TOSCA Automation Tool Training

Participants / Instructors Introduction

- Name
- Project
- Level / Years with Accenture
- Thoughts on Test Automation

Objectives

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

- Identify the TOSCA Test Suite and TOSCA Commander and it’s application in Testing Automation.
- Identify and define the different components of TOSCA.
- Create Modules, Test Cases and Execution List using the TOSCA Commander.

Course Outline

Day 1	Day 2	Day 3
I Introduction / Installation Product Overview Business Dynamic Steering TOSCA Test Suite Components Requirements Testcase Design Cognitive TestCase Execution List Reporting	**VI ExecutionList** **VII TestStepLibrary** **VIII Dynamic TestCase Generation** Template Microsoft Excel as source of data TemplateInstances Conditional instantation	
Break	**Break**	
II TOSCA Commander Elements Workspaces Module Test Case	**IX Special characters** **X Dynamic values** **XI Manual test execution** **XII Multiuser workspace**	**Case Study**
Lunch	**Lunch**	
III Control **IV ActionModes** **V Table Steering** **VI Conditional Statements**	**XII Additional Information** Recording Exploratory Testing Export and Import Subset Best Practices Advanced Training	

accenture

Introduction - Product Overview

- Tricentis 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**™.
- **Tosca XScan** scans 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).

accenture

Introduction - TOSCA: Components

TOSCA Test Suite (Components)

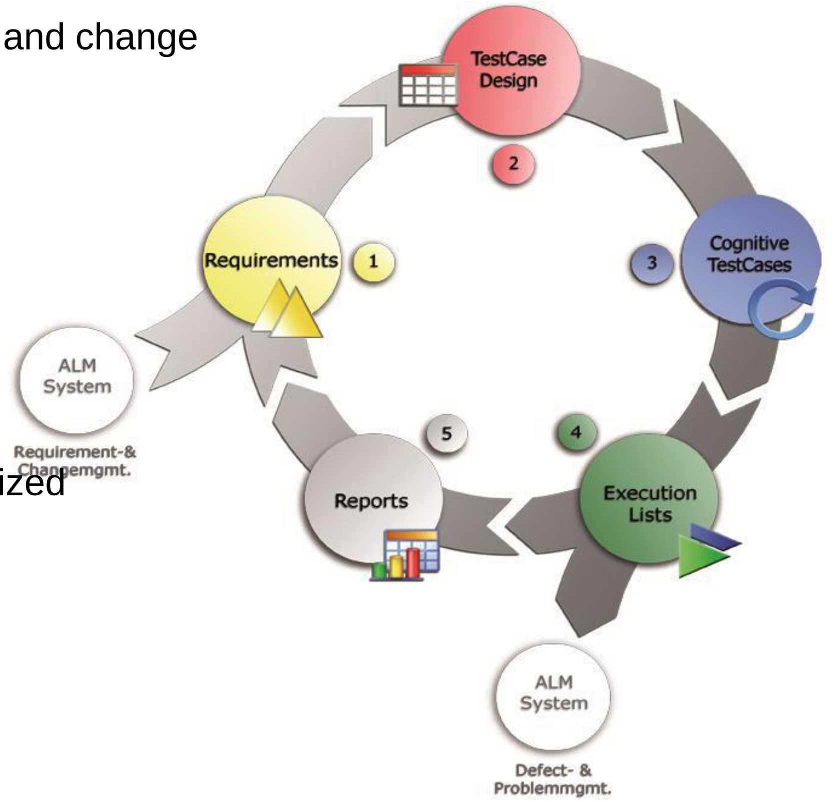
- Requirements
  - These requirements are both the starting and the reference point in software development
  - The test results are projected onto the **Requirements** in order to map the coverage from a test point of view.
- TestCase-Design
  - Business-related TestCases are collected in here.
- TestCases
  - Series of TestSteps that run from a defined starting point to a defined ending point.

Icon	Description
📁	TestCase folder
🔄	TestCase
🔄 🔄	TestStep, XTestStep

TestStepValue

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



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.

Module Folder

Module Attribute

Module

- 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**.

Actual Log

ExecutionListFolder

ExecutionEntry

ExecutionList

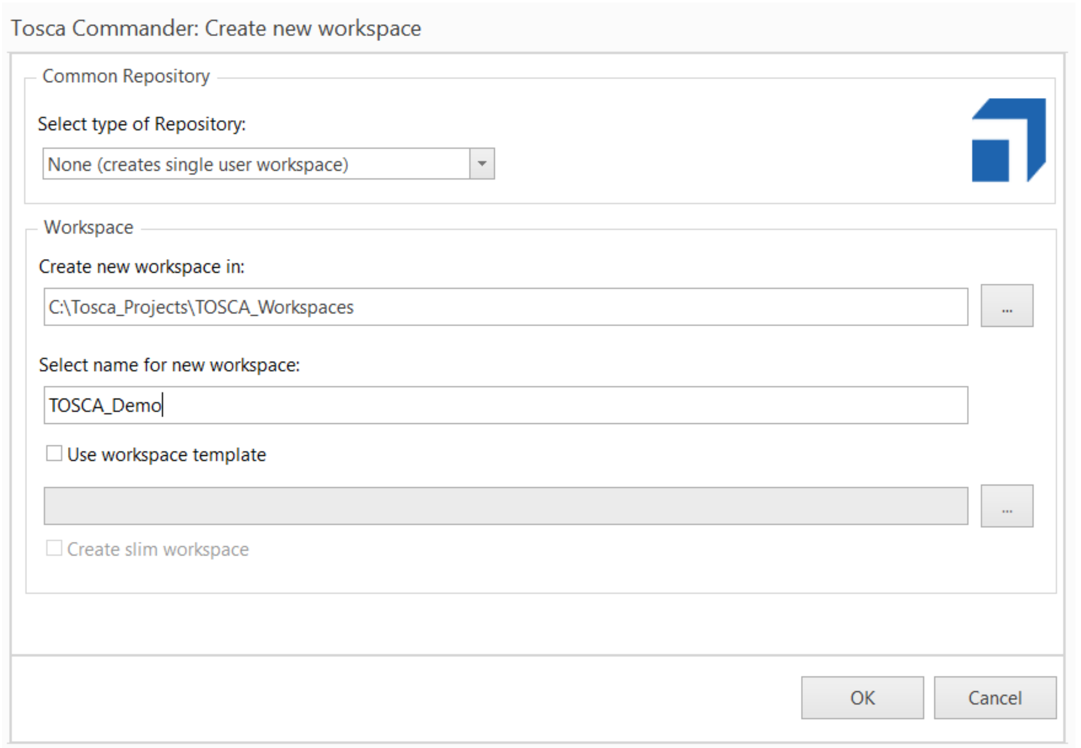
LogEntry

ExecutionEntryFolder

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: Create new workspace

Common Repository
Select type of Repository: None (creates single user workspace)

Workspace
Create new workspace in: C:\Tosca_Projects\TOSCA_Workspaces
Select name for new workspace: TOSCA_Demo
☐ Use workspace template
☐ Create slim workspace

TOSCA Commander - Module

Creating Modules

- 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.

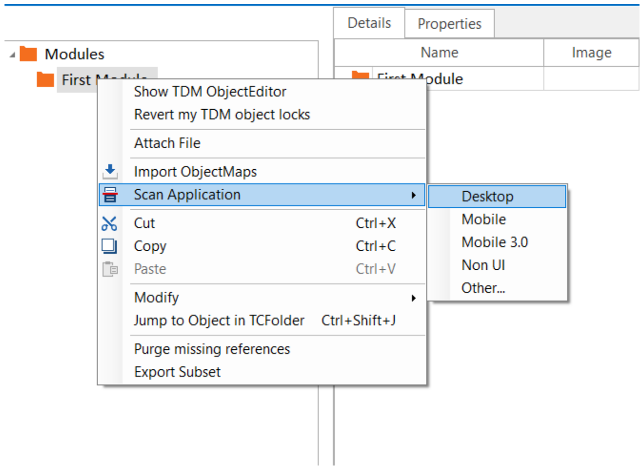
TOSCA Commander - Module

Xscan

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

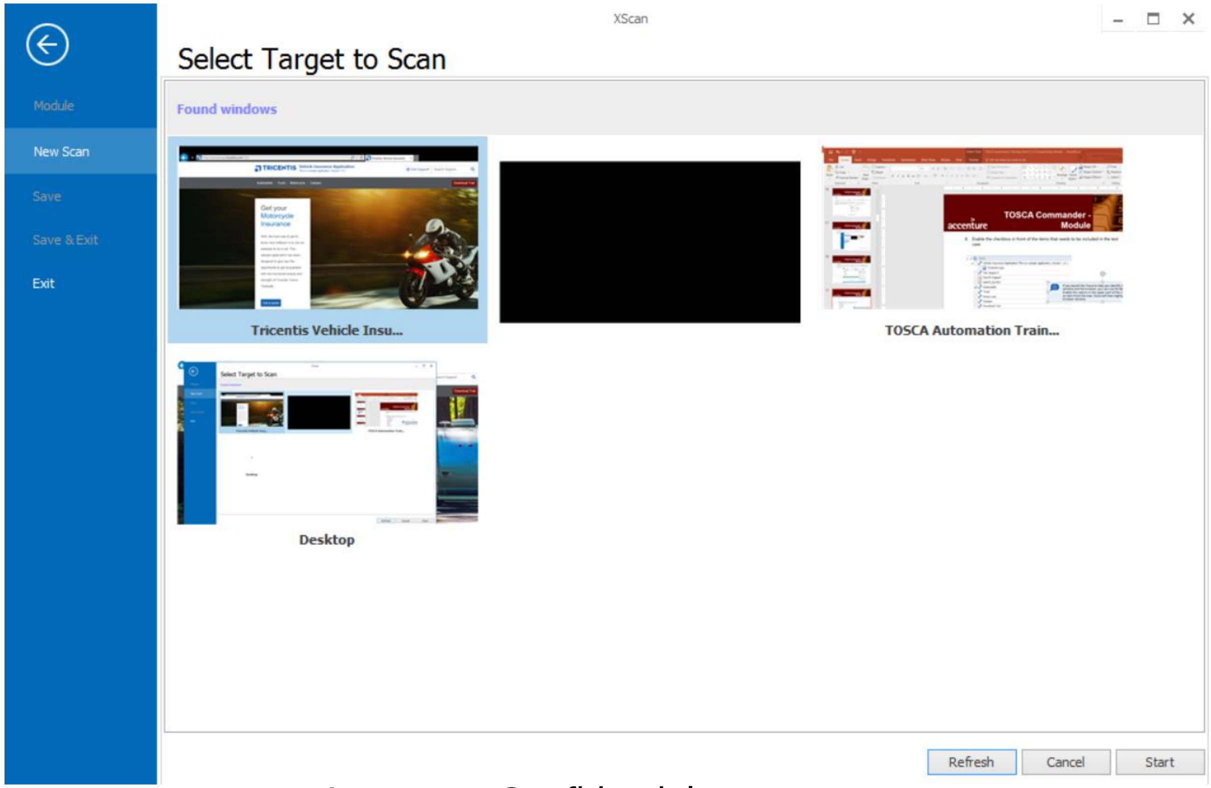
2.Assign a relevant name to the new Module folder.

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



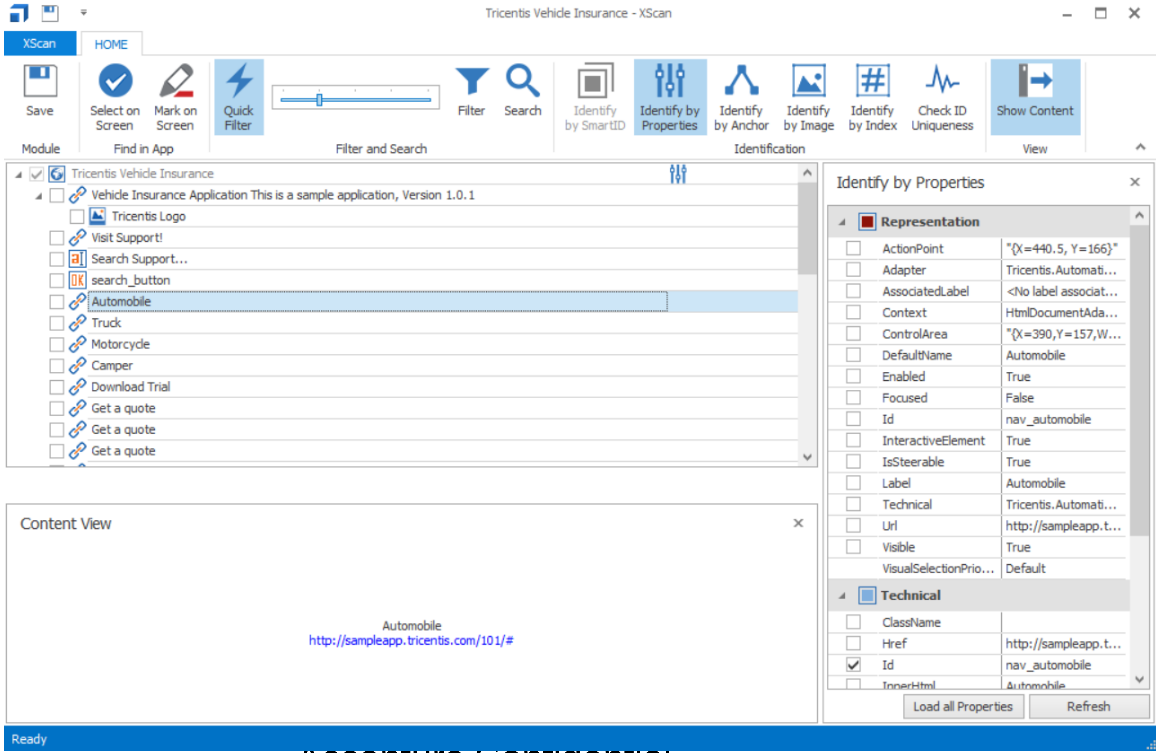
TOSCA Commander - Module

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 then click on **Start.**



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.



