBlocks to create References in a TestCase.

Why is this important?

The creation of logical and easy to follow structures within automated test cases makes maintenance and problem solving easier.

Instructions

1. On the TestCase root folder, create a new TestCase folder named **"TestCase 3"**. Create a sub-folder named **"8b Create the TestCase"**.

2. Within the sub-folder, create a new TestCase, name it "Discount Code". Add the Test configuration parameter: **"Browser"** with the value **"InternetExplorer"**.

3. Add the following Reusable TestStepBlocks from the Library into the new TestCase:

- **"Precondition"**
- **"Order Product "**
- **"Checkout Process"**
- **"Confirmation"**
- **"Verification of Success"**
- **"Postcondition"**

4. Create two new TestStep folders named:

- **"Start Checkout"**
- **"Verification of Prices"**

❖ Reorder the TestStep folders to match the process flow of the Web Shop

5. Add the following Modules into the "Start Checkout" folder:

- **"Top Menu"** – rename the TestStep **"Navigate to Shopping Cart"**
- **"Shopping Cart"** – rename the TestStep **"Shopping Cart Procedures"**

EXERCISES

6. Add the following Module into the **"Verification of Prices"** folder:

• **"Confirm Order"** – rename the TestStep **"Verification of Prices"**

7. Enter the following Values for the TestStepValues as per the table below:

TestStep	TestStepValue	Value	ActionMode
TestStep folder Start Checkout			
Navigate to Shopping Cart	Shopping Cart	X	Input
Shopping Cart Procedures	Enter Discount Code	AutomationDiscount2	Input
	Apply coupon	X	Input
	Terms of Service	True	Input
	Checkout	X	Input

EXERCISES

TestStep folder Verification of Prices			
Verification of Prices	Cart Total		Select
	Column (rename to #2)		Select
	Cell: Choose Sub-Total:	{MATH[{B[PriceBlueJeans]}*25]}	Verify
	Cell: Choose Sub-Total:	SubTotal	Buffer
	Cell Choose Shipping:*	10.00	Verify
	Cell: Choose Discount:	-{MATH[({B[SubTotal]}+10.00)*.2]}	Verify
	Cell: Choose Discount:	Discount	Buffer
	Cell (Total: from drop down)	Use the Sub-Total and Discount Buffers to verify the total including any shipping or fees	Set valid ActionMode

8. Run the TestCase in the ScratchBook

Hints

- Multiply a whole number by 0.20 to calculate a 20% discount.

EXERCISES

Exercise 8c: ActionMode WaitOn

Objective: Create a dynamic wait on a TestStepValue using the ActionMode WaitOn.

Why is this important?

WaitOn instructs Tosca to wait for an element or action to occur before moving on to the next TestStep. This benefits the automation as it can stop possible TestCase failures in situations where the SUT does not react immediately e.g. waiting for a credit card to authorize.

Instructions

1. Duplicate TestCase folder **"8b Create the TestCase"** and name it **"8c ActionMode WaitOn"**.

2. Navigate to the TestStep **"Shopping Cart Procedures"** in the folder **"Start Checkout"**.

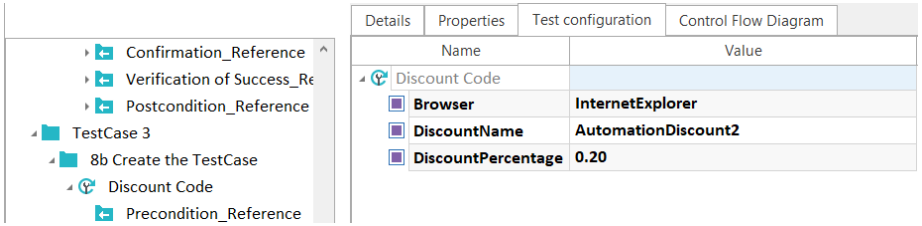
3. Set the ActionMode WaitOn for the TestStepValue **"Coupon applied message"**, instructing Tosca to wait for the message to appear before executing the next TestStep.

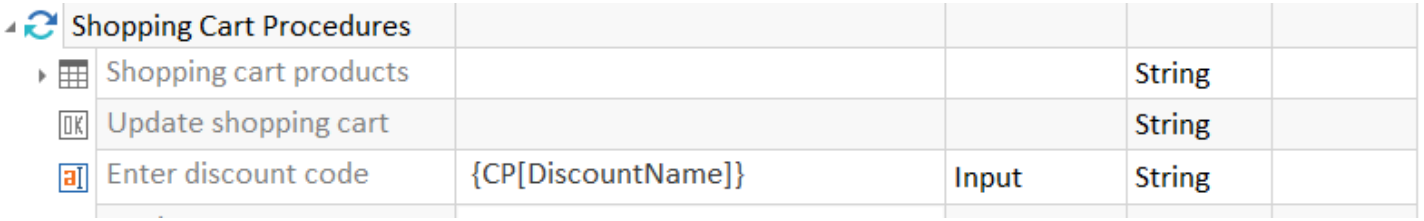
❖ Remember: use the blue downward arrow to select: Visible == True

SELF DEFINED TCP

Self Defined Test Configuration Parameter

- Test configuration **parameters** allow values to be set centrally which can then be used within the TestCases. If the value needs to be changed, rather than changing each occurrence on every TestStep, the value can be amended just **once in the central location**.





EXERCISES

Exercise 9: Self-Defined Test Configuration Parameter

Objective: Set and use self-defined Test Configuration Parameters.

Why is this important?

Test configuration parameters allow values to be set centrally which can then be used within the TestCases. If the value needs to be changed, rather than changing each occurrence on every TestStep, the value can be amended just once in the central location.

Instructions

1. Duplicate TestCase folder **"8c ActionMode WaitOn"** and name it **"9 Self Defined TCP"**.

2. Add two additional Test Configuration Parameters to the TestCase:

- DiscountName – set Value to **"AutomationDiscount2"**
- DiscountPercentage – set Value to **"0.20"**

3. In the **"Shopping Cart Procedures"** TestStep use the **"DiscountName"** TCP in the TestStep Value for **"Enter Discount Code"**.

4. In the **"Verification of Prices"** TestStep amend the Discount calculation to use the **"DiscountPercentage"** TCP instead of the numeric value 0.20 to calculate the discount total.

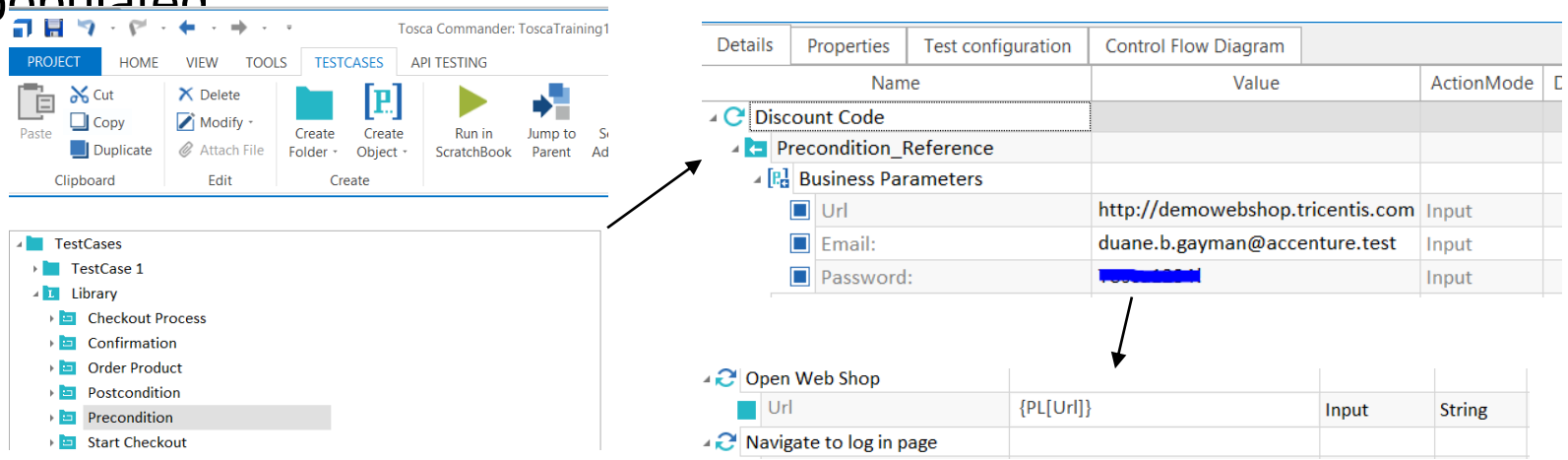
Hints

- **{CP[]}** is the syntax required to use a TCP.

BUSINESS PARAMETERS

Business Parameters enhance Reusable TestStepBlocks, to allow the creation of Reusable TestStepBlocks from TestSteps even when the TestStep contains values that need to be changed in different TestCases.

- Click the Reusable Test Step Block
- Click the Create Object button from the ribbon.
- Test Steps will be **invisible** from the Test Cases that refers to the library and will just have the **Business Parameters** to be populated



EXERCISES

Exercise 10: Business Parameters

Objective: Create Business Parameters in the Library for use in TestCases.

Why is this important?

Business Parameters enhance Reusable TestStepBlocks, to allow the creation of Reusable TestStepBlocks from TestSteps even when theTestStep contains values that need to be changed in different TestCases.

Instructions

1. Duplicate TestCase folder **"9 Self Defined TCP"** and name it **"10 Business Parameters"**.

2. In the Library, navigate to the folder precondition and add three Business Parameters:

- URL
- Email
- Password

3. In the TestStep, enter the correct Business Parameters.

- Url
- http://demowebshop.tricentis.com
- Email
- Registered Email address
- Password
- Registered Password

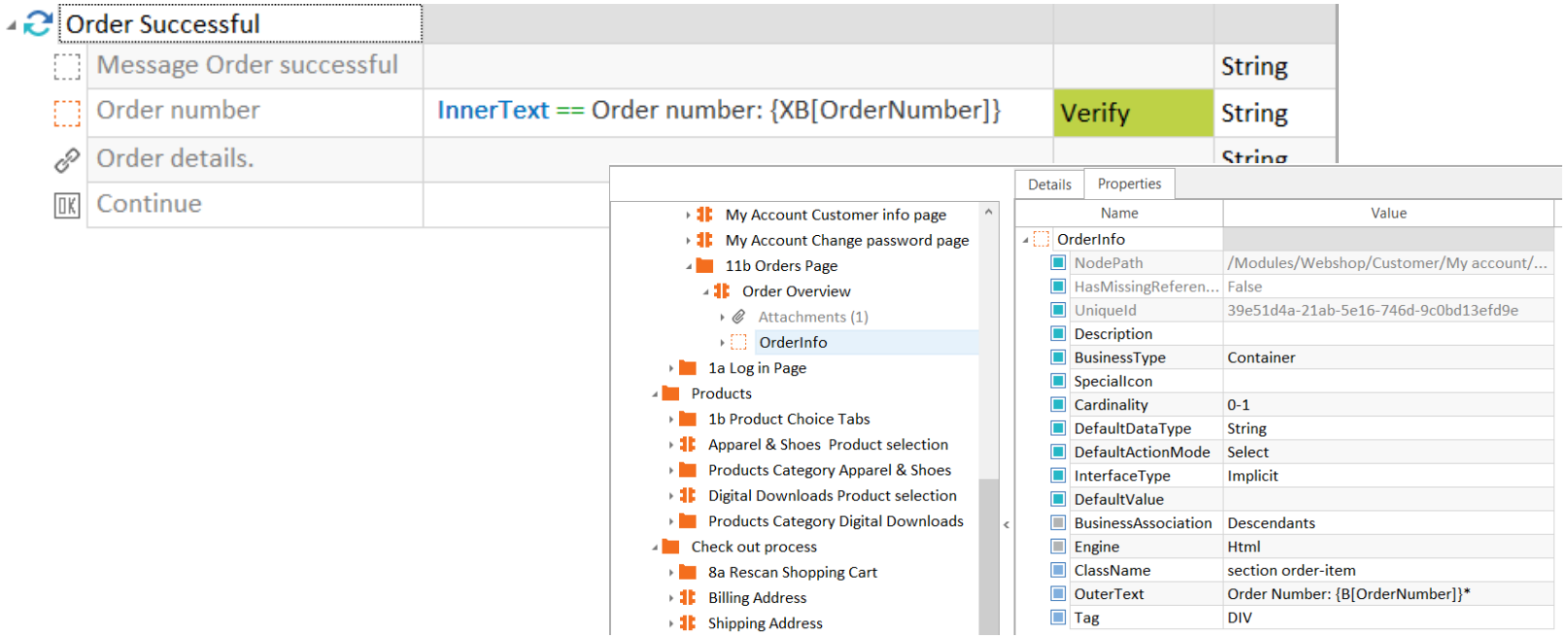
4. *Remember, all previous TestCases will need to have the relevant Business Parameters entered.