Tricentis Vehicle Insurance
- Vehicle Insurance Application This is a sample application, Version 1.0.1
  - Tricentis Logo
- Visit Support!
- Search Support...
- search_button
- Automobile
- Truck
- Motorcycle
- Camper
- Download Trial
- Get a quote
- Get a quote
- Get a quote

Identify by Properties

Representation

ActionPoint	"{X=440.5, Y=166}"
Adapter	Tricentis.Automati...
AssociatedLabel	<No label associat...
Context	HtmlDocumentAda...
ControlArea	"{X=390,Y=157,W...
DefaultName	Automobile
Enabled	True
Focused	False
Id	nav_automobile
InteractiveElement	True
IsSteerable	True
Label	Automobile
Technical	Tricentis.Automati...
Url	http://sampleapp.t...
Visible	True
VisualSelectionPrio...	Default

Technical

ClassName	
Href	http://sampleapp.t...
✓ Id	nav_automobile
InnerHtml	Automobile

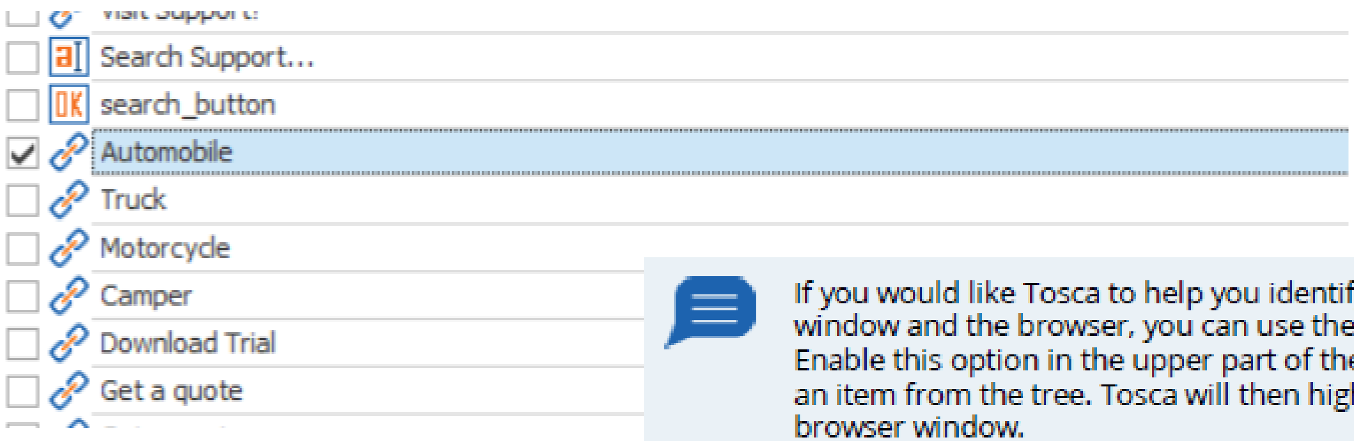
Load all Properties
Refresh

Content View

Automobile
http://sampleapp.tricentis.com/101/#

TOSCA Commander - Module

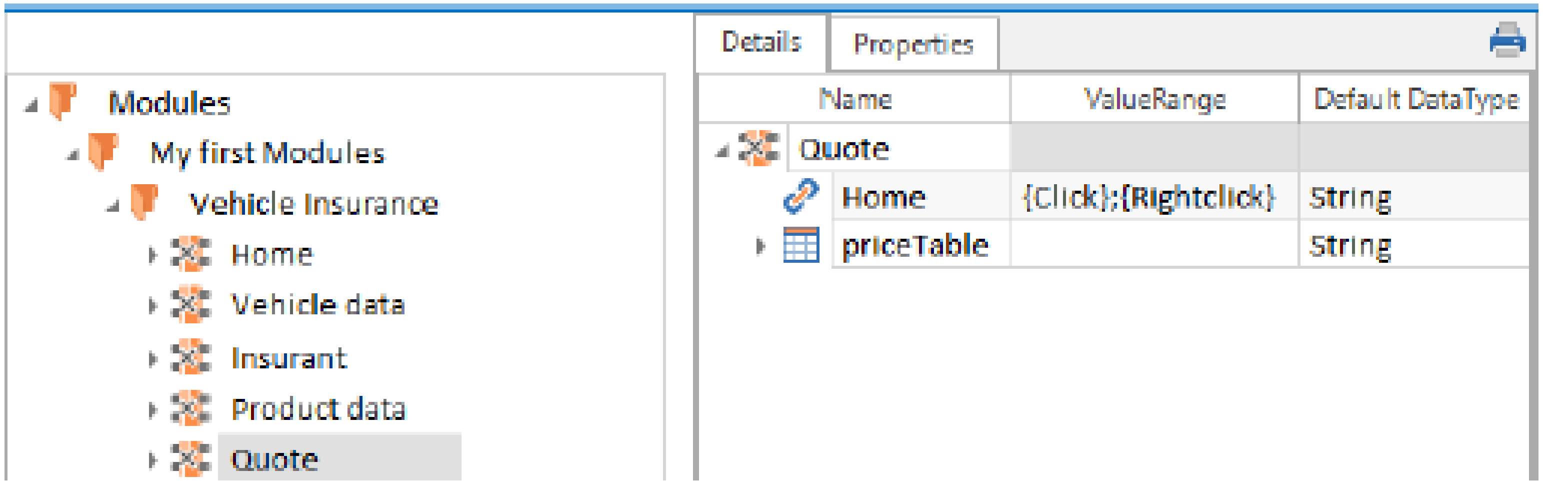
8. Enable the checkbox in front of the items that needs to be included in the test case



If you would like Tosca to help you identify the items in the **XScan** window and the browser, you can use the **Mark on Screen** function. Enable this option in the upper part of the XScan window and select an item from the tree. Tosca will then highlight this in red in the browser window.

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.



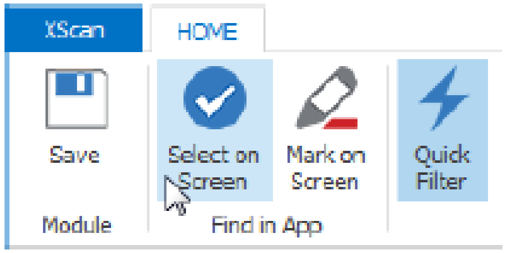
Name	ValueRange	Default DataType
Quote		
Home	{Click};{Rightclick}	String
priceTable		String

Accenture Confidential

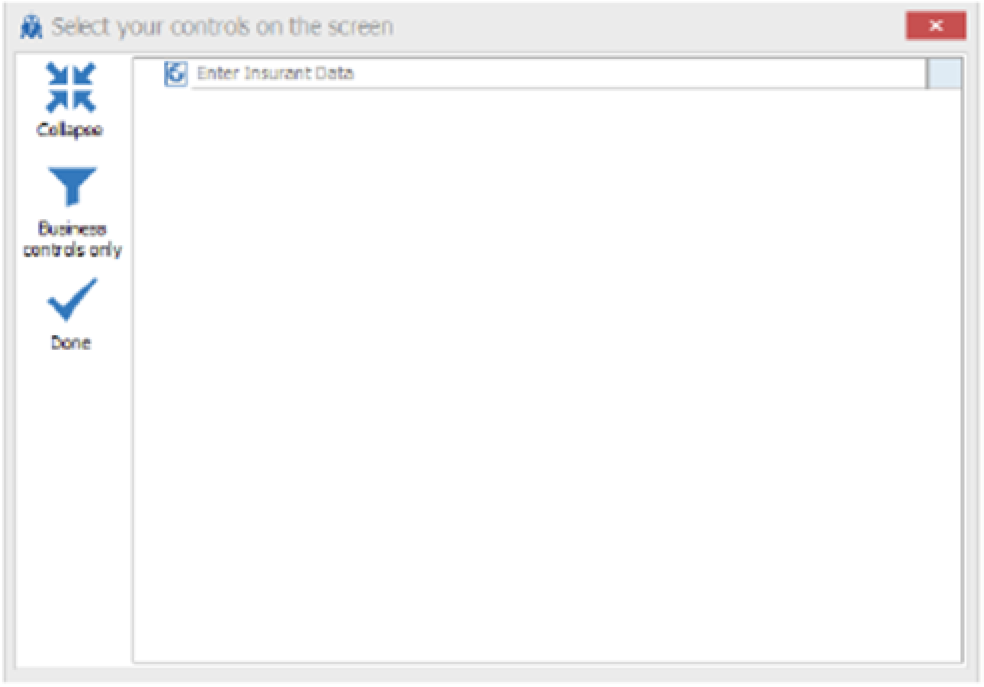
TOSCA Commander - Module

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

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 Commander - Module

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

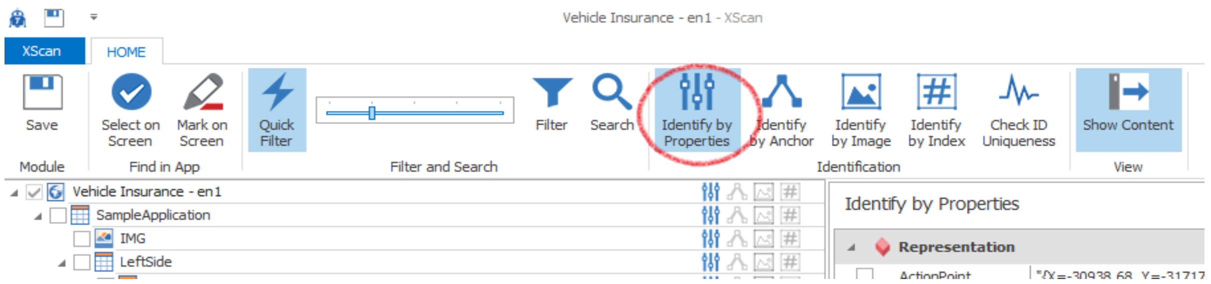
- Menu item Options
  - Scanning -> Scan all Items
    - TOSCA Wizard scans all controls and offers the possibility for them to be stored in an ObjectMap
  - Scanning -> Enable hotkey
    - The scan process is started through use of a single key on the keyboard. This can be defined by selecting Project -> 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

TOSCA Commander - Module

Object Identification and Steering

- There are instances where the default scan setting cannot steer the control under test as the captured control property is either dynamic (property always change) or not unique. Option is to try one of the following identification criteria:

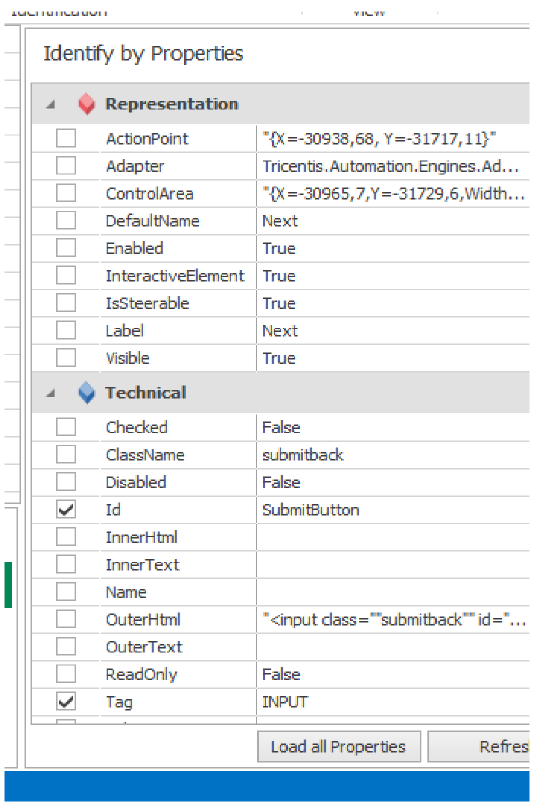
Identify by Properties
This window contains all the technical information and properties of the selected control.



ActionPoint "{X=-30938.68, Y=-31717

TOSCA Commander - Module

By clicking the Load all Properties button, all technical properties will be loaded for the selected control.


Identify by Properties

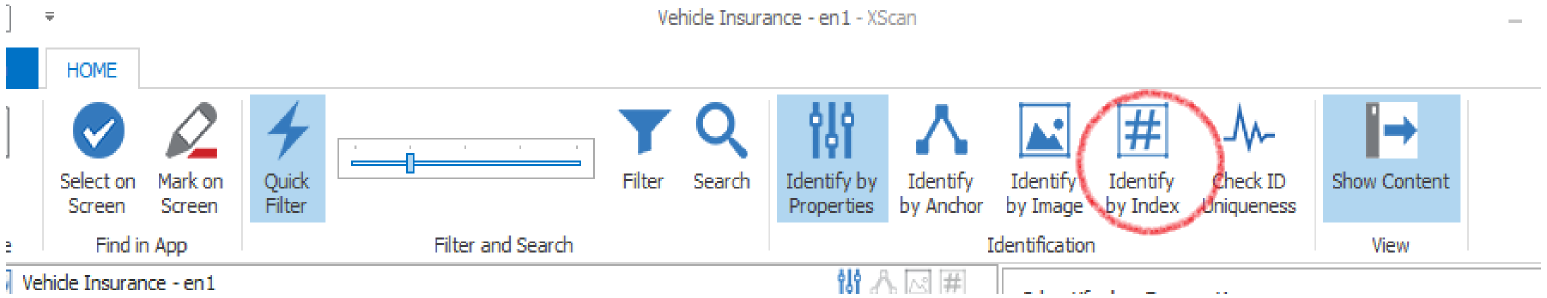
	Representation	
☐	ActionPoint	"{X=-30938,68, Y=-31717,11}"
☐	Adapter	Tricentis.Automation.Engines.Ad...
☐	ControlArea	"{X=-30965,7,Y=-31729,6,Width...
☐	DefaultName	Next
☐	Enabled	True
☐	InteractiveElement	True
☐	IsSteerable	True
☐	Label	Next
☐	Visible	True
	Technical	
☐	Checked	False
☐	ClassName	submitback
☐	Disabled	False
✓	Id	SubmitButton
☐	InnerHtml	
☐	InnerText	
☐	Name	
☐	OuterHtml	"<input class=""submitback"" id="...
☐	OuterText	
☐	ReadOnly	False
✓	Tag	INPUT

From the loaded properties, select the ones that will uniquely identify the object by checking and unchecking the preceding checkbox.

TOSCA Commander - Module

Identify by Index

If a selected control does not have a unique ID, you can select an index to be used upon test execution to identify the control.

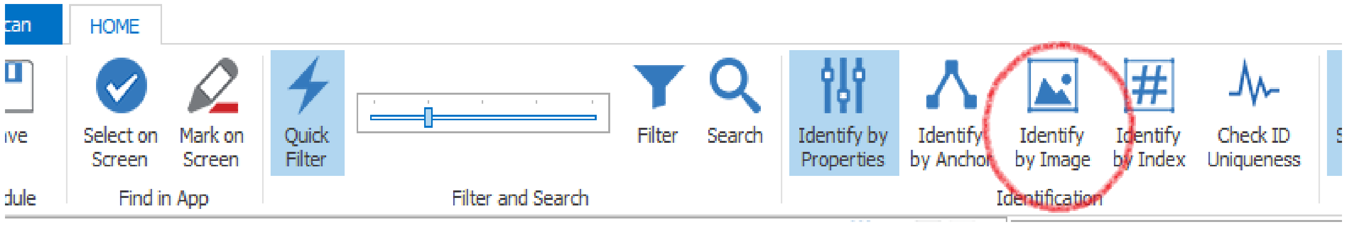


The index is calculated dynamically and on the basis of already selected identification criteria.

TOSCA Commander - Module

Identify by Image

Here, you can define an image to used for identifying controls.

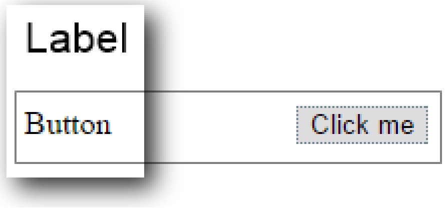


Tosca is able to identify controls from an image by creating a screenshot for a specific control.

TOSCA Commander - Module

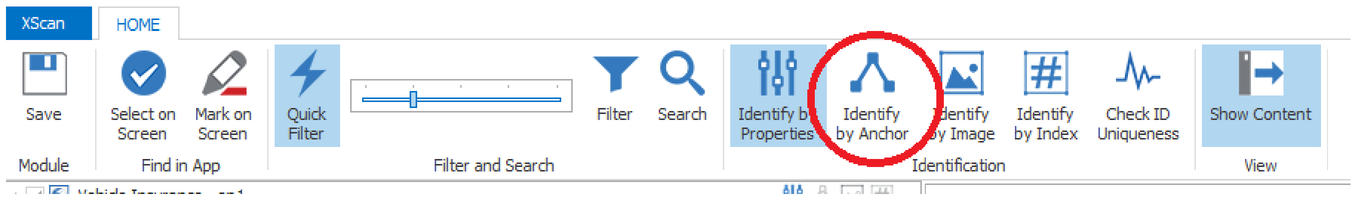
Identify controls via anchors

In Tosca, you are able to use technical properties of controls in order to identify other controls, for instance if you copy a textbox Label to the Textbox. In this case, Tosca copies the identification criteria from the anchor control to the selected control.



To use identify by anchor, follow steps below:

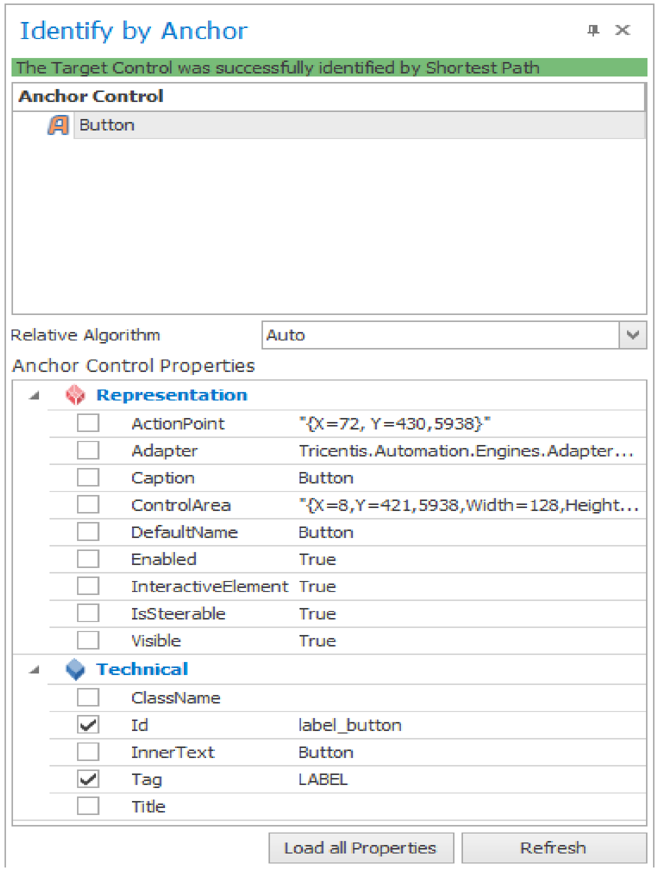
1. Scan the required test object by using Tosca XScan.
2. Open the Identify by Anchor window by clicking on the corresponding button in the HOME menu.



Accenture Confidential

TOSCA Commander - Module

3. Select the control which should be identified.
4. Use drag and drop to move the control, whose identification criteria should be copied, to the Anchor Control field. The control is shown in this field as soon as you release the mouse button.


Identify by Anchor

The Target Control was successfully identified by Shortest Path

Anchor Control

Button

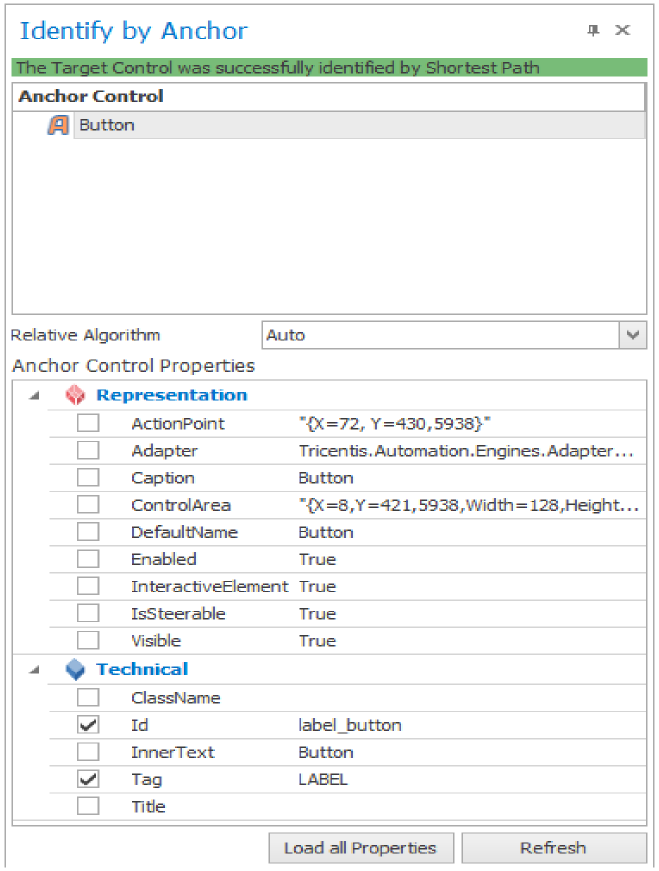
Relative Algorithm: Auto

Anchor Control Properties

	Representation	
☐	ActionPoint	"{X=72, Y=430,5938}"
☐	Adapter	Tricentis.Automation.Engines.Adapter...
☐	Caption	Button
☐	ControlArea	"{X=8,Y=421,5938,Width=128,Height...
☐	DefaultName	Button
☐	Enabled	True
☐	InteractiveElement	True
☐	IsSteerable	True
☐	Visible	True
	Technical	
☐	ClassName	
✓	Id	label_button
☐	InnerText	Button
✓	Tag	LABEL
☐	Title	

TOSCA Commander - Module

3. Select the control which should be identified.
4. Use drag and drop to move the control, whose identification criteria should be copied, to the Anchor Control field. The control is shown in this field as soon as you release the mouse button.
5. The Anchor Control Properties field shows the properties that were transferred along with the control.



Identify by Anchor

The Target Control was successfully identified by Shortest Path

Anchor Control

Button

Relative Algorithm: Auto

Anchor Control Properties

Representation

☐	ActionPoint	"{X=72, Y=430,5938}"
☐	Adapter	Tricentis.Automation.Engines.Adapter...
☐	Caption	Button
☐	ControlArea	"{X=8,Y=421,5938,Width=128,Height...
☐	DefaultName	Button
☐	Enabled	True
☐	InteractiveElement	True
☐	IsSteerable	True
☐	Visible	True

Technical

☐	ClassName	
✓	Id	label_button
☐	InnerText	Button
✓	Tag	LABEL
☐	Title	

Exercises

Please do Exercise 1 – 3. Refer to the Activities for instructions.

accenture

TOSCA Commander - Test Case

A TestCase in TOSCA Commander™

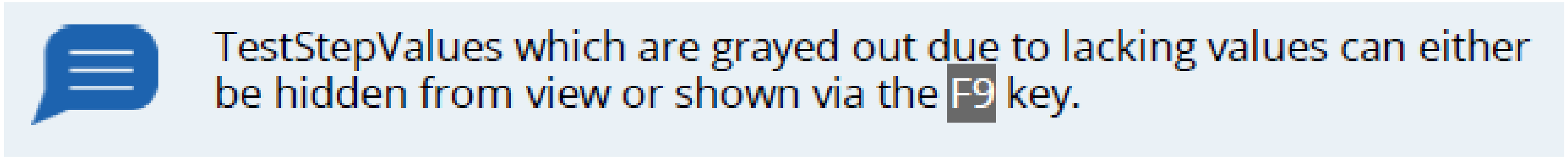
- Consists of m TestSteps, which contain n TestStepValues
- Describes the functional test process
- Is processed step by step from top to bottom
- Will always be linked to precise values
- Loops and If statements are possible

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.