5. Set the TestCase WorkState to **"COMPLETED"**

DYNAMIC COMPARISON

Dynamic Comparison

- {XB} allows you to verify a **dynamic string** by excluding the dynamic part with the additional option to **Buffer the excluded value**. This helps verify strings with dynamic elements as well as use the buffered elements.



EXERCISES

TEST CASE FOUR

Exercise 11a: Dynamic Comparison

Objective: To set a Dynamic Comparison to exclude and buffer part of a string.

Why is this important?

{XB} allows you to verify a dynamic string by excluding the dynamic part with the additional option to Buffer the excluded value. This helps verify strings with dynamic elements as well as use the buffered elements.

Instructions

1. Create a new TestCase folder, named **"Test Case 4"**

- Create a subfolder named **"11a Dynamic Comparison"**
- Add a TestCase named **"Reorder"**
- Add the Test Configuration Parameter named **"Browser"** with the value **"InternetExplorer"**

2. Create the TestCase using Reusable TestStepBlocks or creating new folders where necessary:

- **"Precondition"**
- **"Order Product"**
- **"Start Checkout"**
- **"Checkout Process"**
- **"Verification of Prices" (create new folder)**
- **"Confirmation"**
- **"Verification of Success"**
- **"Buffer Order Number" (create new folder)**
- **"Previous Orders" (create new folder)**
- **"Reorder" (create new folder)**

EXERCISES

- **"Start Checkout"**
- **"Checkout Process"**
- **"Verification of Prices" (create new folder)**
- **"Confirmation"**
- **"Verification of Success"**
- **"Postcondition"**

3. To the both occurrences of the **"Verification of Prices"** folder:

- Add the Module **"Confirm Order"** and rename the TestStep **"Verify Prices"**

4. To the **"Buffer Order Number"** folder:

- Add the Module **"Order Successful"**

5. To the **"Previous Orders"** folder:

- Add the Module **"Top Menu"** and rename the TestStep **"Navigate to my Account"**
- Add the Module **"My Account Menu"** and rename the TestStep **"Navigate to Orders"**

6. To the **"Reorder"** folder:

- Add the Module **"Order Details"**

7. In the **"Buffer Order Number"** TestStep folder, change the TestStepValue **"Order number"** to Verify and Buffer the Order Number from the property **"InnerText"**, Syntax for XBuffer is: **{XB[]}**

EXERCISES

Exercise 11b: Parent Control and Dynamic ID

Objective: Use XScan to create a Module with controls that have been identified by a Parent. Use XScan to create a Module with a dynamic ID.

Why is this important?

If the SUT has dynamic information, additional methods are required to fully automate.

Instructions

1. Duplicate TestCase older **"11a Dynamic Comparison"** and re name it **"11b Parent Control and Dynamic ID"**.

2. Navigate to the folder **"11b Orders Page"** in the Modules section (in Webshop>> Customer >> Myaccount).

3. Open the Web Shop application log in, and navigate to My Account >> Orders.

4. Use XScan to scan the Orders page. Make the following changes to the Module:

- Identify the first DIV which contains the order Information and the **"Details"** button
- Select this DIV the **"Details"** button and the LI that contains the order total
- Rename the DIV **"OrderInfo"** and the LI **"OrderTotal"**
- Uniquely identify **"OrderInfo"** by selecting the ClassName and the Outertext
- Identify the **"OrderTotal"** (LI) by its "**InnerText**"
- Rename the Module **"Order Overview"**
- Save the Module

❖ Remember - The DIV must be uniquely identifiable

5. Change the property **"OuterText"** for the attribute **"OrderInfo"** in the Module we first created to reflect the buffered order number from earlier in the TestCase. Ensure that all the other information in the outer text has been deleted and replaced with a wildcard **(*)**

6. To the **"Previous Orders"** TestStep folder:

Add the **"Order Overview"** Module rename it **"Navigate to Last Order"**

7. Enter the following Values for the TestStepValues as per the table below:

EXERCISES

TestStep	TestStepValue	Value	ActionMode
TestStep folder ***Previous Orders***			
Navigate to My Account	My Account	X	Input
Navigate to Orders	Orders	X	Input
Navigate to Last Order	Order Info		Select
	Details	X	Input
Reorder			
Order Details	Reorder	X	Input
TestStep folder ***Verification of prices***			
Verify Prices	Cart total		Select
	Column (rename #2)		Select
	Cell (Sub-Total: from drop down)	{MATH[{B[PriceBlueJeans]}*25]}	Verify
	Cell (Sub-Total: from drop down)	SubTotal	Buffer
	Cell (Shipping:* from drop down)	10.00	Verify
	Cell (Total: from drop down)	{MATH[{B[SubTotal]}+10]}	Verify

EXERCISES

8. Run the TestCase in the ScratchBook.

9. Set the TestCase WorkState to **“COMPLETED”**.

Hints

- Remember to add the correct Values in the Business Parameters.

EXPLICITNAME, RESOLVE REFERENCE, TBOX SET BUFFER

ExplicitName

- Renaming TestStepValues with an ExplicitName Configuration Parameter allows Tosca to steer using the index, and take into account any dynamic changes.

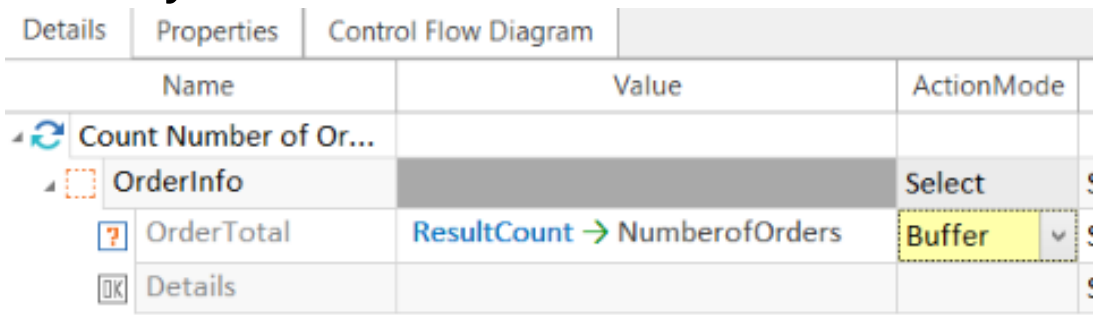
Resolve Reference

- You can resolve a testcase or test step which refers to a reusable test step block.

TBox Set Buffer

- Allows you to specify a Buffer value at the beginning of execution.

ResultCount → Counts how many of a certain control exists



EXERCISES

TEST CASE FIVE

Exercise 12a: ExplicitName

Objective: To be able to rename the Attributes of a Module with the use of an ExplicitName Configuration Parameter and use the property ResultCount within a TestStep.

Why is this important?

Renaming TestStepValues with an ExplicitName Configuration Parameter allows Tosca to steer using the index, and take into account any dynamic changes.

Instructions

1. Navigate to the Module folder **"11b Orders Page"** under Customer>>My Account.

2. Navigate to the properties of the **"Order Total"** Module Attribute of the **"Order Overview"** Module. Change the InnerText to identify any text containing **"Order Total:"** within the string.

3. Add the Configuration Parameter **"ExplicitName"** with the Value "True" for the control **"OrderInfo"**.

4. Navigate to the TestCases section, create a folder named **"TestCase 5"** and a subfolder named **"12a ExplicitName and ResultCount"**

- Add a TestCase named **"Total of All orders"**
- Add the Test Configuration Parameter named **"Browser"** with the value **"InternetExplorer"**
- Add the Reusable TestStepBlocks **"Precondition"** and **"Postcondition"**

❖ Resolve the reference to the ReusableTestStepBlock **"Postcondition"** Delete the TestStep **"Continue"**

5. Create a TestStep folder named **"Orders Page"** between the Precondition and Postcondition folders.

Create two TestSteps in the **"Orders Page"** folder, using the Modules:

- Top Menu – rename to **"Navigate to My Account"**
- My Account Menu – rename to **"Navigate to My Orders"**

EXERCISES

6. Enter the **Business Parameters.**

7. In the **"Navigate to My Account"** and **"Navigate to My Orders"** TestSteps, enter the correct Values.

8. Run the Precondition and Orders Page TestStep folders in the **ScratchBook** to open the Demo WebShop to the correct page

Exercise 12b: TBox Set Buffer and ResultCount

Objective: Use Tbox Set Buffer Module within a TestStep to set the Buffer **"SUM"** to the value **"0"**.

Why is this important?

When using the **"SUM"** Buffer, ensuring that it is at **"0"** prior to the TestCase running will prevent errors caused by existing Buffer values from previous TestCases still being present. Result count calculates how many controls with the same properties are on the page.

Instructions

1. Duplicate TestCase folder **"12a ExplicitName and ResultCount"** and name it **"12b TBox Set Buffer"**.

2. Create a TestStep folder named **"Set Buffer"** after the **"Orders Page"** folder. Add the **"TBox Set Buffer"** Module (Tricentis Standard Modules >> Tbox Automation Tools >> Buffer Operations >> TBox Set Buffer). Rename to **"Set SUM to Zero"**.

3. Change the **Buffer** names to:

- **"SUM"** - set the Value to 0
- **"OrderNumber"** - leave the Value blank

4. Add the **Module "Order overview"** – rename **"Count Number of Orders"**.

5. In the **"Count Number of Orders"** TestStep, enter the correct syntax to count the number of orders (number of OrderInfo containers) in the **Buffer "NumberofOrders"**.

Hint:

- The ResultCount property indicates how many of a certain control exist.

6. Run the TestStep folder **"Set Buffer"** in the **ScratchBook**. Check the buffer that the correct number of orders has been counted.

EXERCISES

Exercise 12c: Repetition on Folder Level

Objective: To instruct Tosca to repeat a TestStep a certain predefined number of times.

Why is this important?

Certain processes may need actions to be repeated a certain number of times. The number of repetitions can be set within this property.

Instructions

1. Duplicate the TestCase folder **"12b TBox Set Buffer"** and name it **"12c Repetition on folder Level"**.

2. Create a new TestStep folder named **"CountUp"**, between the **"Set Buffer"** and. **"Postcondition"**

TestStep folders.