TOSCA Commander - Test Case

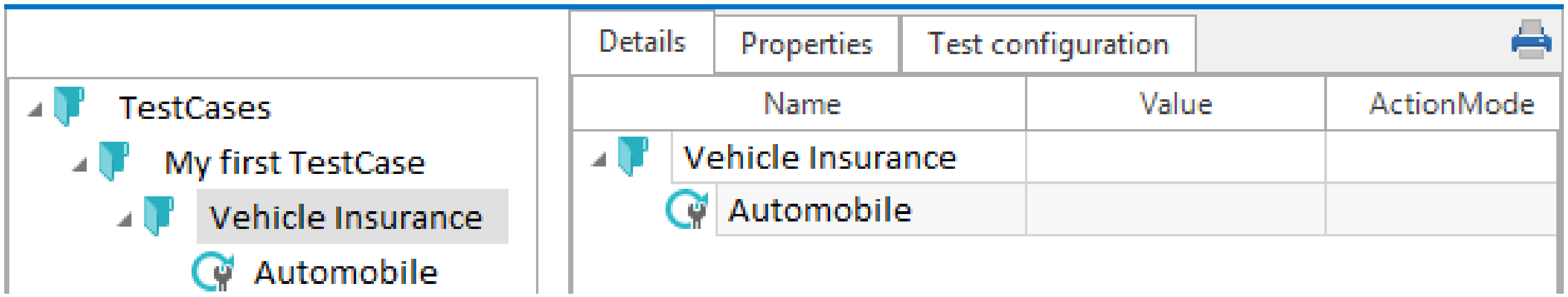
- **TestStepValue**
  - TestStepValues contain the actual information required for steering the system under test and can vary in form.
  - The TestStepValue icons are grayed out until you define actions for these values.


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.

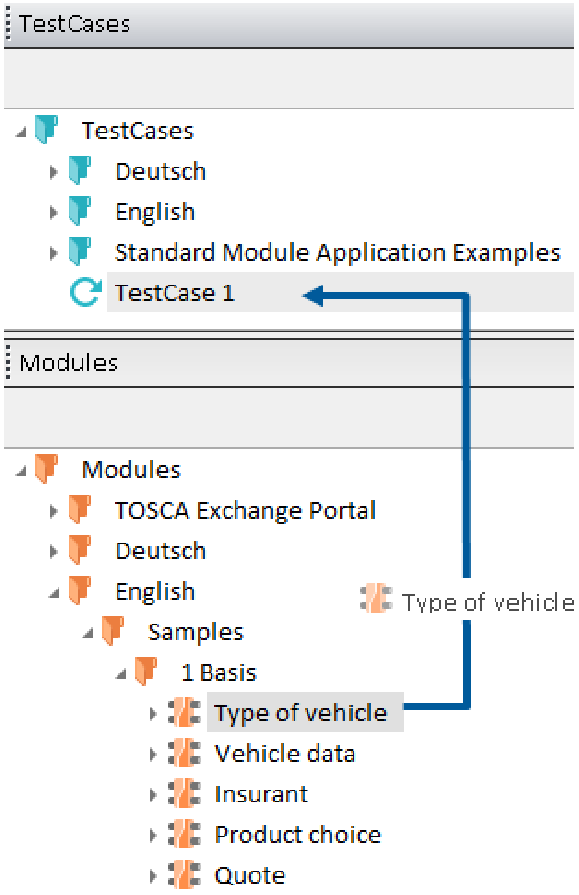


Name	Value	ActionMode
Vehicle Insurance		
Automobile		

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.


TestCases

TestCases
- Deutsch
- English
- Standard Module Application Examples
- TestCase 1

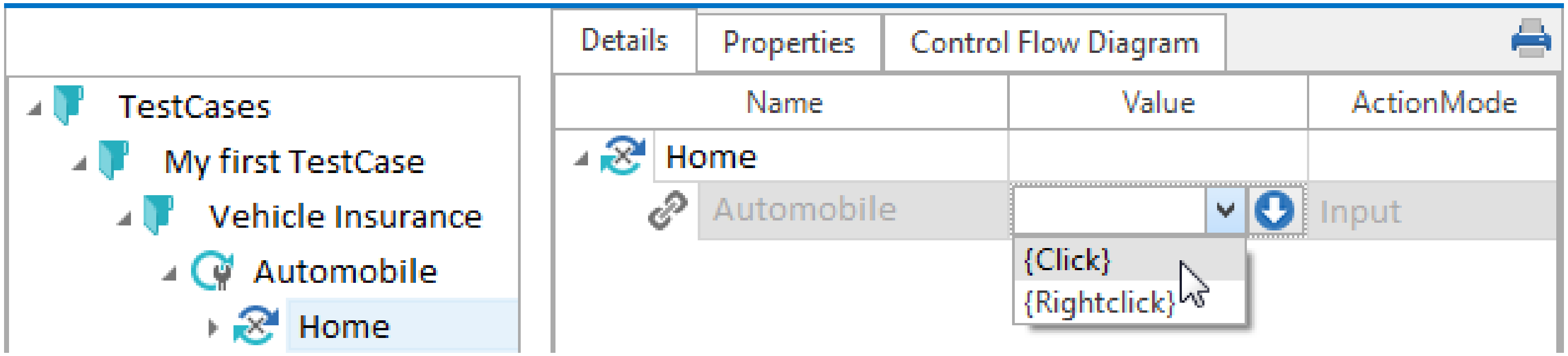
Modules

Modules
- TOSCA Exchange Portal
- Deutsch
- English
  - Samples
    - 1 Basis
      - Type of vehicle
      - Vehicle data
      - Insurant
      - Product choice
      - Quote

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,



Name	Value	ActionMode
Home		
Automobile		Input

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

Please do Exercise 4. Refer to the Activities for instructions.

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 specifiying which column should be steered
  - Row: Entry specifiying 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.

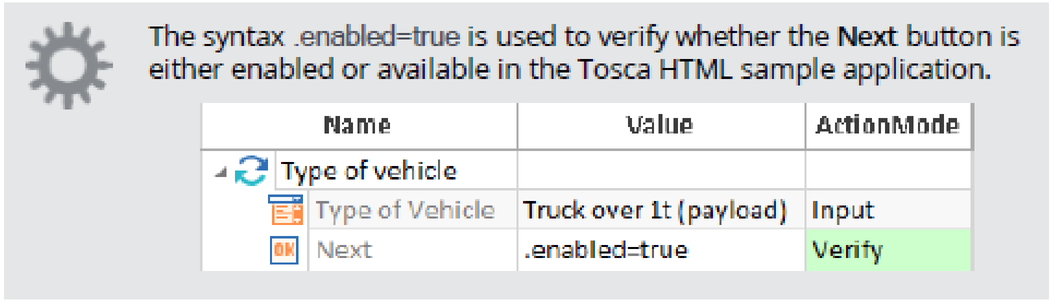
- List of ActionModes:
  - Input
  - Select
  - Verify
  - Buffer
  - WaitOn
  - Constraint

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.

accenture

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

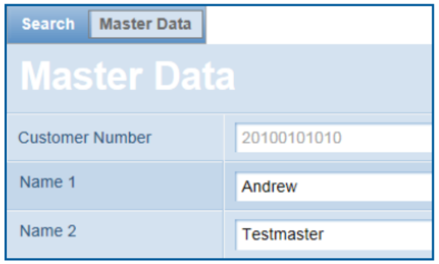
Name	Description
.contains	This verifies whether a specific content exists in a ComboBox or a ListView.
.list	The entire content of a ComboBox or ListView is verified. The entered values are case-sensitive, and they are verified according to the specified sequence. The content of the entire list is verified.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

Exercise

Please do Exercise 5. Refer to the Activities for instructions.

ActionMode Buffer

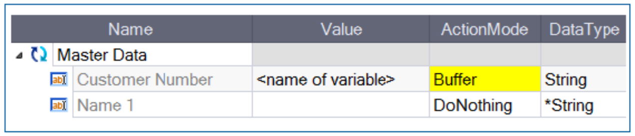
- The ActionMode Buffer is used to save variable values from the system under test to a local buffer (variable storage memory).
  - Stored values can be used at any stage during test execution :
    - {B[name of saved variable]}
  - Stored values can be viewed in the menu item:
    - Tools->Settings->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



Customer Number	20100101010
Name 1	Andrew
Name 2	Testmaster

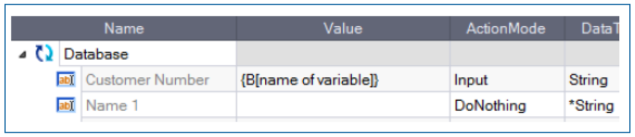
Illustration : TRICENTIS Bank - GUI

1 Save variable in local buffer



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

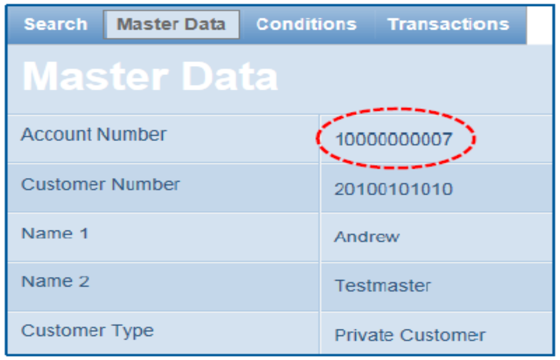
2 Use saved variable from local buffer



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

Illustration : TRICENTIS Bank - database

ActionMode Buffer – Table Steering

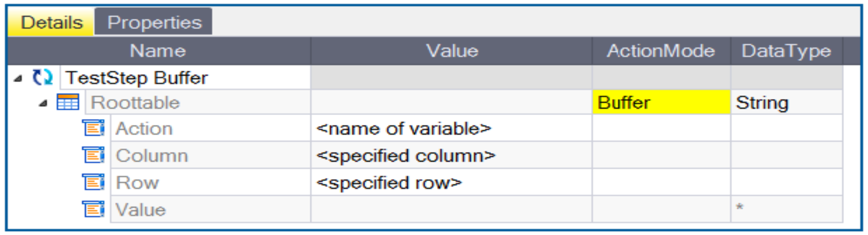


Search | Master Data | Conditions | Transactions

Master Data

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

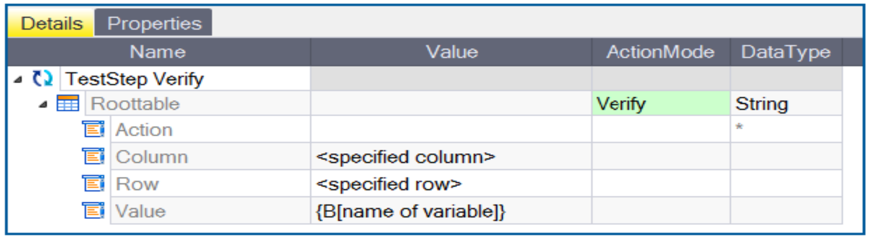
1 Save variable in local buffer



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

Exercise

Please do Exercise 6. Refer to the Activities for instructions.

ActionMode WaitOn

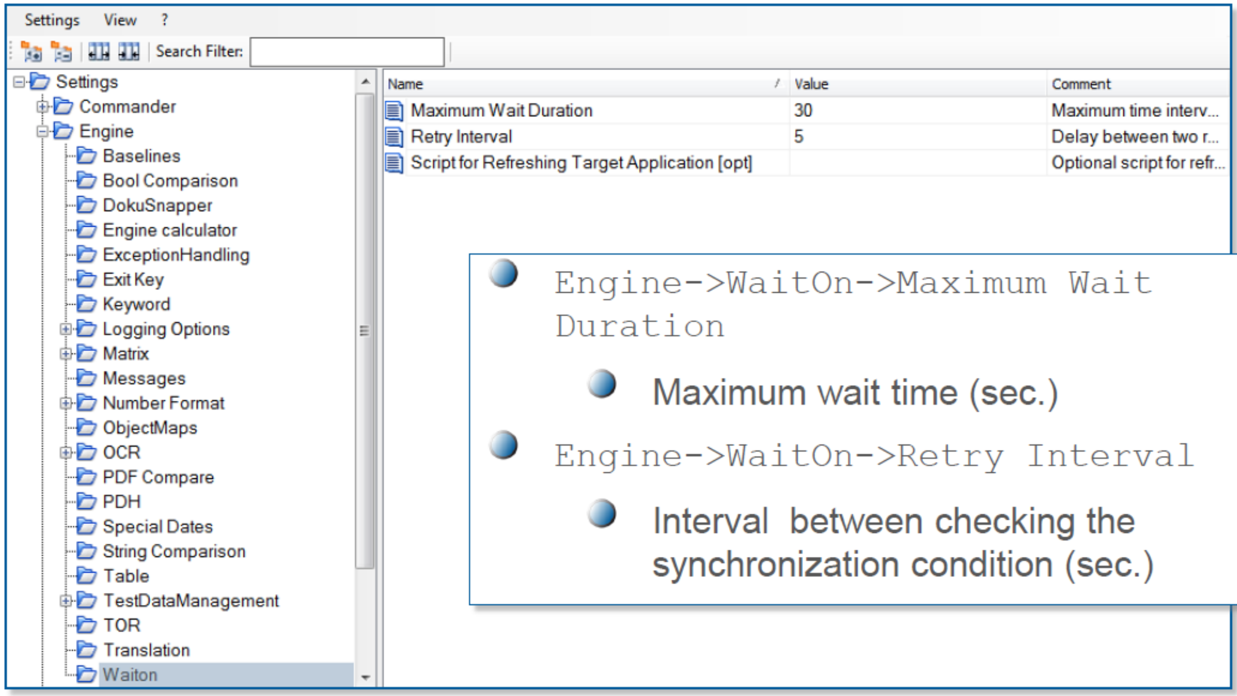
- Asynchronous process operations occur in practically every complex test project
- Test execution must be adapted to these operations, so that test instructions are not prematurely sent to the system under test (which is not yet ready)
- Dynamic synchronization with the process speed of the system under 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)

ActionMode WaitOn

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



Name	Value	Comment
Maximum Wait Duration	30	Maximum time interv...
Retry Interval	5	Delay between two r...
Script for Refreshing Target Application [opt]		Optional script for refr...

- `Engine->WaitOn->Maximum Wait Duration`
  - Maximum wait time (sec.)
- `Engine->WaitOn->Retry Interval`
  - Interval between checking the synchronization condition (sec.)

Illustration: TOSCA Commander™ Settings

Exercise

Please do Exercise 7. Refer to the Activities for instructions.

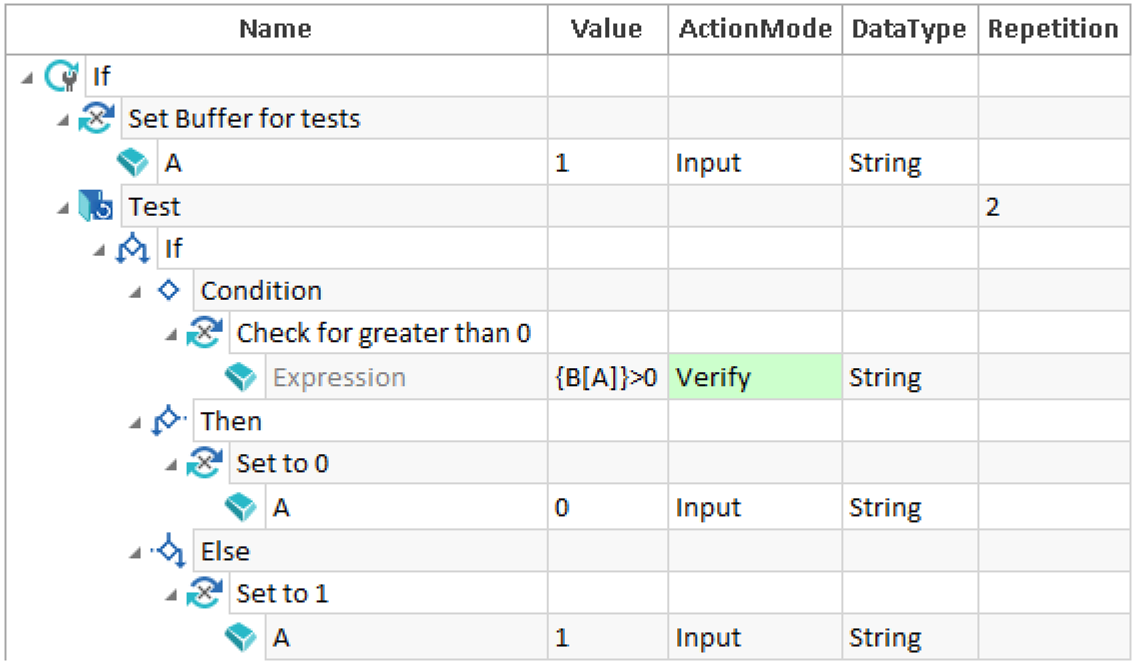
Conditional Statement

- In Tosca, you can define **IF**, **DO** and **WHILE** statements if you would like to run TestSteps repeatedly. These statements can be applied to any nested structures.

- Conditional statements can be created from the context menu of TestCases or TestStep folders:
  - IF Statement
  - DO Statement
  - WHILE Statement

IF Statement

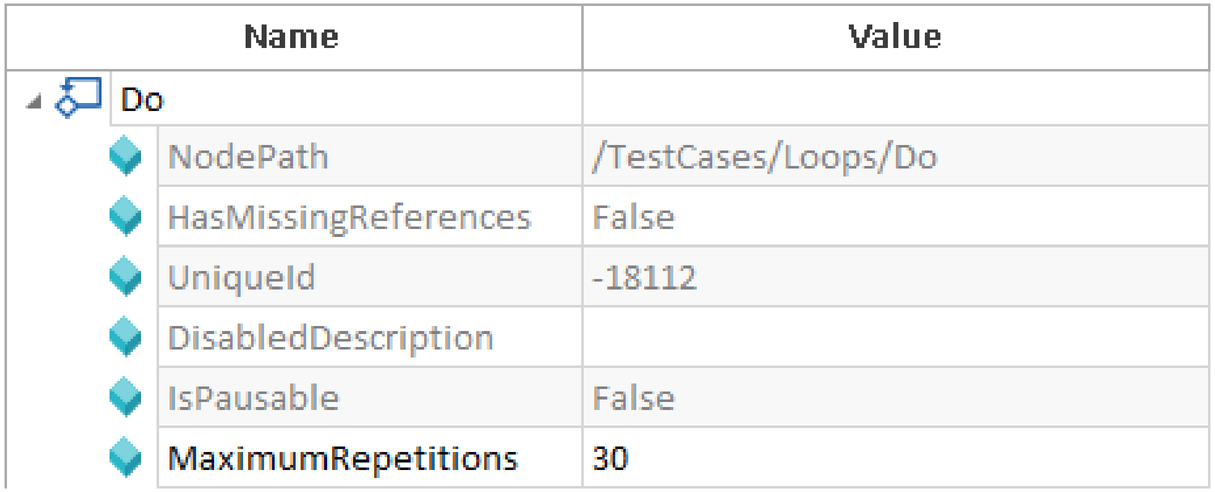
To create an IF Statement, right click on the Test Case and click the If Statement logo.



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	

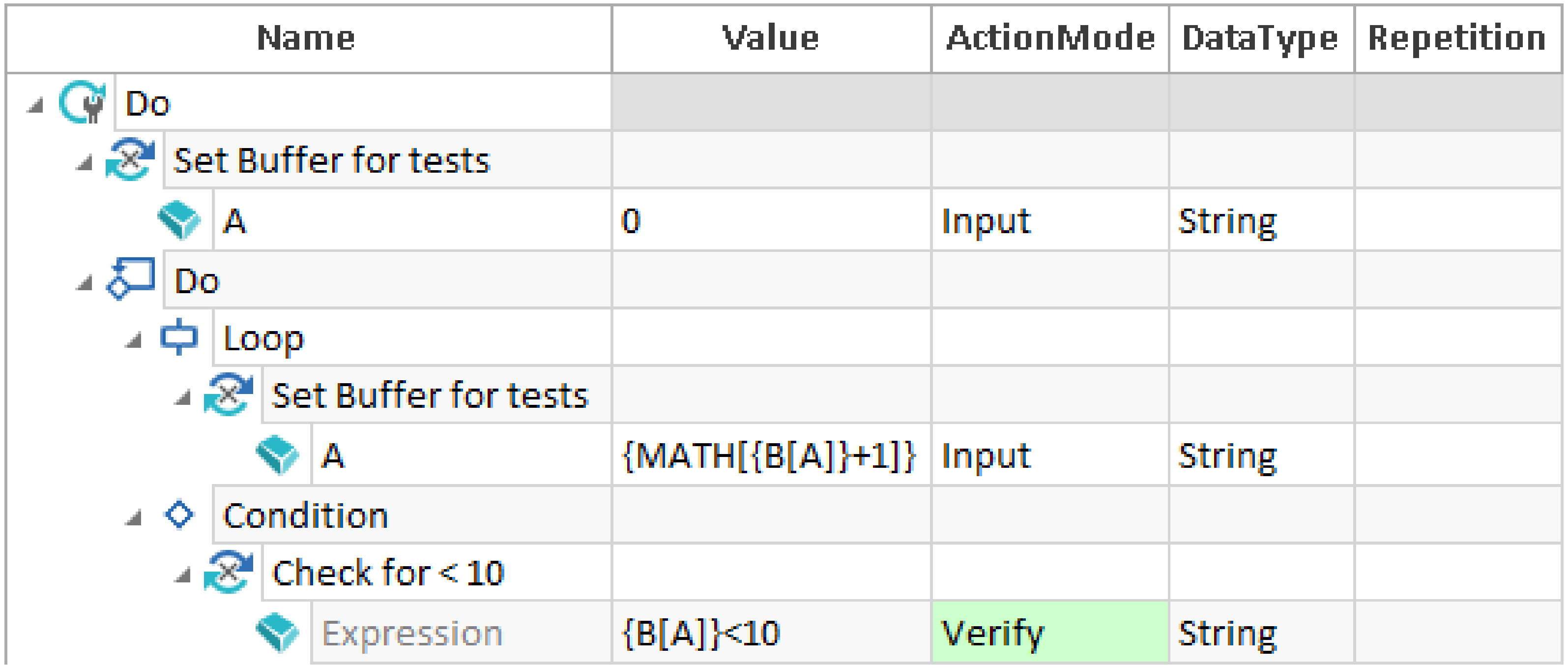
DO and WHILE Statement

- They contain the property **MaximumRepetitions** in order to avoid infinite loops. The **Value** column shows the maximum number of attempts for a TestStep to be run. The default value is set to **30**.filled. The result of the last repetition is thus negative.



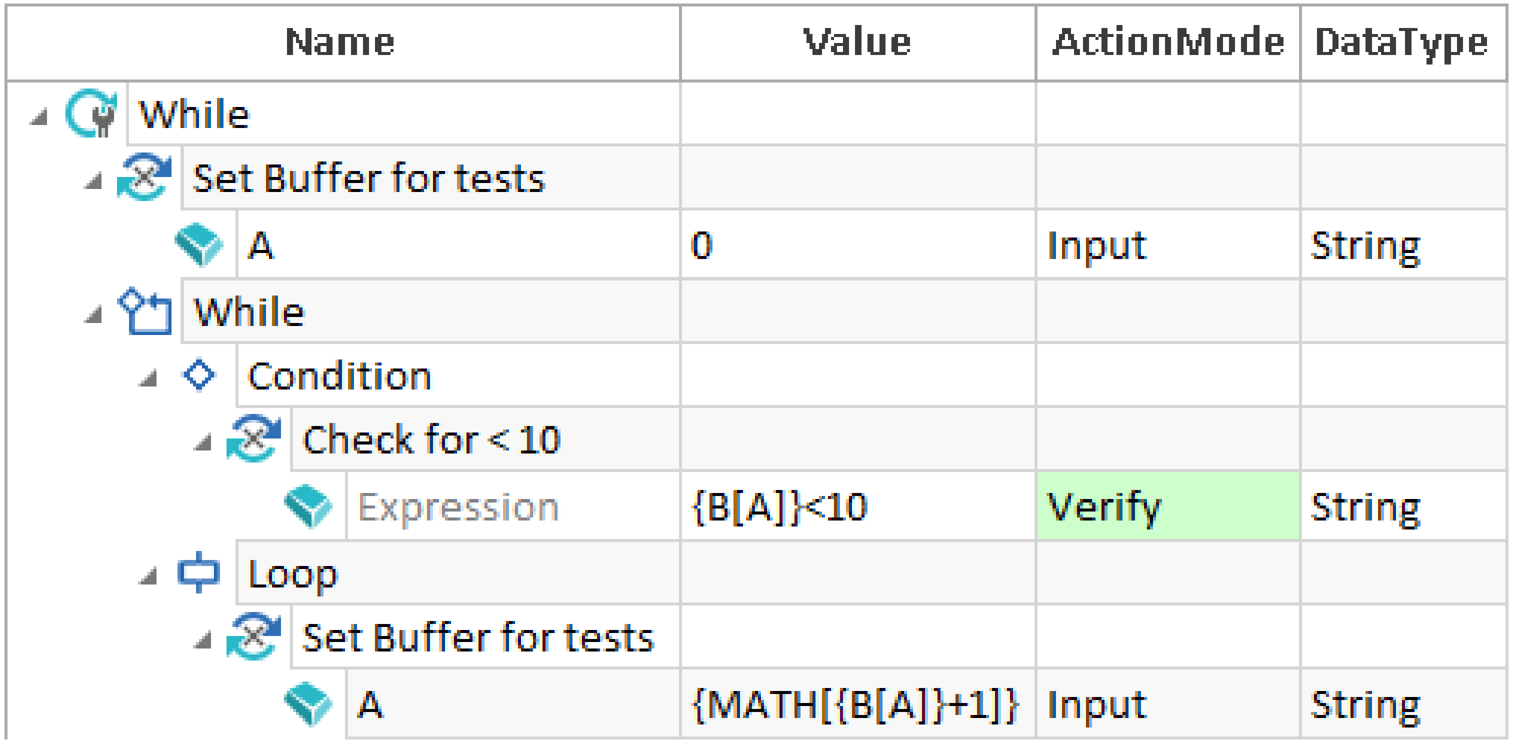
Name	Value
Do	
NodePath	/TestCases/Loops/Do
HasMissingReferences	False
UniqueId	-18112
DisabledDescription	
IsPausable	False
MaximumRepetitions	30

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	

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

accenture

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

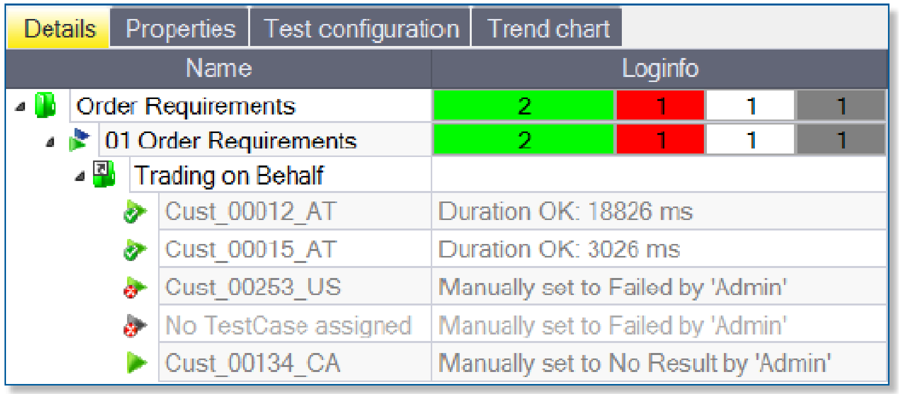
Execution List

- **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**.