3. Set the **"Repetition"** Property Value to the Buffer: **"NumberofOrders"**.

Hints

- Repetition can be found in the Properties tab of the TestStep folder.

Exercise 12d: ExplicitName on TestStepValue

Objective: Change the name of a TestStepValue to steer elements by index and dynamically.

Why is this important?

When elements do not have a unique identifier. ExplicitName allows Tosca to steer using the index, and take into account any dynamic changes.

Instructions

1. Duplicate the TestCase folder **"12c Repetition on folder Level"** and name it **"12d ExplicitName on TestStepValue"**.

2. Add the Module **"Order Overview"** into the **"Count Up"** TestStep folder. Rename it **"Buffer Order Total"**.

3. In the TestStep **"Buffer Order Total"**, locate the DIV **"Order Info"**. Rename it from **"Order Info"** to #{Repetition}. This will ensure that whichever repetition we are on, the corresponding DIV will be selected.

EXERCISES

4. In the **“CountUp”** folder, navigate to the **“Order Total”** control.

5. Use **{XB}** to verify and Buffer the price of the **“OrderTotal”**.

6. Add a TestStep to CountUp folder by adding the TBox Set Buffer Module: Tricentis Standard Modules >> TBox Automation Tools >> Buffer Operations >> TBox Set Buffer. Rename to **"Sum Up”**. Enter syntax for the Buffer **“SUM”** Value which will add the buffered **“OrderTotal”** amount to the existing Buffer **“SUM”**.

7. Log out of the Demo Web Shop and close the Browser. Run the TestCase in the **ScratchBook**.

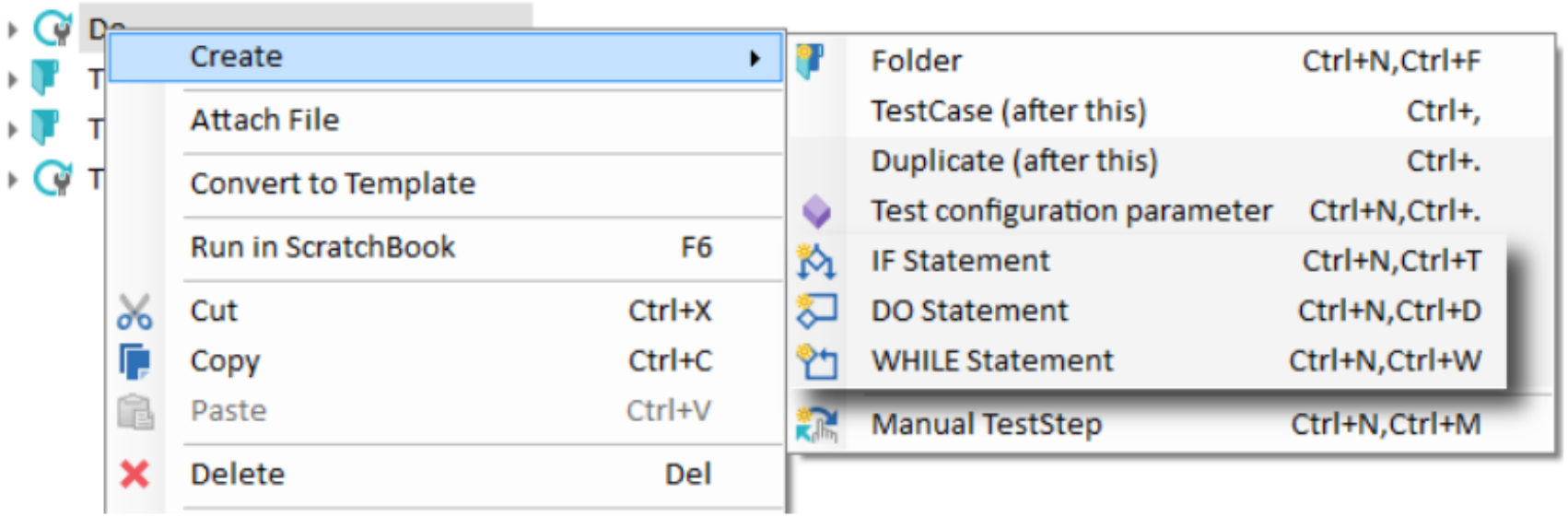
8. Set the **TestCase WorkState** to **“COMPLETED”**.

CONDITIONAL STATEMENTS AND LOOPS

You can define **IF**, **DO** and **WHILE** statements if you would like to run TestSteps with branches. These statements can be applied to any **nested structures**.

If the number of repetitions for a TestStep is known beforehand, use the property **Repetition** of a TestStepFolder as described in chapter "Running TestSteps repeatedly - Repetitions".

Conditional statements and loops can be created from the context menu of TestCases or TestStep folders:

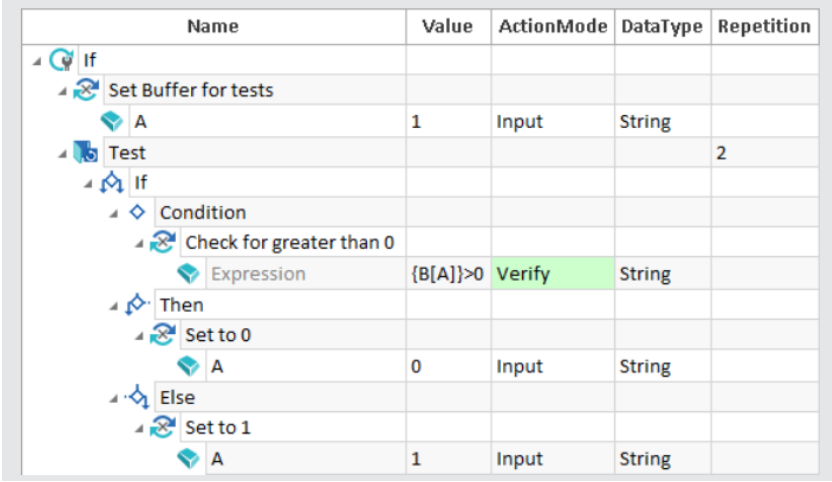


IF STATEMENT

Creating IF-THEN statements

When you create **IF** statements, Tosca will automatically create a **THEN** statement along with the Conditions object. You can also create an **ELSE** statement via the context menu of IF statements if required.

If the condition is fulfilled, the TestSteps are executed once in the THEN statement. If the condition is not fulfilled however, the TestSteps are run once in the ELSE statement.



Order of conditions:

- IF
- Condition (within the IF statement)
- THEN (within the IF statement)
- ELSE (optional)

For more details, see this sample for If Statement

IF STATEMENT

Name	Value	ActionMode	DataType	Repetition
If				
Set Buffer for tests				
A	1	Input	String	
Test				2
If				
Condition				
Check for greater than 0				
Expression	{B[A]}>0	Verify	String	
Then				
Set to 0				
A	0	Input	String	
Else				
Set to 1				
A	1	Input	String	

The test result looks as follows:

Name	Loginfo
Loops	1
If	
Set Buffer for tests	
A	Buffer with name: "A" has been set to value: "1"
Test	
Repetition: 1	
If	
Condition	
Check for greater than 0	
Expression	'1>0' evaluated to 'True'
Then	
Set to 0	
A	Buffer with name: "A" has been set to value: "0"
Repetition: 2	
If	
Condition	
Check for greater than 0	
Expression	'0>0' evaluated to 'False'
Else	
Set to 1	
A	Buffer with name: "A" has been set to value: "1"

DO STATEMENT

Creating DO statements

When you create **DO** statements, Tosca will automatically create a **Loop** object along with the **Conditions** object.

TestSteps within the **Loop** object are run repeatedly until the condition is no longer fulfilled.

Do statement contain the property **Maximum Repetitions** in order to avoid infinite loops

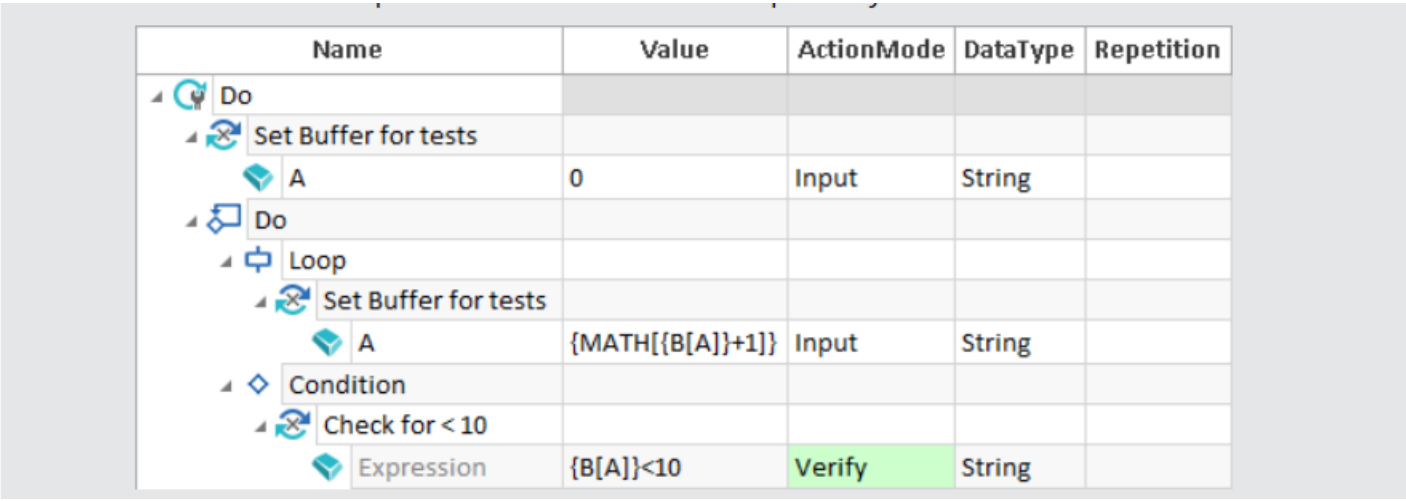
Name	Value
Do	
NodePath	/TestCases/Loops/Do
HasMissingReferences	False
UniqueId	-18112
DisabledDescription	
IsPausable	False
MaximumRepetitions	30

DO STATEMENT

Order of Conditions

- DO
- Loop (Within the DO statement)
- Condition (Within the DO statement)

Sample DO Statement



Name	Value	ActionMode	DataType	Repetition
Do				
Set Buffer for tests				
A	0	Input	String	
Do				
Loop				
Set Buffer for tests				
A	{MATH[{B[A]}+1]}	Input	String	
Condition				
Check for < 10				
Expression	{B[A]}<10	Verify	String	

DO STATEMENT

Name	Value	ActionMode	DataType	Repetition
Do				
Set Buffer for tests				
A	0	Input	String	
Do				
Loop				
Set Buffer for tests				
A	{MATH[{B[A]}+1]}	Input	String	
Condition				
Check for < 10				
Expression	{B[A]}<10	Verify	String	

The test result looks as follows:

Name	Loginfo
DO	1
Do	
Set Buffer for tests	
A	Buffer with name: "A" has been set to value: "0"
Do	
Repetition: 1	
Loop	
Set Buffer for tests	
Condition	
Check for < 10	
Expression	'1<10' evaluated to 'True'
Repetition: 2	
Repetition: 3	
Repetition: 4	
Repetition: 5	
Repetition: 6	
Repetition: 7	
Repetition: 8	
Repetition: 9	
Repetition: 10	
Loop	
Set Buffer for tests	
A	Buffer with name: "A" has been set to value: "10"
Condition	
Check for < 10	
Expression	'10<10' evaluated to 'False'

WHILE STATEMENT

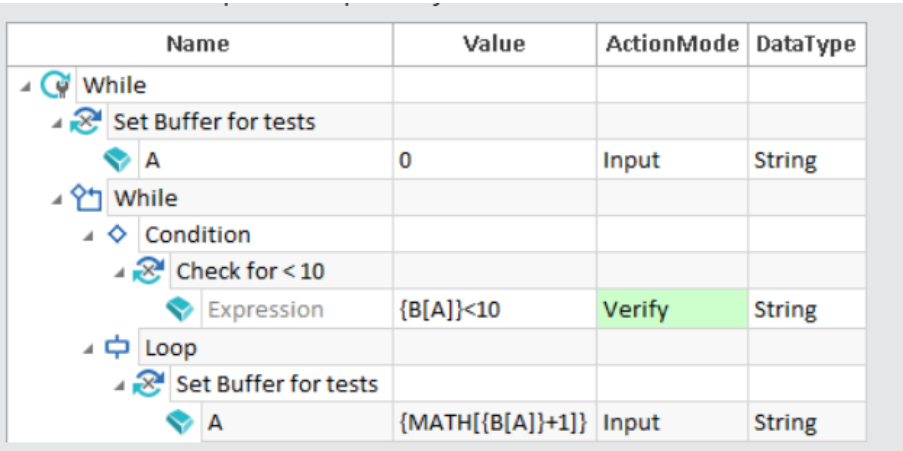
Creating WHILE statements

When you create **WHILE** statements, Tosca will automatically create a **Loop** object along with the **Conditions** object.