Details | Properties | Test configuration | Trend chart

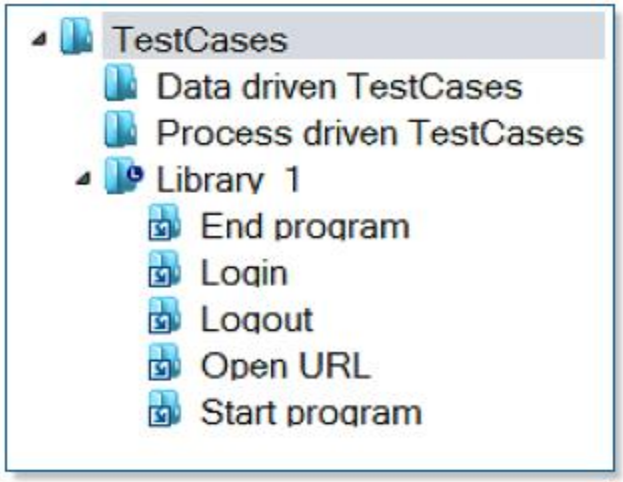
Name	Loginfo			
Order Requirements	🟩 2	🟥 1	1	▨ 1
01 Order Requirements	🟩 2	🟥 1	1	▨ 1
Trading on Behalf				
Cust_00012_AT	Duration OK: 18826 ms			
Cust_00015_AT	Duration OK: 3026 ms			
Cust_00253_US	Manually set to Failed by 'Admin'			
No TestCase assigned	Manually set to Failed by 'Admin'			
Cust_00134_CA	Manually set to No Result by 'Admin'			

Exercise

Please do Exercise 8. Refer to the Activities for instructions.

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**



TestCases
Data driven TestCases
Process driven TestCases
Library 1
End program
Login
Logout
Open URL
Start program

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**).

Exercise

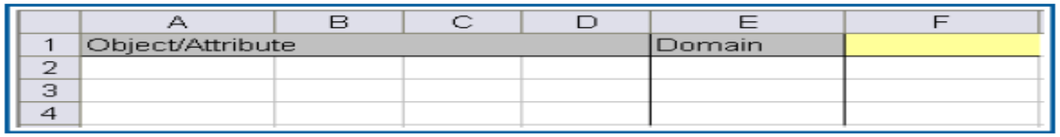
Please do Exercise 9. Refer to the Activities for instructions.

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 is collected in a Microsoft® Excel Worksheet as test data combinations.
- 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

- The TestCase data is collected in **Microsoft® Excel**. The attributes and test data are organized in rows and columns in a **Worksheet**.

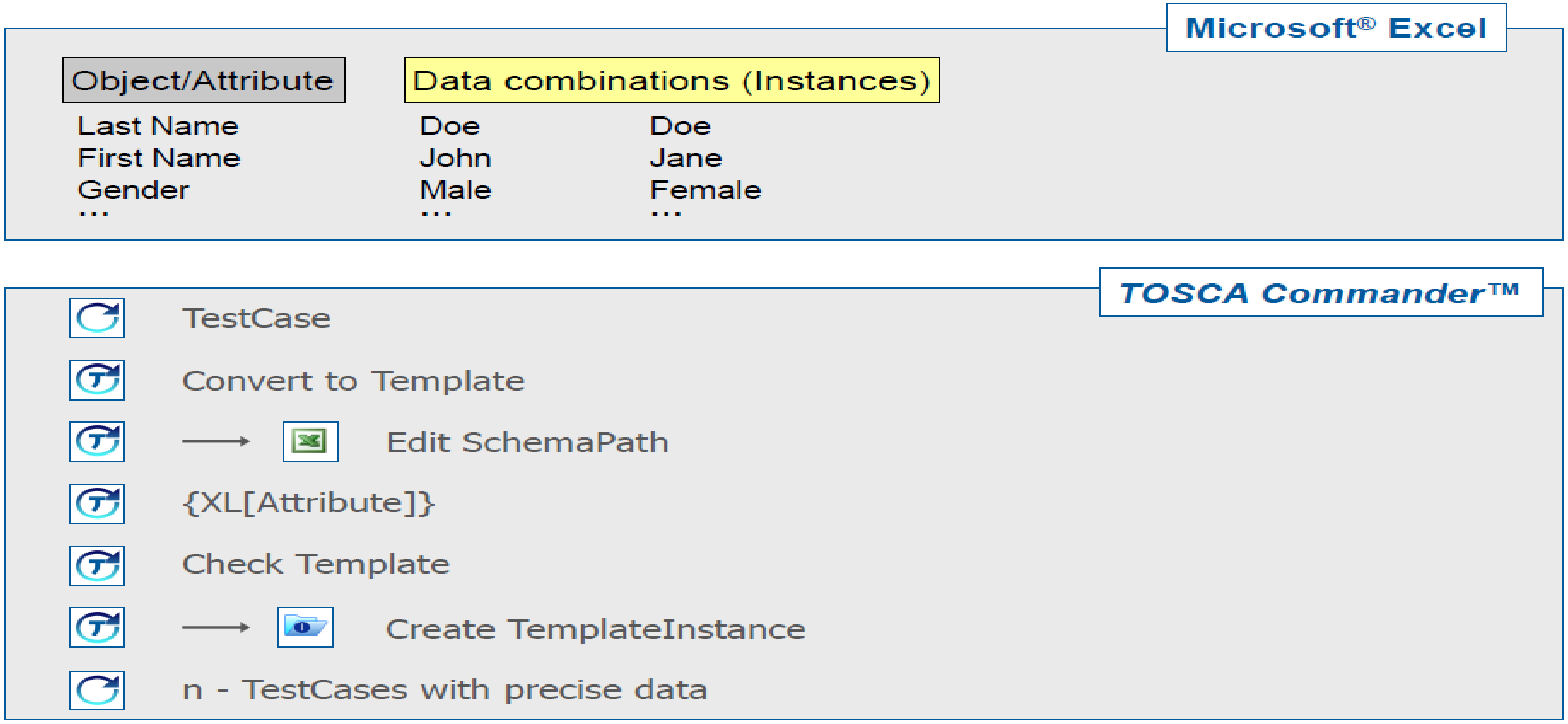


	A	B	C	D	E	F
1	Object/Attribute				Domain	
2						
3						
4						

- Column descriptions appear in **row 1**.
  - The attributes are entered into the **columns A to D**.
  - The attributes are arranged in columns
- Special characters e.g. dots » . « should not be used.
- The test data for TestCases is defined from **column F** onwards
- Each test data combination is given a unique TestCase name in row **1**.
- The background colors must be maintained.

Microsoft® Excel as a source of data

Dynamic Test Case Generation Sequence.


Microsoft® Excel

Object/Attribute	Data combinations (Instances)	
Last Name	Doe	Doe
First Name	John	Jane
Gender	Male	Female
…	…	…

TOSCA Commander™

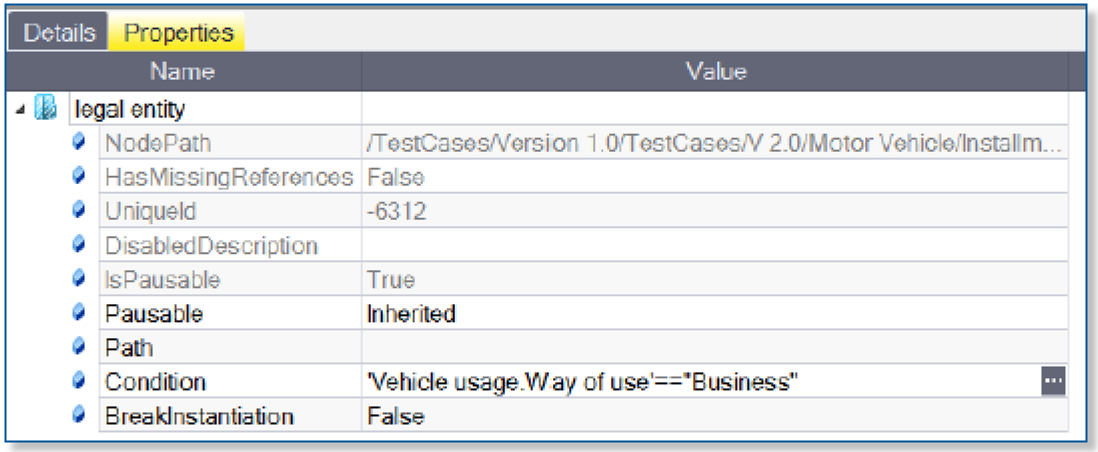
- TestCase
- Convert to Template
- → Edit SchemaPath
- {XL[Attribute]}
- Check Template
- → Create TemplateInstance
- n - TestCases with precise data

Exercise

Please do Exercise 10. Refer to the Activities for instructions.

Conditional Instantiation

- Within a **Template**, it is only possible to incorporate specific TestSteps into **TemplateInstances** if certain business-based conditions are met.
- In conditional instantiation, the dialog sequences in the TestCase depend on the availability of the values for a specific business object or attribute in the Microsoft® Excel worksheet.
- TestSteps and TestStep Folders have the property **condition** in Templates.



Name	Value
legal entity	
NodePath	/TestCases/Version 1.0/TestCases/V 2.0/Motor Vehicle/Installm...
HasMissingReferences	False
UniqueId	-6312
DisabledDescription	
IsPausable	True
Pausable	Inherited
Path	
Condition	'Vehicle usage.Way of use'=="Business"
BreakInstantiation	False

Conditional Instantiation

- The **syntax** of the condition is:
  - <BusinessObject/Attribute><Boolean operator><Value>
  - e.g. ‘City’==“Vienna” or ‘Address.City’==“Vienna”
- The conditions of attributes can be combined
  - e.g. ‘Address.City’==“Vienna” && ‘Age’>=“18”
- This allows the possibility of representing different sequences of dialog in a Template.

Exercise

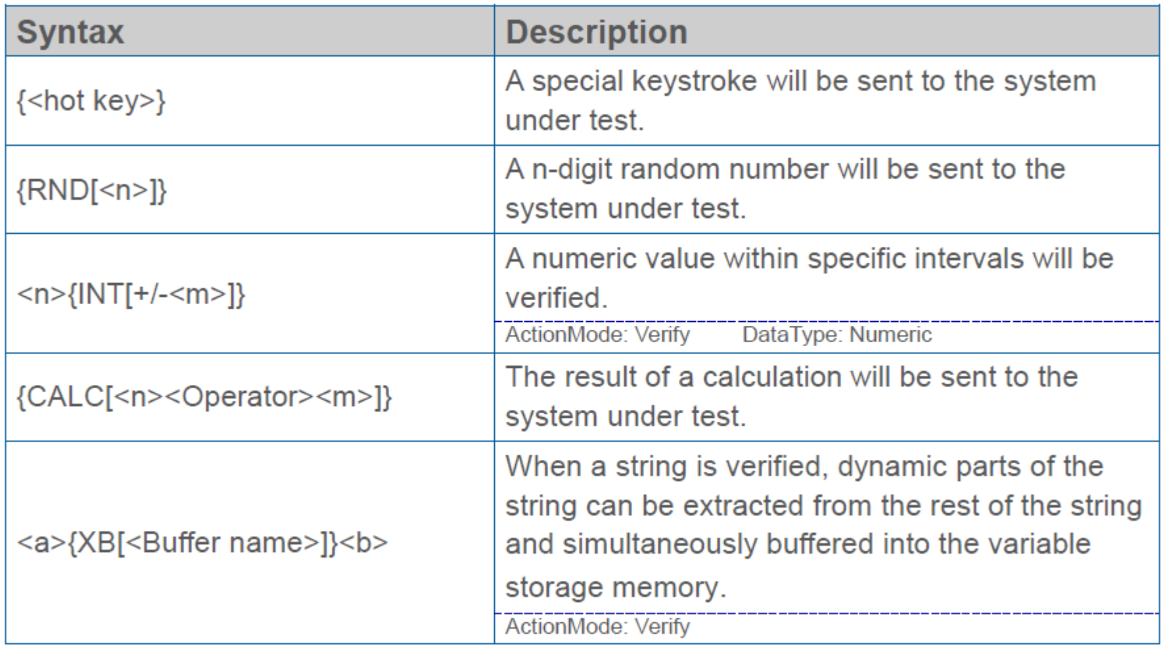
Please do Exercise 11. Refer to the Activities for instructions.