TestSteps within the **Loop** object are run repeatedly until the condition is no longer fulfilled. The result of the last repetition is negative

Do statement contain the property **Maximum Repetitions** in order to avoid infinite loops

Sample WHILE Statement



Name	Value	ActionMode	DataType
While			
Set Buffer for tests			
A	0	Input	String
While			
Condition			
Check for < 10			
Expression	{B[A]}<10	Verify	String
Loop			
Set Buffer for tests			
A	{MATH[{B[A]}+1]}	Input	String

WHILE STATEMENT

The test result looks as follows:

Name	Loginfo
WHILE	1
While	
Set Buffer for tests	
A	Buffer with name: "A" has been set to value: "0"
While	
Repetition: 1	
Condition	
Check for < 10	
Expression	'0<10' evaluated to 'True'
Loop	
Set Buffer for tests	
A	Buffer with name: "A" has been set to value: "1"
Repetition: 2	
Repetition: 3	
Repetition: 4	
Repetition: 5	
Repetition: 6	
Repetition: 7	
Repetition: 8	
Repetition: 9	
Repetition: 10	
Repetition: 11	
Condition	
Check for < 10	
Expression	'10<10' evaluated to 'False'

EXERCISES

Exercise 13: WHILE statement

Objective: Implement a WHILE statement within a TestStep.

Why is this important?

A **WHILE** statement allows the test case to execute an action repeatedly until a condition is no longer met.

Instructions

1. In the Library, create a Reusable TestStepBlock named **"Empty Shopping Cart"**; add the following Modules:

- **"Top Menu"**, rename it **"Navigate to Cart"**
- **"Top Menu"** again, rename it **"Log Out"**
- **"TBoxWindow Operation"**: Tricentis Standard Modules >> TBox Automation Tools >> BasicWindows Operations >> TBox Window Operation, rename it **"Close Web Shop"**

2. Within the TestStep folder **"Empty Shopping Cart"** enter the Values in the TestSteps that will perform the actions, as per the table below:

TestStep	TestStepValue	Value	ActionMode
Navigate to Cart	Shopping cart	X	Input
Log Out	Log out	X	Input
Close Web shop	Caption	Demo*	Input
	Operation	Close	Input

3. Within the Reusable TestStep Block **"Empty Shopping Cart"** create a **WHILE** statement between the TestSteps **"Navigate to Cart"** and **"Log Out"**.

EXERCISES

4. Within the WHILE Statement, add the Module **"Shopping Cart"** both in the Condition and in the Loop. Rename the TestStep in the Condition **"Verify Table Exists"**, and the TestStep in the Loop, **"EmptyCart"**. Add the Values to the TestSteps to complete the following actions:

- **Condition - "Shopping cart products"** table - Exists
- **Loop –** select the **"Remove"** checkbox within the **"Shopping cart products"** Table and then click the **"Update Shopping Cart"** button

Condition: TestStep	TestStepValue	Value	ActionMode
Verify Table Exists	Shopping cart products table	Exists == True	Verify

Loop: TestStep	TestStepValue	Value	ActionMode
Empty Cart	Shopping cart products table		Select
	Row: $1		Select
	Cell: Remove		Select
	Remove checkbox	True	Input
	Update Shopping cart	X	Input

5. Manually add multiple items to the Demo Web Shop then run the While statement in the **ScratchBook**.

Hints:

- Use the Control Flow Diagram to get a visual representation of the **WHILE** Statement.
- A Condition verifies that something meets the set state (exists, visible, has a value of etc.)
- A Loop instructs Tosca to steer a process, which then continues until the Condition is no longer met or the maximum number of repetitions is reached.

EXECUTION LIST

ExecutionListFolder 

- ExecutionList folders are used to structure and manage ExecutionLists.

ExecutionList 

- ExecutionLists offer a flexible possibilty to organize TestCases for repeated test execution.
- An ExecutionList contains links to TestCases, which need to be carried out.

ActualLog 

- Each ExecutionList contains at least one actual log.
- Both current and historical execution results are managed in the actual

EXECUTION LIST

ExecutionEntryFolder 

- ExecutionEntry folders are used to structure and manage execution entries.

ExecutionEntry 

- An ExecutionEntry represents a link between a TestCase and an ExecutionList.
- The latest execution result is graphically displayed at ExecutionEntry level.

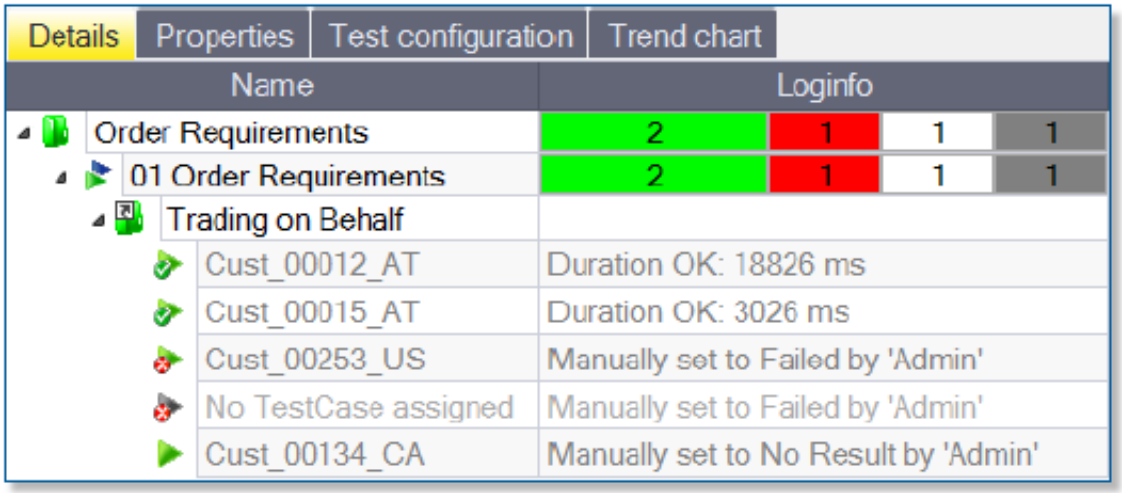
LogEntry 

- The historical execution results of an ExecutionEntry are managed in a log entry.

EXECUTION LIST

Execution Summary Displaying Test Results

- In all display versions, four colors are used for the clear display of test results:
  - A positively executed TestCase (no error) is displayed **green**.
  - A TestCase which results in an error is displayed **red**.
  - A TestCase which has not been executed is displayed **white**.
  - A TestCase, which is no longer available in the workspace is displayed **gray**.



DYNAMIC TEST CASE GENERATION

- The concept behind dynamic test case generation offers the possibility to manage TestCases separately from the test data contained within them.
- A **Template** is used as a guideline for the TestCase. It is a converted TestCase and can be converted back at any time.
  - **Template** Symbol 
- The test data can be managed in a **Microsoft® Excel Worksheet**.
- The **Template** and the **Microsoft® Excel Worksheet** are linked to one another and the test data combinations generate **TemplateInstances**.
- The **TemplateInstances** are physically available for execution.
- Changes in the Template and in the Microsoft® Excel Worksheet can be simply synchronized with TemplateInstances.

MICROSOFT® EXCEL AS A SOURCE OF DATA

- **Test case data** can be managed in Excel. This requires an Excel spreadsheet which arranges the test data in rows and columns.

	A	B	C	D	E	F	G	H	I
1	Object/Attribute				Domain	TC_Object_001	TC_Object_002	TC_Object_003	TC_Object_004
2	Object					Object_001	Object_002	Object_002	Object_001
3		height				11	14	14	11
4		width				12	15	15	12
5		depth				13	16	16	13
6	Material					MadeOf_001	MadeOf_002	MadeOf_001	MadeOf_002
7		color				green1	red1	red1	green1
8		material				wood1	iron1	wood1	iron1

- The **TestCases** are represented by **columns**, and individual **data sets** are represented by the **rows**.
  - The **columns A to D** contain objects and attributes. Data can be structured by using **indentations**. The names must be unique.
- Special characters e.g. dots » . « should not be used.