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

- **Special characters**
  - A complete list of special characters can be found in the *TOSCA Commander™* manual.

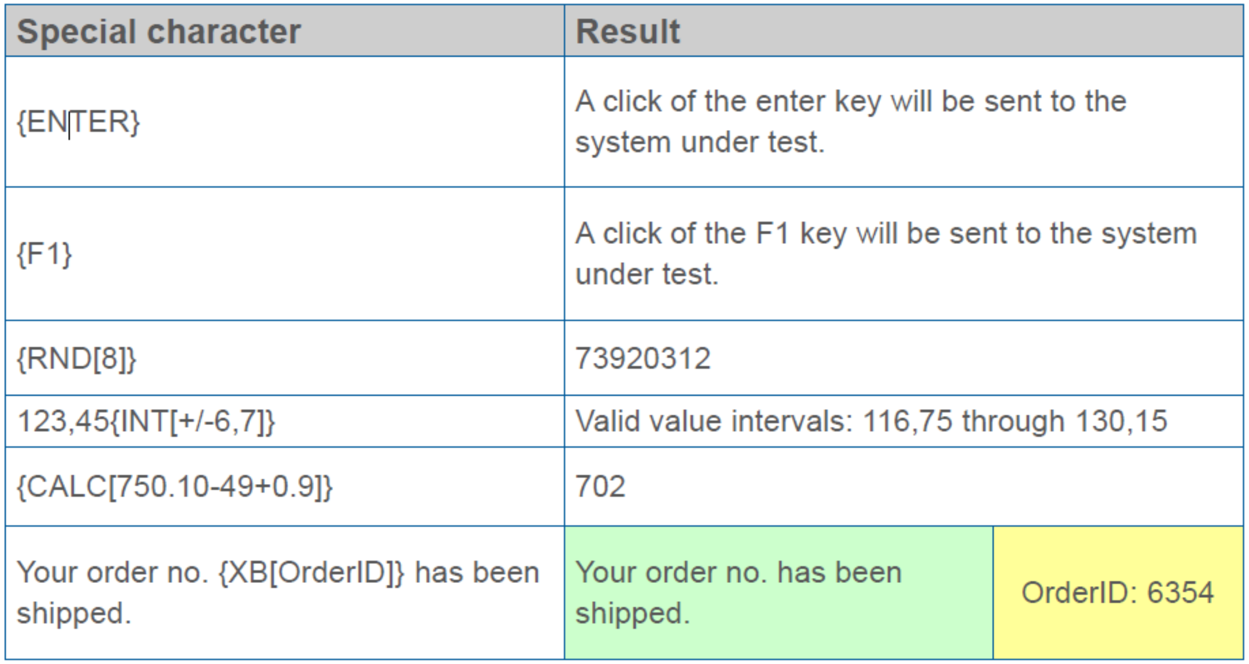


Syntax	Description
{<hot key>}	A special keystroke will be sent to the system under test.
{RND[<n>]}	A n-digit random number will be sent to the system under test.
<n>{INT[+/-<m>]}	A numeric value within specific intervals will be verified. ActionMode: Verify DataType: Numeric
{CALC[<n><Operator><m>]}	The result of a calculation will be sent to the system under test.
<a>{XB[<Buffer name>]}<b>	When a string is verified, dynamic parts of the string can be extracted from the rest of the string and simultaneously buffered into the variable storage memory. ActionMode: Verify

accenture

Special Characters

- **Examples**
  - Valid characters in HTML

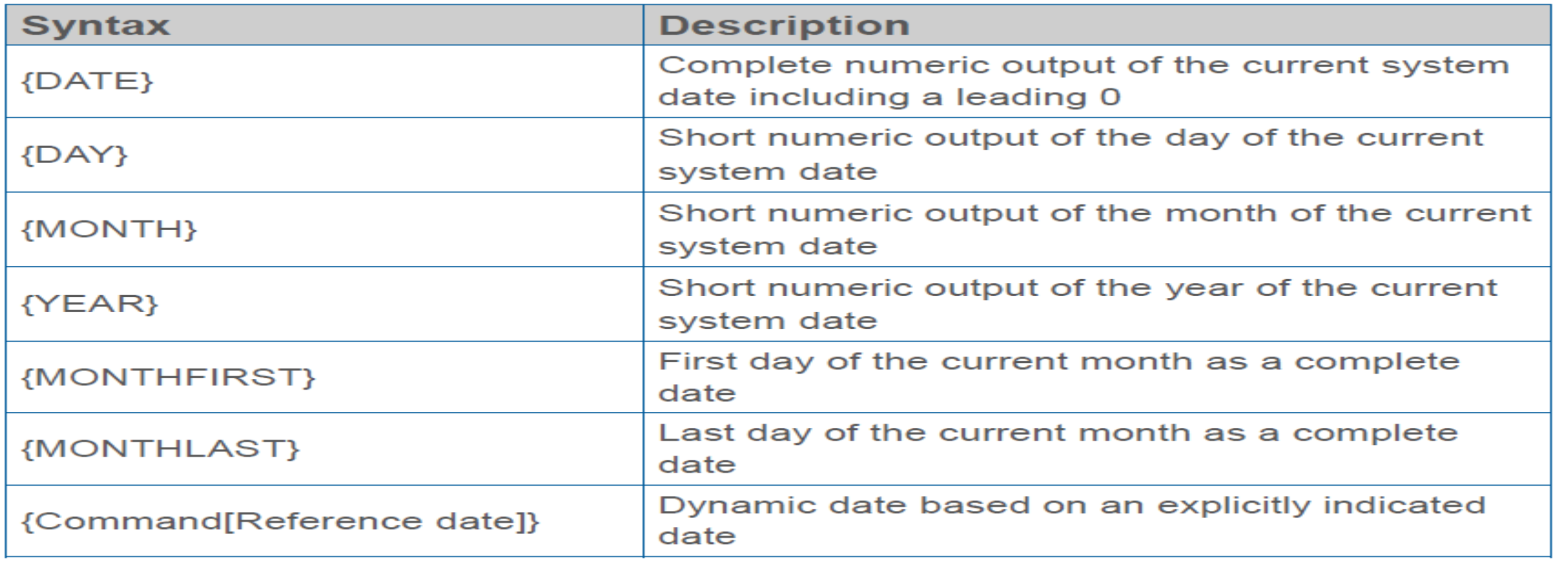


Special character	Result	
{ENTER}	A click of the enter key will be sent to the system under test.	
{F1}	A click of the F1 key will be sent to the system under test.	
{RND[8]}	73920312	
123,45{INT[+/-6,7]}	Valid value intervals: 116,75 through 130,15	
{CALC[750.10-49+0.9]}	702	
Your order no. {XB[OrderID]}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

Dynamic Values

Dynamic Date

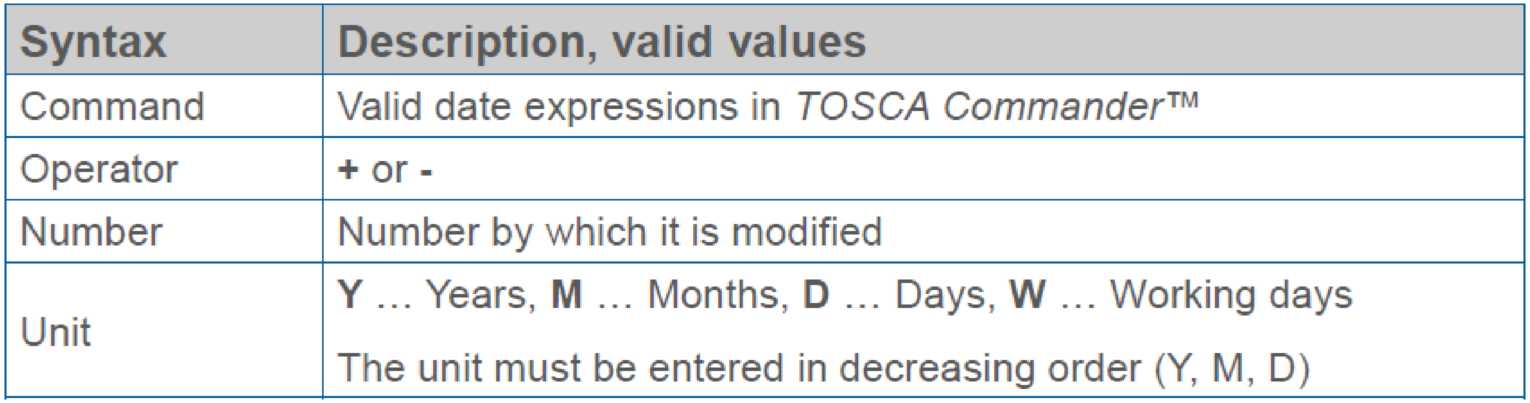
German and English syntaxes are supported



Syntax	Description
{DATE}	Complete numeric output of the current system date including a leading 0
{DAY}	Short numeric output of the day of the current system date
{MONTH}	Short numeric output of the month of the current system date
{YEAR}	Short numeric output of the year of the current system date
{MONTHFIRST}	First day of the current month as a complete date
{MONTHLAST}	Last day of the current month as a complete date
{Command[Reference date]}	Dynamic date based on an explicitly indicated date

Dynamic Values

- **Modification of terms for dates**
  - The basic date can be incrementally and decrementally modified by 0 – *n*
  - German and English syntaxes are supported
  - {Command operator number unit}



Syntax	Description, valid values
Command	Valid date expressions in *TOSCA Commander™*
Operator	**+** or **-**
Number	Number by which it is modified
Unit	**Y** … Years, **M** … Months, **D** … Days, **W** … Working days The unit must be entered in decreasing order (Y, M, D)

Dynamic Values

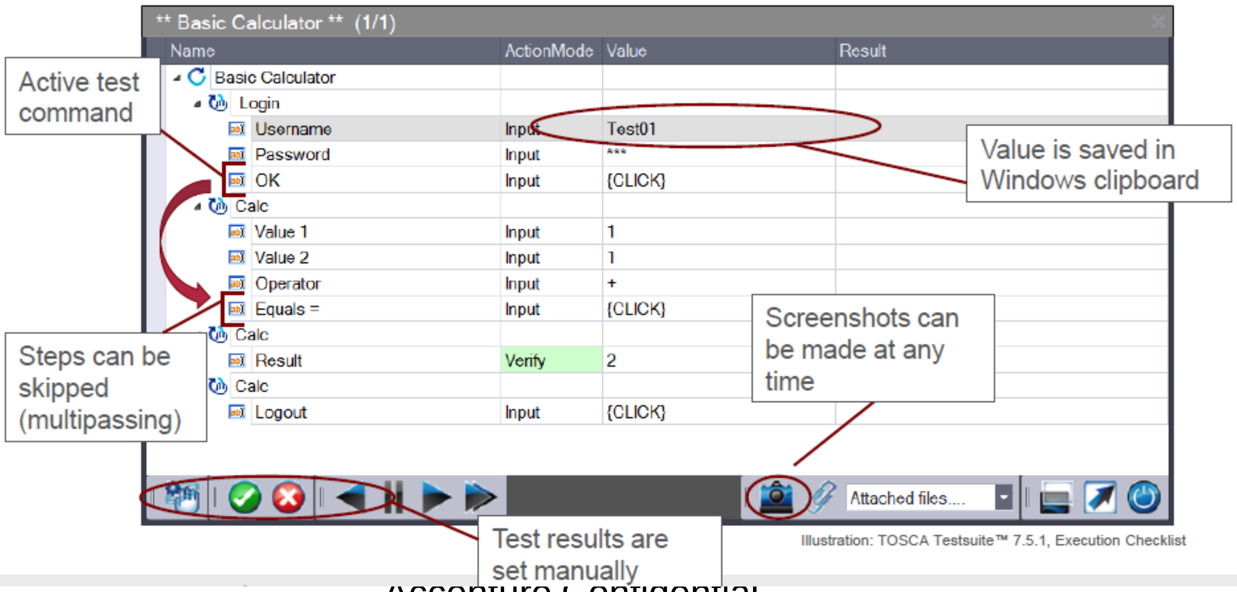
Syntax	Result
`{Ldate}`	15 April 2010
`{Date+1M+1D}`	16/05/2010
`{LDay}`	Thursday
`{LMonthfirst+35D}`	06 May 2010
`{LMonth+1M}`	May
`{Date[15/04/2010]+3D}`	18/04/2010
`{LMonthfirst[15/04/2010]+3D}`	04 April 2010
`{Quarterfirst[{Date[15/04/2010]}]+3M}`	01/07/2010 (first day of next quarter)
`{Trimesterfirst[{Date[15/04/2010]}]+4M}`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

Exercise

Please do Exercise 12. Refer to the Activities for instructions.