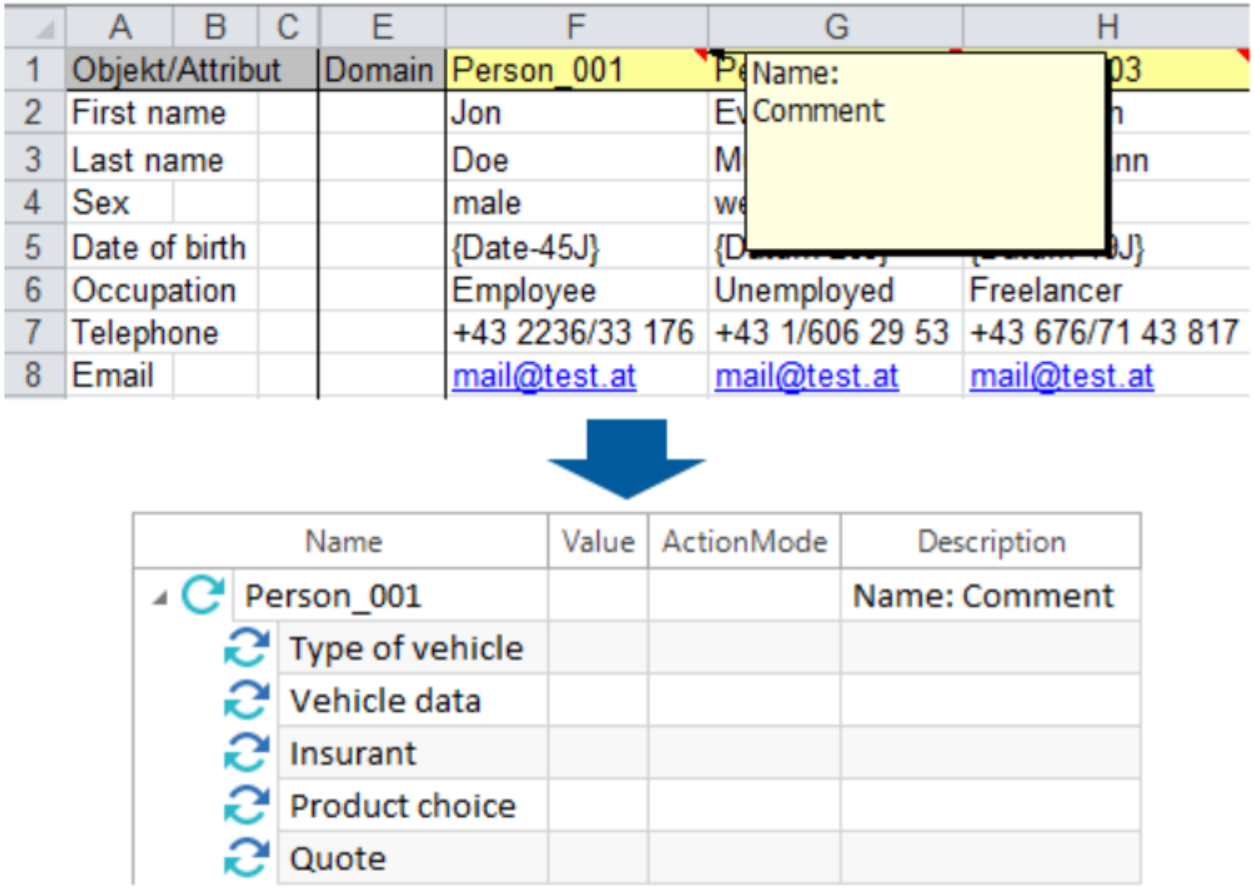
MICROSOFT® EXCEL AS A SOURCE OF DATA

	A	B	C	D	E	F	G	H	I
1	Object/Attribute				Domain	TC_Object_001	TC_Object_002	TC_Object_003	TC_Object_004
2	Object					Object_001	Object_002	Object_002	Object_001
3		height				11	14	14	11
4		width				12	15	15	12
5		depth				13	16	16	13
6	Material					MadeOf_001	MadeOf_002	MadeOf_001	MadeOf_002
7		color				green1	red1	red1	green1
8		material				wood1	iron1	wood1	iron1

- From **column F** onwards, the **headline contains the business objects**, also referred to as instances. Tosca Commander™ converts them to **TestCase names**.
  - The rows are filled with data sets.
- The background colors must be maintained.
- Only **one Excel Sheet** can be used for every **one TestCase Template**
- A sample Excel template to be used with Tosca can be found in the **%TRICENTIS_PROJECTS%\ToscaCommander\Templates\BO** directory.

MICROSOFT® EXCEL AS A SOURCE OF DATA

In Excel you can define comments for business objects. When TestCases are instantiated, these comments are copied to the template instance and shown in the **Description** column.



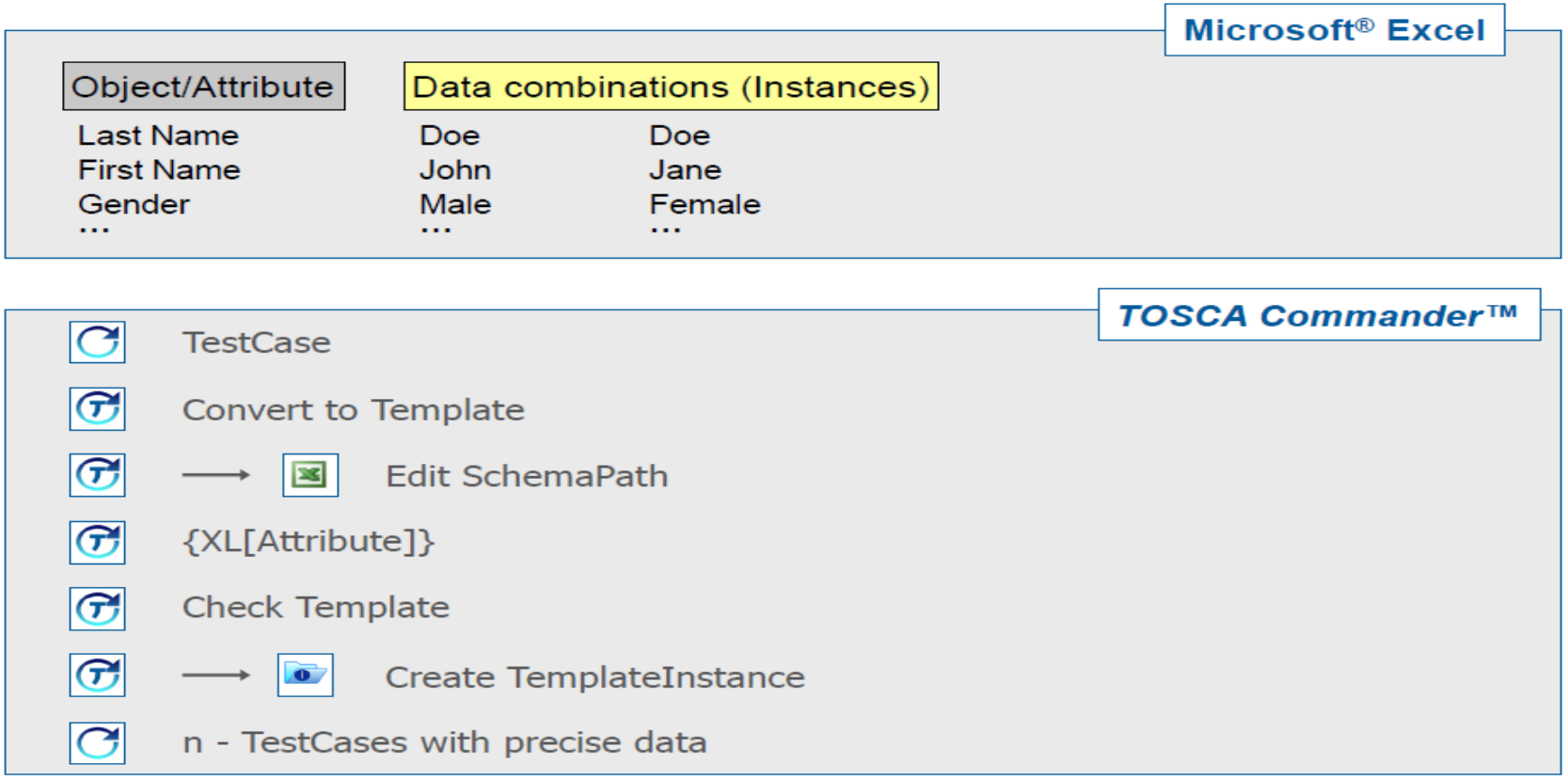
	A	B	C	E	F	G	H
1	Objekt/Attribut			Domain	Person_001	Pe	03
2	First name				Jon	Ev	n
3	Last name				Doe	M	nn
4	Sex				male	we	
5	Date of birth				{Date-45J}	{D	0J}
6	Occupation				Employee	Unemployed	Freelancer
7	Telephone				+43 2236/33 176	+43 1/606 29 53	+43 676/71 43 817
8	Email				mail@test.at	mail@test.at	mail@test.at

Name:
Comment

Name	Value	ActionMode	Description
Person_001			Name: Comment
Type of vehicle			
Vehicle data			
Insurant			
Product choice			
Quote			

MICROSOFT® EXCEL AS A SOURCE OF DATA

Dynamic Test Case Generation Sequence.



EXERCISES

Exercise 14: Excel as a Source of Data

Objective: Execute a simple login-logout scenario with different UserNames/Email using a Microsoft® Excel Worksheet as the source of data.

Instructions

1. Create a TestCase folder **“Excel Test”** and then create a TestCase under it **“Login Logout Smoke Test”**.
2. Drag and Drop the ‘Precondition’ and ‘Postcondition’ folders from the Library to your newly created TestCase
3. Right click on the Precondition in your newly created testcase and select ‘Resolve Reference’ from the context menu.
4. Delete the values of the ‘Email’ and ‘Password’ in the Login TestStep under Precondition.
5. Create an Excel file using the following template:

	A	B	C	D	E	F	G
1	Object/Attribute				Domain	TC_001	TC_002
2	UserName						
3	Password						

EXERCISES

Note: Microsoft® Excel templates can be found in the %TRICENTIS_PROJECTS%\ToscaCommander\Templates\BO directory.

6. Fill-in the cells F2 to G3 with the required data, as in the example below:

	A	B	C	D	E	F	G
1	Object/Attribute				Domain	TC_001	TC_002
2	UserName					Jake_Cobb@gmail.test	Ben_Joe@yahoo.test
3	Password					Tosca1234!	Tosca4321!

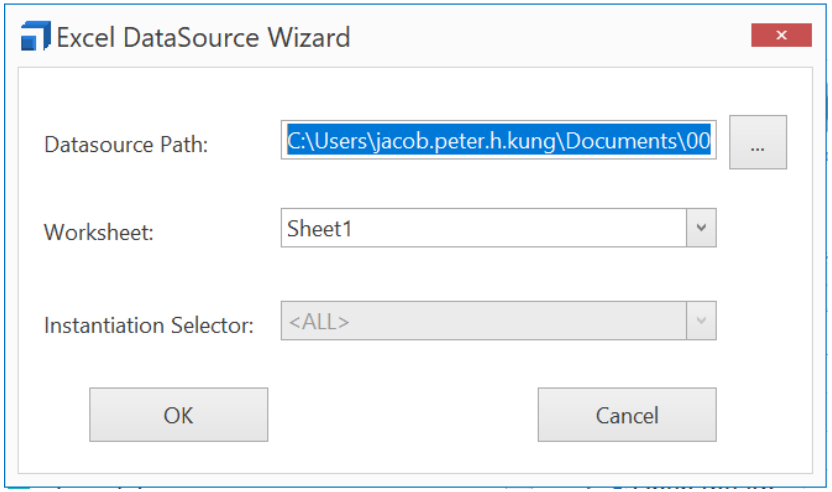
7. Save you file and take note of the Directory path used.
8. Right click on the TestCase and select ‘Convert to Template’ from the context menu.
9. In the Login TestStep under Precondition, enter the following values for the Email and Password:

Name	Value
Log in	
Email	{XL[UserName]}
Password	{XL[Password]}

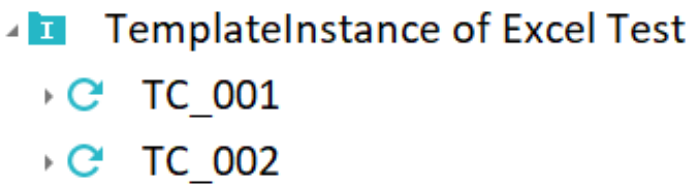
10. Right click on the TestCase and click the Create TemplateInstance icon (Ctrl+N, Ctrl+I). The **Excel DataSource Wizard** pop-up window should be displayed.

*Use the ‘Check Template’ first in the context menu to be sure there are no errors

EXERCISES



11. Enter Directory path you used for your file in the Datasource Path field (you can also use the button beside it to browse and search for your file).
    e.g. C:\Users\<e.i.d>\Documents\<filename>
12. Click the OK button. TestCase Instances should be created in the TemplateInstance folder based on the data you created in the Excel file.



13. Run the TemplateInstance and change the TestCase WorkState to “COMPLETED” once successful.

TEST CASE DESIGN - TESTSHEETS

- The **TestCase Template** can also be linked to a **TestSheet** (instead of an Excel Worksheet) to generate **TemplateInstances**.

TestSheet

- basic framework where test cases (combinations of variants I can enter into an application) for testing a functionality are specified

Attribute - features of a specific person or thing

Instance - features of a specific person or thing

*This will be covered in more detail in succeeding Advanced Tosca Courses

SPECIAL CHARACTERS

- Values are often used in test specification, which are not generated until such time as the test is executed.
  - **Time-dependent** values (e.g. today‘s date, insurance policy start date …)
  - **Item-dependent** values (e.g. transaction number, customer number, …)

- **Special** characters and **dynamic** values are displayed in the following format in *TOSCA Commander*™:
  - {<Syntax>} and/or. {<Command>[<Parameter>]}

- **Precise** values, special characters and dynamic values can be combined with one another.
  - e.g. the input of a value and the confirmation with the return button

- The availability of the special character depends on the individual engine.

SPECIAL CHARACTERS

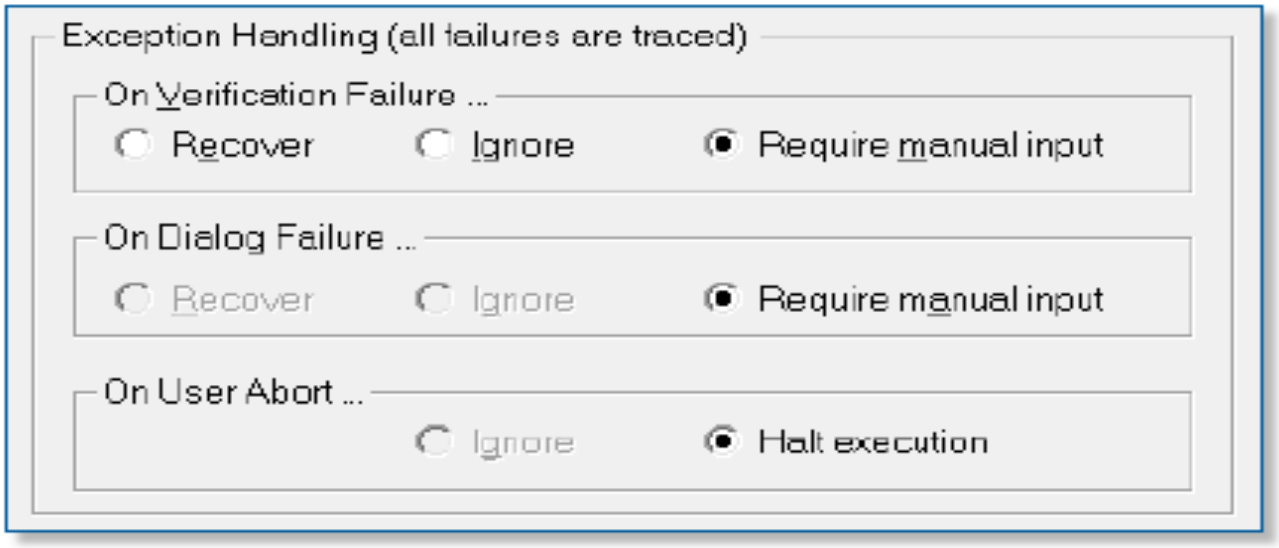
Examples

- Valid characters in HTML

Special character	Result	
{ENTER}	A click of the enter key will be sent to the system under test.	
{F1}	A click of the F1 key will be sent to the system under test.	
{RND[8]}	73920312	
123,45{INT[+/-6,7]}	Valid value intervals: 116,75 through 130,15	
{CALC[750.10-49+0.9]}	702	
Your order no. {XB[OrderID]} has been shipped.	Your order no. has been shipped.	OrderID: 6354

TOSCA EXECUTOR | EXCEPTION HANDLING

- TOSCA Executor offers flexible exception handling possibilities during test execution.
- There are three different types of errors in **exception handling:**
  - **Verification-Failure**: Verification did not yield the expected result
  - **Dialog-Failure**: The operation of a control is not possible
  - **User Abort:** Test execution has been interrupted by the user



TOSCA EXECUTOR | EXCEPTION HANDLING

The response to errors can be specifically set for each project. For every situation, a log will be produced documenting any errors.

It‘s possible to define at runtime how each of these error types is handled:

- **Recover**: Test execution is set to continue without error dialog
- **Ignore**: Test execution continues without error dialog
- **Require manual Input**: Test execution is set to abort with error dialog

An **execution status icon** is displayed in the info area of the Windows taskbar during test execution.

- The **execution status icon**

EXECUTION SETTINGS | ERROR MESSAGES

Should *TOSCA Commander*™ not react to the inputs, one option is to wait for the execution timeout to be displayed.

A frequent error message:

- **Unable to locate an object with ID …** (TOSCA Executor cannot steer a specific control)

Possible causes:

- The control doesn't exist on the screen that is being steered
- The screen in question cannot be steered
- The technical ID of the control has been changed

MANUAL TEST EXECUTION

TOSCA Testsuite™ offers the possibilty of manually processing test cases as desired (in the context of execution lists).

- Command **Run as Manual Testcase**

Automated and manual test steps can be manually worked with the aid of a **checklist**..

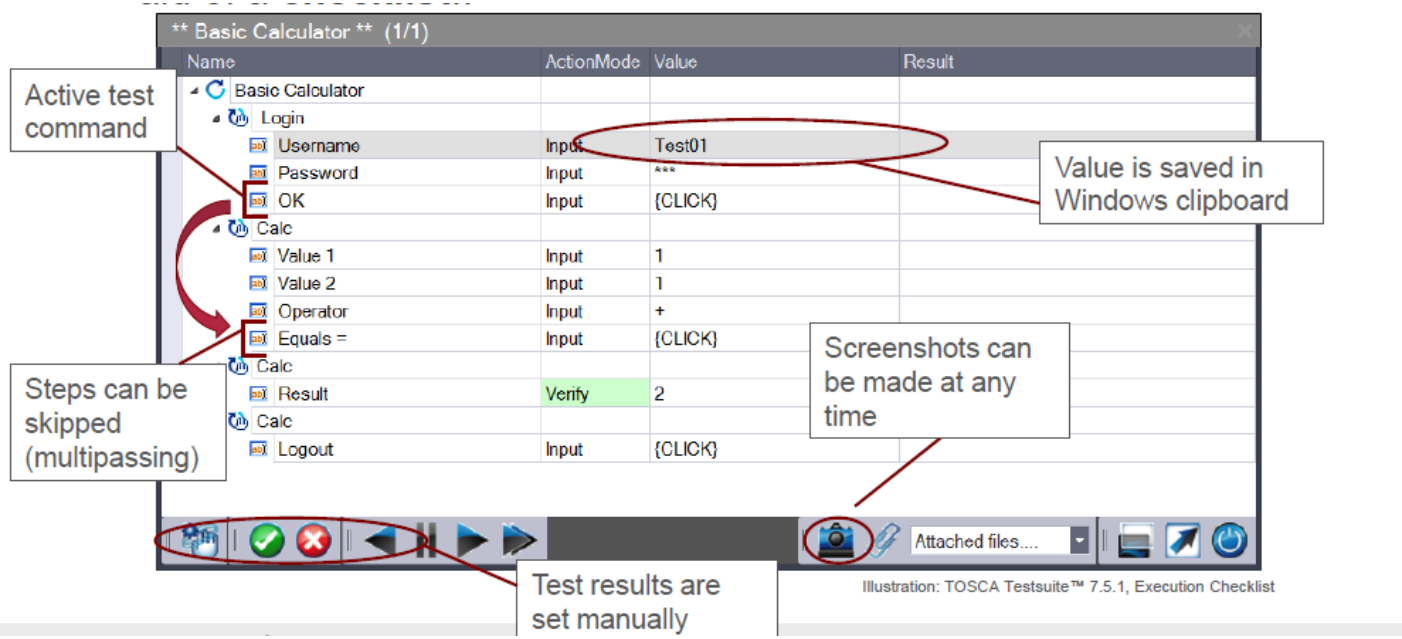
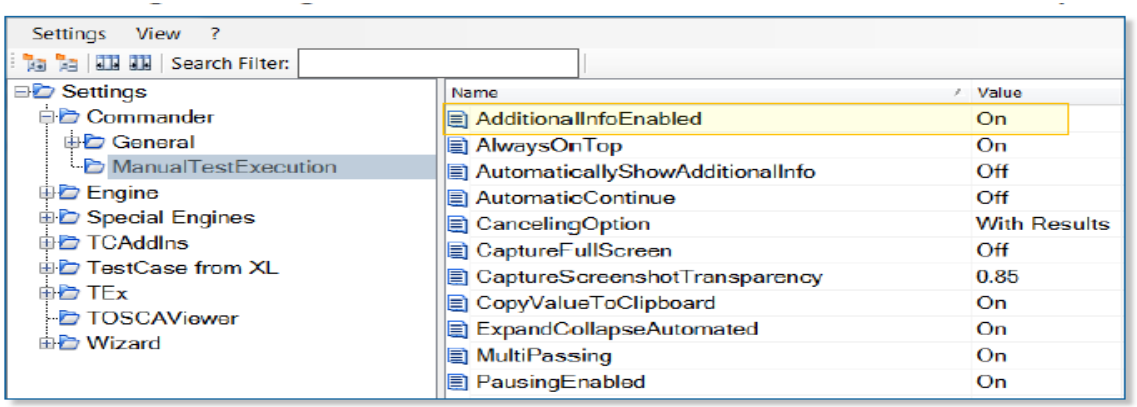


Illustration: TOSCA Testsuite™ 7.5.1, Execution Checklist

MANUAL TEST EXECUTION

- The following settings are recommended based on best practices:



AutomaticallyShowAdditionalInfo - On

- During test execution additional information about the current test description is shown.

MULTIUSER WORKSPACE

- Several users can simultaneously work on a project in the Multiuser Environment in *TOSCA Commander*™.
- The project is initially created by an administrator in a **Common Repository** and made available to all users.
  - The Common Repository is where the data is centrally stored
- Authorized users on different workstations have the possibility of working on different objects in the Common Repository at the same time.
- Can be managed to determine that a single object can only be worked Access on by a single user at a certain time.
  - The handling of an object is always carried out in a Local Repository
- The mechanisms for managing access to data are **Check Out, Check out tree, Check in all** and **Update all.**

MULTIUSER WORKSPACE | MECHANISM

CheckOut, CheckOut Tree (read-and-write access) 

- The selected group of objects in common repository is exclusively reserved for the current user.

CheckIn All (adopting modifications) 

- All objects that are new and checked out by the current user are checked in to the common repository.

Update All (synchronization) 

- All modifications by other users that are checked in are transferred to the local workspace and displayed.

MULTIUSER WORKSPACE | MECHANISM

Status of objects in the Multiuser workspace:

- Newly created object
- Object checked out by another user
- Object checked out by the user
- Object excluded from synchronization
- Objects which have the status *checked in* in the Common Repository are grayed out in the Multiuser workspace.

OPTIONS

- *TOSCA Commander*™ specific settings can be configured by going to **Options** in the menu bar.
  - Menu: Tools -> Options...
- The settings displayed here are locally valid for the presently active instance of *TOSCA Commander*™.

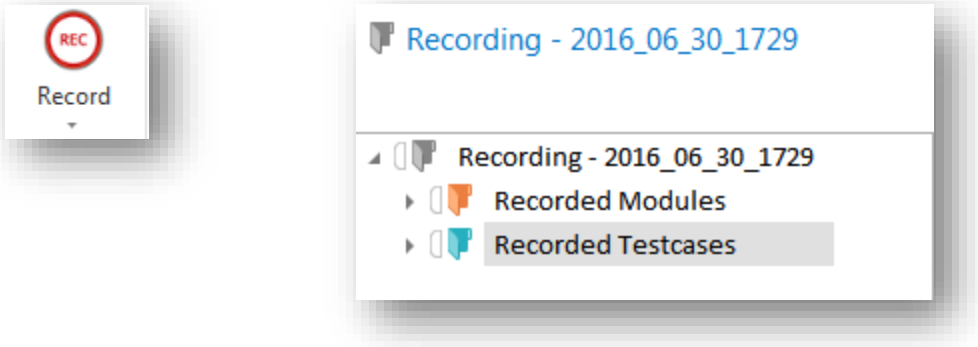
SETTINGS

- *TOSCA Testsuite*™ supports the administration of various test projects with different configuration requirements.
- TOSCA Engine specific settings can be configured by going to **Settings** in the menu bar.
  - Menu: Tools -> Settings -> Engine
  - Menu: Tools -> Settings -> Special Engines
- TOSCA Wizard specific settings can be configured by going to **Settings** in the menu bar.
  - Menu: Tools -> Settings -> Wizard
- Settings are described in more detail in the *TOSCA Testsuite*™ manuals:
  - Engine specific settings can be found in the engines manuals.
  - General settings are described in the *TOSCA Commander*™ manual.

RECORDING

Tricentis' Tosca Recorder

Allows you to just **record your actions** and it will automatically create and easily execute your test cases. The best part is that Tosca Testsuite **automatically recognizes previously recorded controls** and is able to re-use your test assets, meaning redundancy-free test cases.



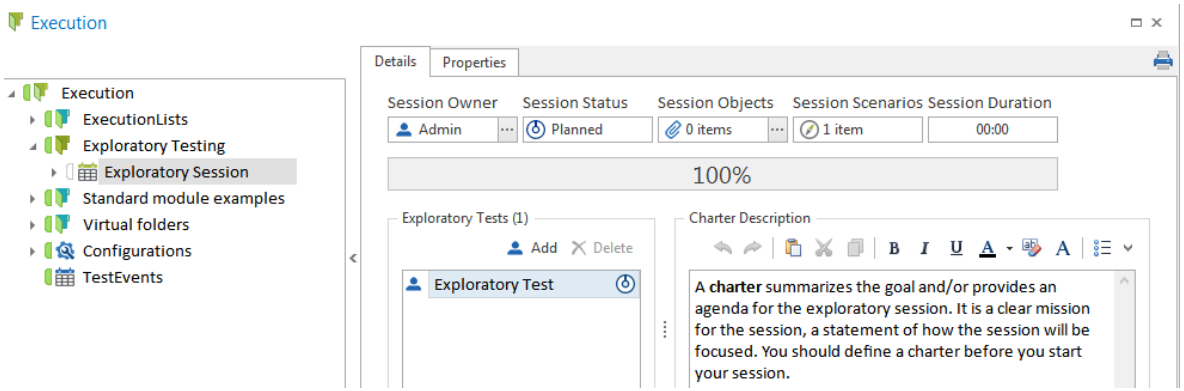
EXPLORATORY TESTING

Exploratory Testing

Testers create scenarios, they make use of videos, screenshots and steps (TestSteps) to document both the scenarios and any found errors in the test object.

Creating exploratory sessions.

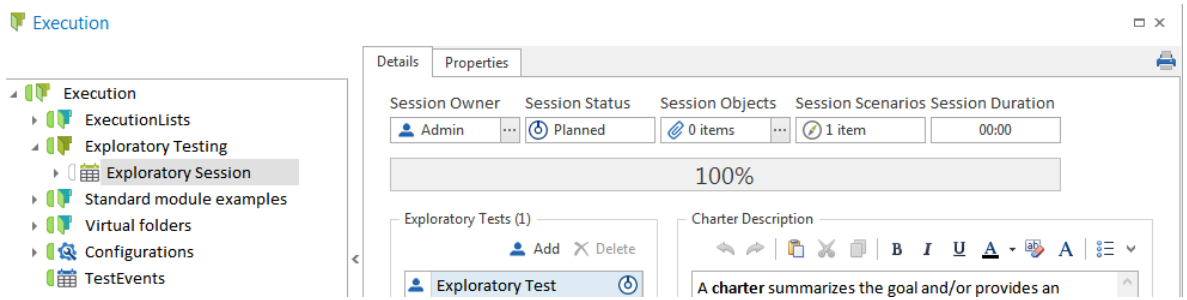
The session owner plans exploratory sessions for the testers to participate. He/She describes the test objectives in the Charter. The exploratory test results, the test status and duration are summed up in the session.



EXPLORATORY TESTING

- **Session Owner**: Define the session owner either manually, or select a user from the drop-down list in multiuser workspaces.
- **Session Status**: The session status is automatically determined (**Planned**, **In Progress**, **Completed**). If the status of all exploratory tests is set to Completed, the session status is also set to Completed.
- **Session Objects**: Use drag and drop to move a TestCase to this field in order to use this as a scenario template. You can also drag any objects and files that are relevant for testing to this field.
- **Session Scenarios**: The number of exploratory test scenarios of the session.
- **Session Duration**: Overall duration of all exploratory tests in this session.

The row below the Session owner field shows the results of the exploratory tests in relation to each other. A test is classified as successful if all test scenarios are successful. If at least one scenario result is set to failed, then this test is classified as failed.



EXPLORATORY TESTING

Creating and editing exploratory tests

- Exploratory tests are created for each tester.
- In exploratory testing you capture scenarios, you write a test summary, and you may receive further instructions from the session owner.

Steps:

1. Select the required Exploratory Session

2. Click on Add in the Exploratory Tests section of the exploratory session to create and open a new exploratory test.

EXPLORATORY TESTING

Tester: Define the tester either manually, or select a tester from the drop-down list in multiuser workspaces.

Test Status: Specify the test status (Planned, In Progress, Completed).

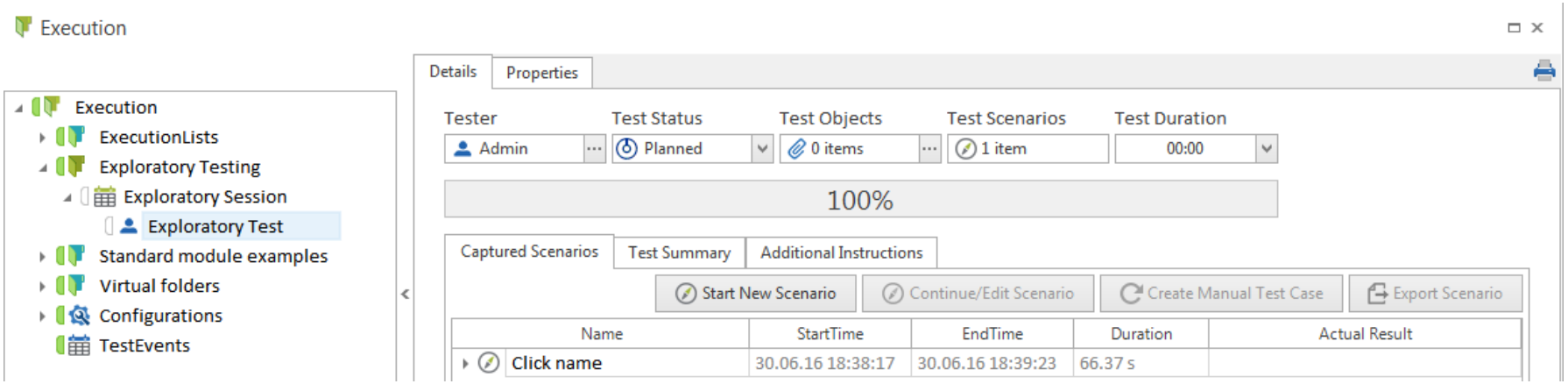
Test Objects: Use drag and drop to move a TestCase to this field in order to use this as a scenario template. You can also drag any objects and files that are relevant for testing to this field

Test Scenarios: The number of test scenarios.

Test Duration: Enter the overall duration of the test here.

EXPLORATORY TESTING

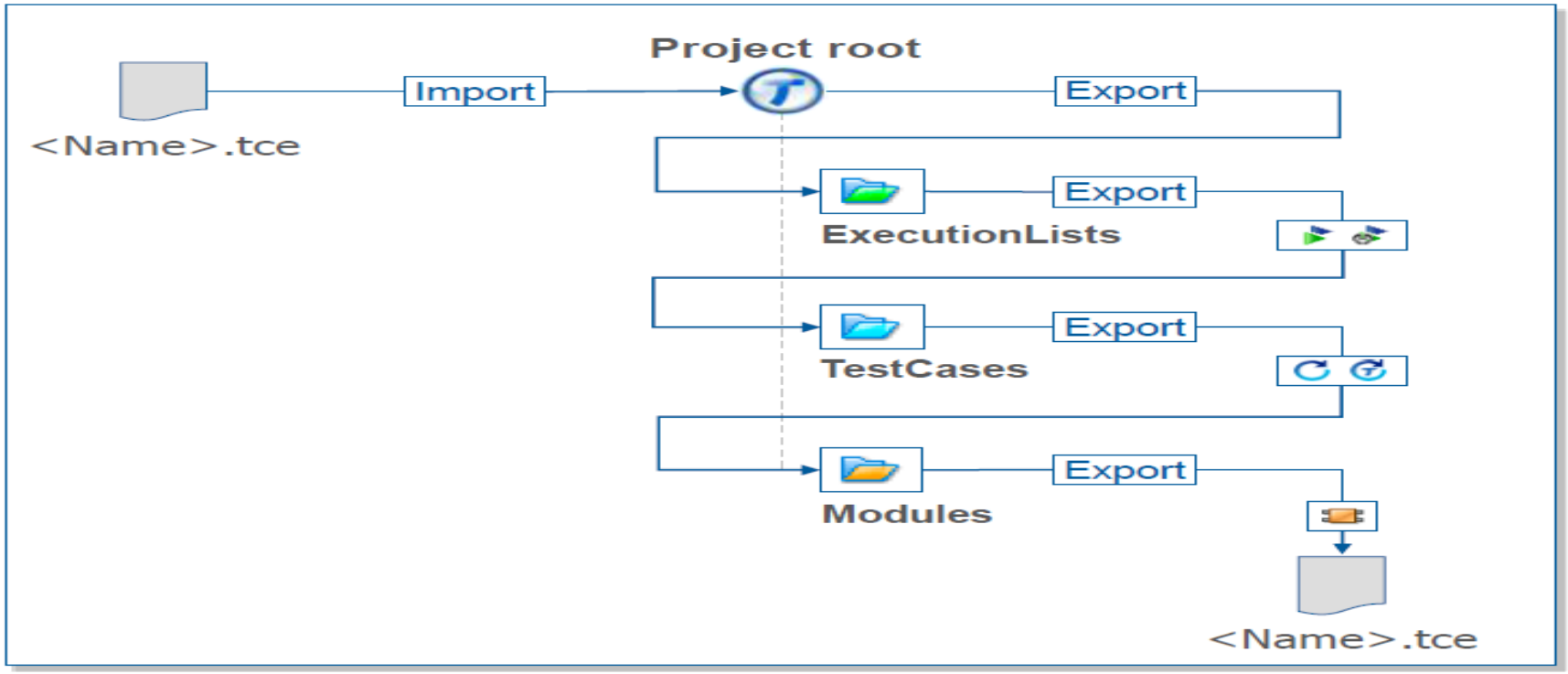
The row below the Tester field indicates the scenario results in relation to each other. A scenario is classified as successful if **all scenario steps** or the scenario itself are **successful**. If at least one scenario result is set to failed, then the entire scenario is classified as failed.



EXPORT AND IMPORT OF SUBSETS

- All objects in *TOSCA Commander*™ can be exported in subsets. (using the command **Export Subset**).
- Subsets are compressed data files in XML format, which are saved with the file extension **<name>.tce**.
- *TOSCA Commander*™ automatically exports all objects which are necessary to restore the selected objects in another project (using the command **Export Subset**).
- The **import** of a subset in *TOSCA Commander*™ is done through the context menu of the **project root element**.
- The import process does not overwrite any existing data - it adds data to a separate folder in the project instead..
- Subsets allows unrestricted data transfer between Singleuser und Multiuser projects.
- The use of subsets is possible for all user groups.

EXPORT AND IMPORT OF SUBSETS



KEYBOARD SHORTCUTS

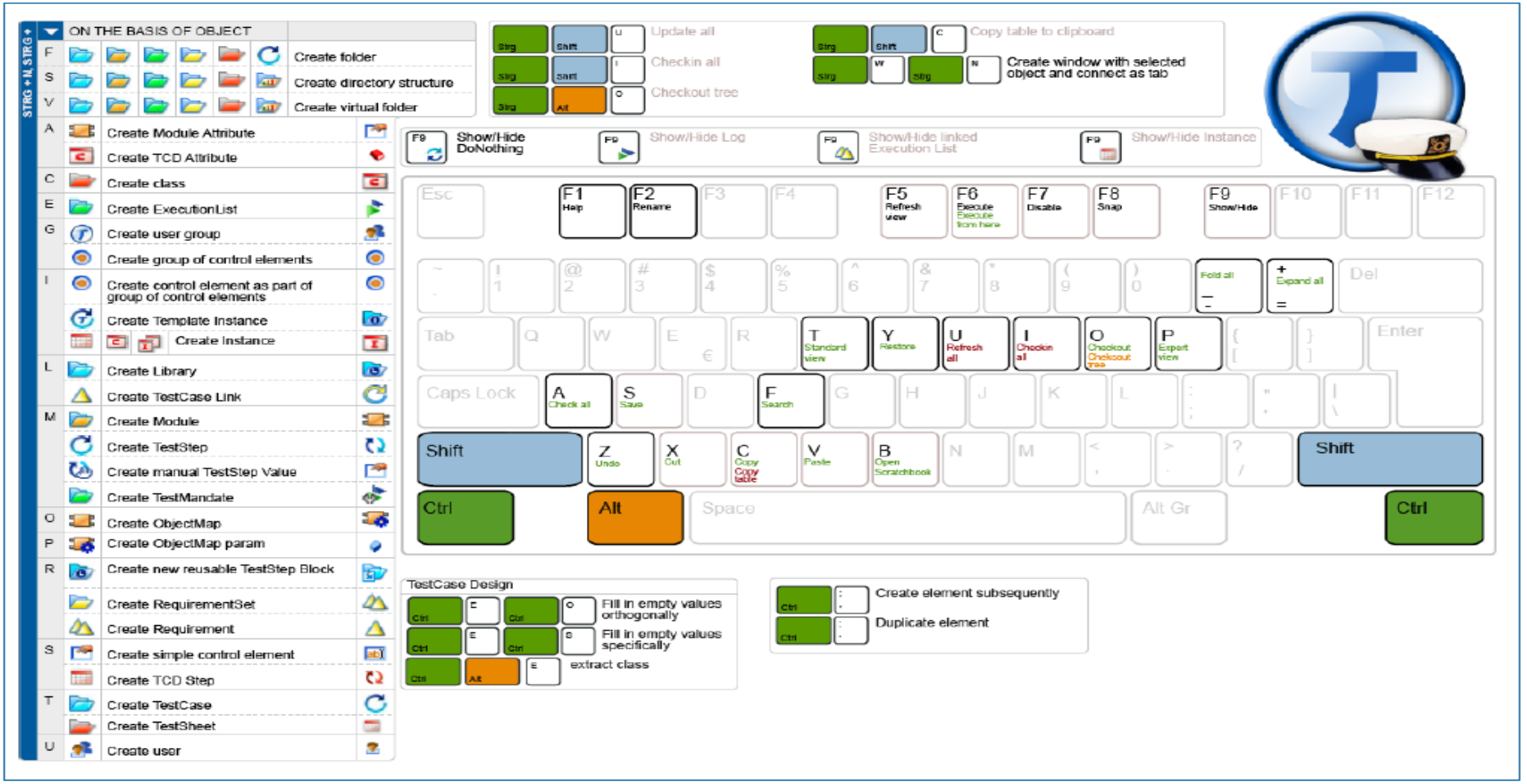
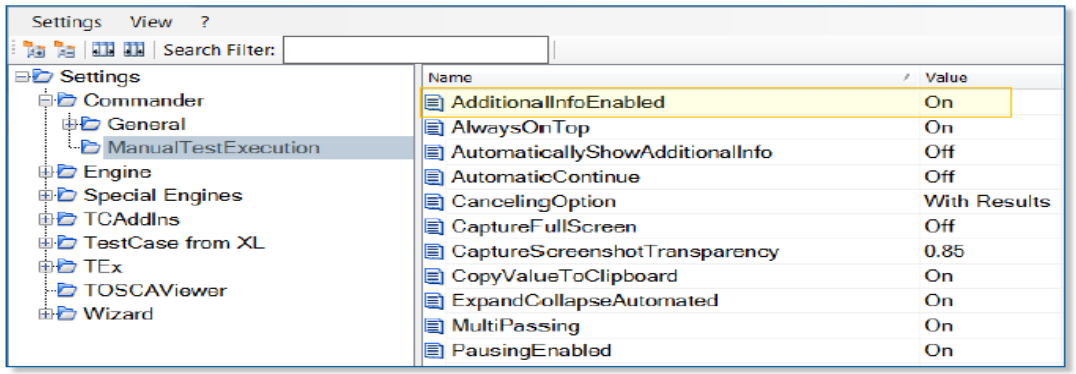


Illustration | Keyboard shortcuts TOSCA Testsuite™, english keyboard layout

BEST PRACTICES

- The following settings are recommended based on best practices:



- **AutomaticallyShowAdditionalInfo -** On
  - During test execution additional information about the current test description is shown.

TOSCA WIZARD

- The **TOSCA Wizard** provides the technical information required to identify the controls of a system under test and to steer it.
- There are two ways to start the TOSCA Wizard:
  - Using the command **Start Wizard** in the context menu of a folder in the Modules area (recommended).
  - Using the **start menu** of Microsoft® Windows.
- Menu item View
  - **Hide window during capturing**
    - The TOSCA Wizard window is hidden during the scanning process
  - **Show application info**
    - The technical attributes of the system under test at hand are recorded and displayed
  - **Show only selected items**
    - Only selected controls will be displayed

TOSCA WIZARD

- **Current filter**
  - The display can be limited to individual control types

Menu item Options

- **Scanning -> Scan all Items**
  - TOSCA Wizard scans all controls and offers the possibility for them to be stored in an ObjectMap
- **Scanning -> Enable hotkey**
  - The scan process is started through use of a single key on the keyboard. This can be defined by selecting Project -> Edit Settings in the menu bar
- **Use UIAEngine**
  - An alternative steering option
  - The UIAEngine is used for scanning an ObjectMap and for steering - even if this could be done using another engine

ICONS – TEST CASES

TestCases	
	TestCase folder
	TestCase
	Business TestCase
	TestStep folder
	TestStep, XTestStep
	Manual TestStep
	TestCase Templates, Business TestCase Templates
	TemplateInstance

	TestStep disabled
	TestStep folder disabled
	Manual TestStepValue
	Table
	Input, ControlSimple
	Select, ComboBox
	Button
	RadioButton
	A (Label), HTML Link
	CustomControl

	TreeIcon
	XTestStepValue
	Missing reference
	Property: standard, user-defined
	Folder structure
	Virtual Folder
	TestStep Library
	Reusable TestStepBlock
	Reusable TestStepBlock Reference
	TestCase Planned
	TestCase In Work
	Test Passed
	Test Failed

ICONS – MODULES

Modules	
	Module folder
	Module, XModule
	ModuleAttribute
	Table
	Input, ControlSimple
	Select, ComboBox
	Button
	RadioButton
	A (Label), HTML Link
	CustomControl
	TreeIcon
	Import ObjectMap
	Delete ObjectMap
	Open new Module Window

ICONS – MULTI USER WORKSPACE

Multiuser Workspace	
	Element checked out
	Element checked out by different user
	New element
	Excluded element
	Excluded folder
	UserGroup
	User
	Checkout tree, Checkout
	Update all
	Checkin all

ICONS – EXECUTION

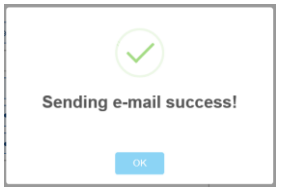
Execution	
	Execution folder
	ExecutionList folder
	ExecutionList
	Business ExecutionList
	TestMandate
	ExecutionEntry
	Business ExecutionEntry
	ExecutionLog (ActualLog)
	TestCase log (Execution TestCaseLog)
	TestStep log (Execution TestStepLog)
	TestStepValue log (Execution TestStepValueLog)
	ExecutionEntry disabled
	ExecutionEntry folder disabled
	Synchronous ExecutionEntry folder
	AutoMerge
	Import TestResult
	ExecutionEntry - TestCaseWorkState Test planned
	ExecutionEntry - TestCaseWorkState Test In Work
	Test Passed
	Test Failed
	Exploratory Testing folder
	Exploratory Session
	Exploratory Test
	Configuration
	TestEvent

VERSION HISTORY

Created Date	Revised By	Version	Description of Changes
2/29/2016	Rowell Gie Malong	1.0	Document Creation
5/24/2016	Joseph Evardome	1.1	Expanded Modules and Test Case chapters
6/30/2016	Duane Gayman	1.3	Reorganized the curriculum and updated additional information for the training based on the updated manual.
03/12/2018	Duane Gayman	2.0	Updated the deck with Tosca version 11 functions
04/20/2018	Benjie Colastre	2.1	Updated based on Internal Review (icons, bold text, etc.)
04/26/2018	Jacob Kung	2.2	Updated based on Additional Internal Review (layout, template, copyright, etc.)
05/07/2018	Benjie Colastre, Jacob Kung	2.3	Updated the Exercises and the ‘Microsoft® Excel as a source of data’ section

EXAM

- Open http://sampleapp.tricentis.com/101/
- Create 2 automated Test Cases with different test data inputs using TOSCA Commander for a Automobile insurance quote inquiry.
- Automated Test Case should have the following:
  - Use of Standard Modules (Tbox)
  - Use Self Defined TCP
  - Use Library
  - Use WaitOn
  - Random Values:
    - Plate number should have 3 alphabets followed by 3 numbers {RANDOMREGEX[]}
    - Date of birth should be a date greater or equal to 18 years in the past
      - Use random numbers to calculate for the date
  - Verify the submit successful message after quote inquiry submission



  - {XB[<variable name]} → At the 'Enter Vehicle Data Page', notice the number after "Enter Vehicle Data" which is "7". Buffer this control then populate the required fields. Use MATH and {B[]} to verify that the control is 0 required fields are already populated.