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**..

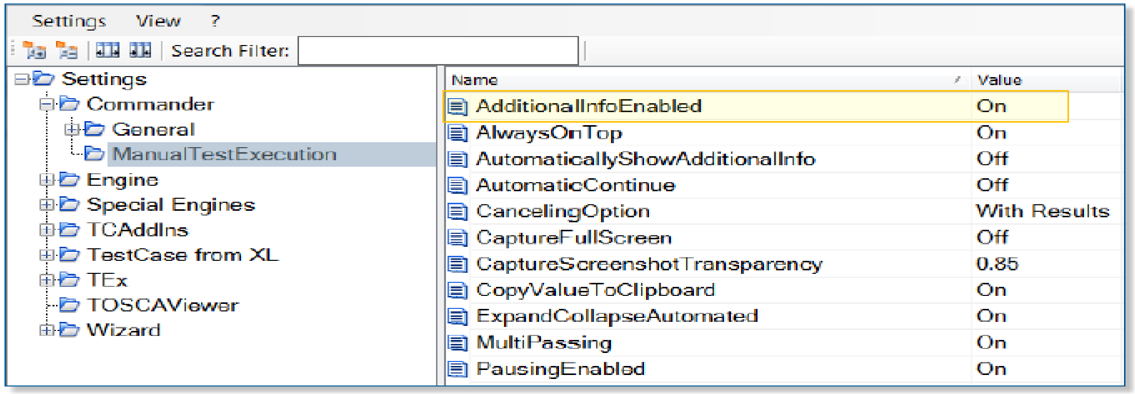

** Basic Calculator ** (1/1)
Active test command
Value is saved in Windows clipboard
Steps can be skipped (multipassing)
Screenshots can be made at any time
Test results are set manually
Attached files....

Name	ActionMode	Value	Result
Basic Calculator			
Login			
Username	Input	Test01	
Password	Input	***	
OK	Input	{CLICK}	
Calc			
Value 1	Input	1	
Value 2	Input	1	
Operator	Input	+	
Equals =	Input	{CLICK}	
Calc			
Result	Verify	2	
Calc			
Logout	Input	{CLICK}	

Illustration: TOSCA Testsuite™ 7.5.1, Execution Checklist

Manual Test Execution

- The following settings are recommended based on best practices:



Name	Value
AdditionalInfoEnabled	On
AlwaysOnTop	On
AutomaticallyShowAdditionalInfo	Off
AutomaticContinue	Off
CancelingOption	With Results
CaptureFullScreen	Off
CaptureScreenshotTransparency	0.85
CopyValueToClipboard	On
ExpandCollapseAutomated	On
MultiPassing	On
PausingEnabled	On

- **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 possibilty of working on different objects in the Common Repository at the same time.
- an be managed to determine that a single object can only be worked Access con 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:

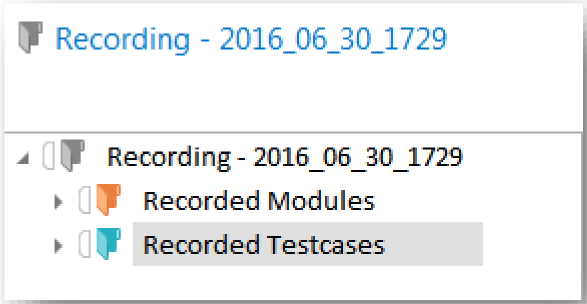
- ▯ Newly created object
- 🟥 Object checked out by another user
- 🟩 Object checked out by the user
- ▯ Object excluded from synchronization

Objects which have the status *checked in* in the Common Repository are grayed out in the Multiuser workspace.

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.



accenture

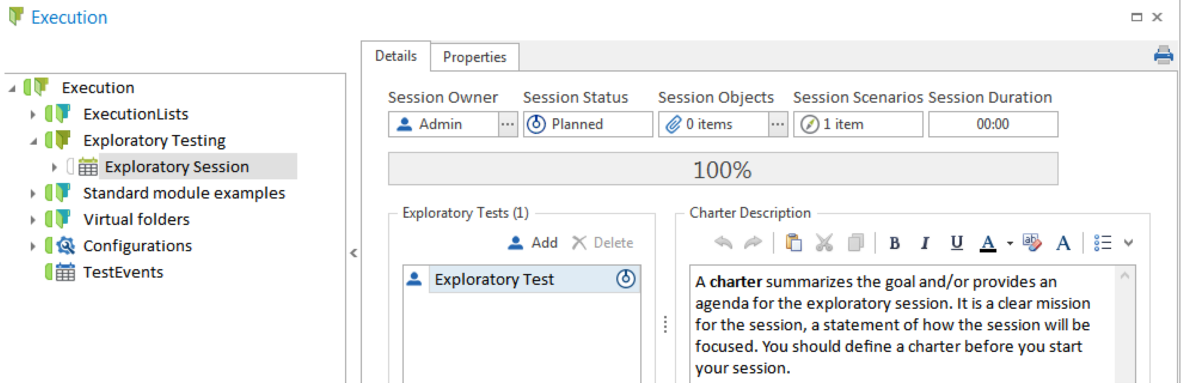
Exploratory Testing

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.



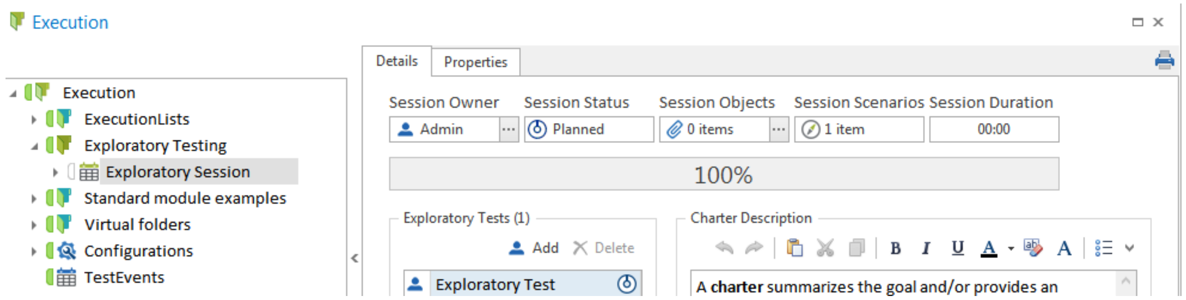
Session Owner	Session Status	Session Objects	Session Scenarios	Session Duration
Admin	Planned	0 items	1 item	00:00

100%

A **charter** summarizes the goal and/or provides an agenda for the exploratory session. It is a clear mission for the session, a statement of how the session will be focused. You should define a charter before you start your 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.



Session Owner	Session Status	Session Objects	Session Scenarios	Session Duration
Admin	Planned	0 items	1 item	00:00

100%

Exploratory Tests (1): Exploratory Test

Charter Description: A charter summarizes the goal and/or provides an

Exploratory Testing

Creating and editing exploratory tests

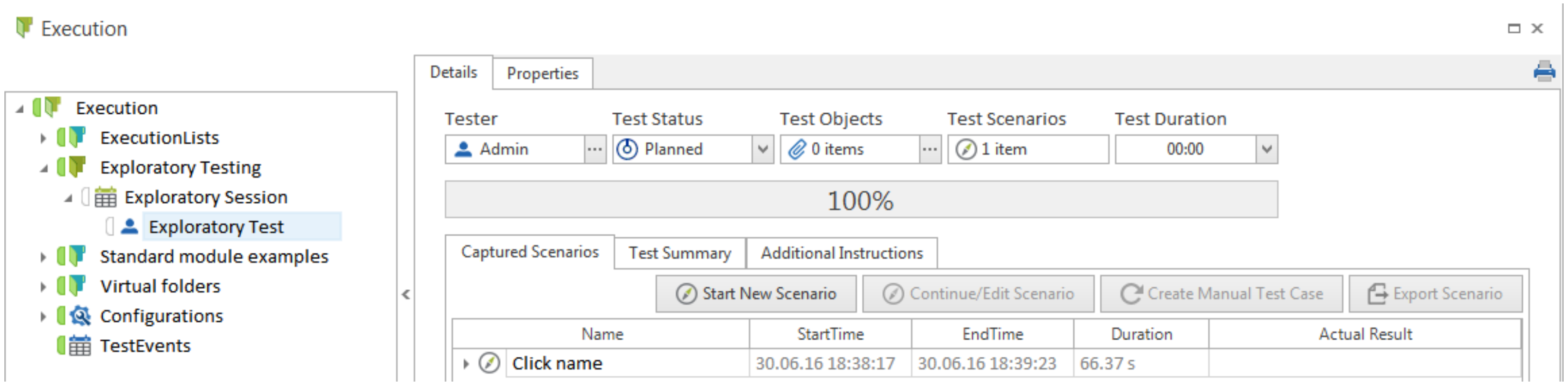
- Exploratory tests are created for each tester.
- In exploratory testing you capture scenarios, you write a test summary, and you may receive further instructions from the session owner.

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.
- 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.



Tester	Test Status	Test Objects	Test Scenarios	Test Duration
Admin	Planned	0 items	1 item	00:00

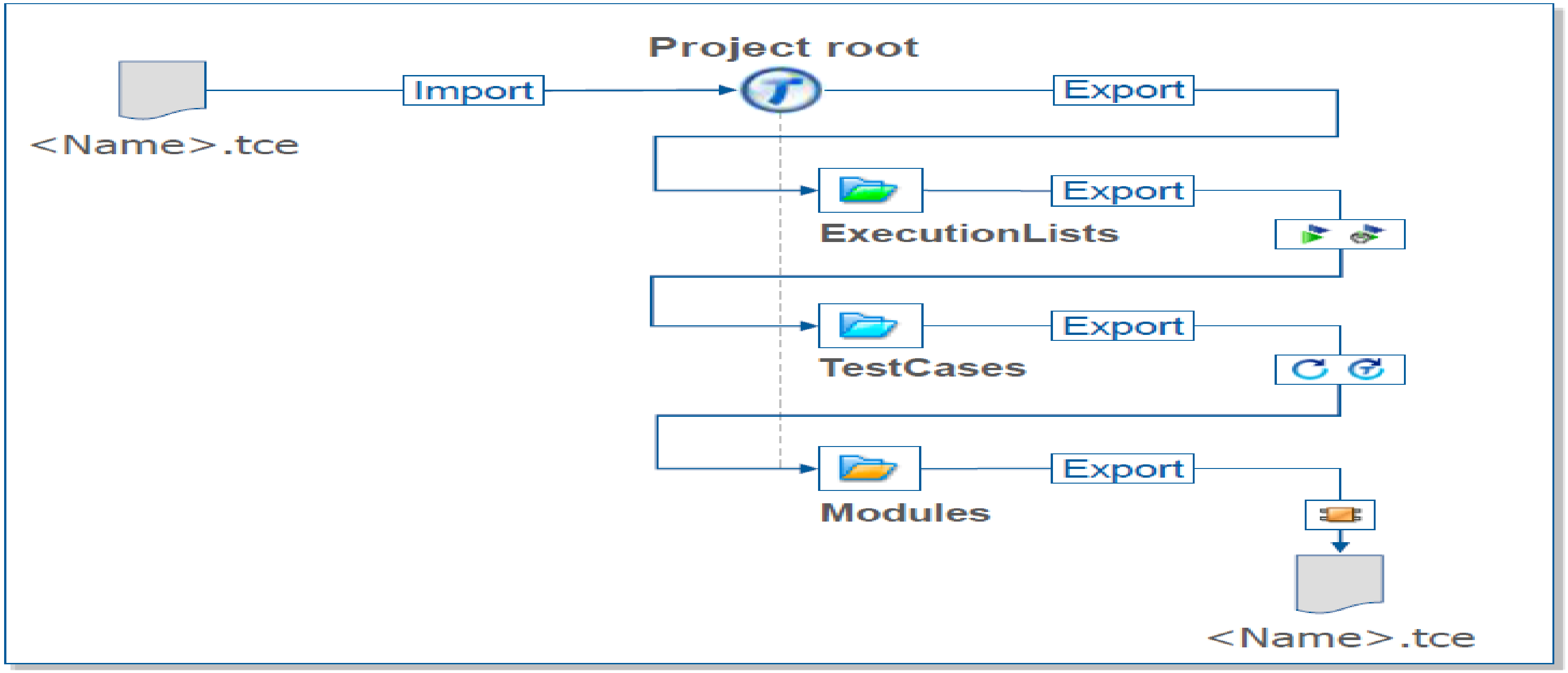
100%

Name	StartTime	EndTime	Duration	Actual Result
Click name	30.06.16 18:38:17	30.06.16 18:39:23	66.37 s	

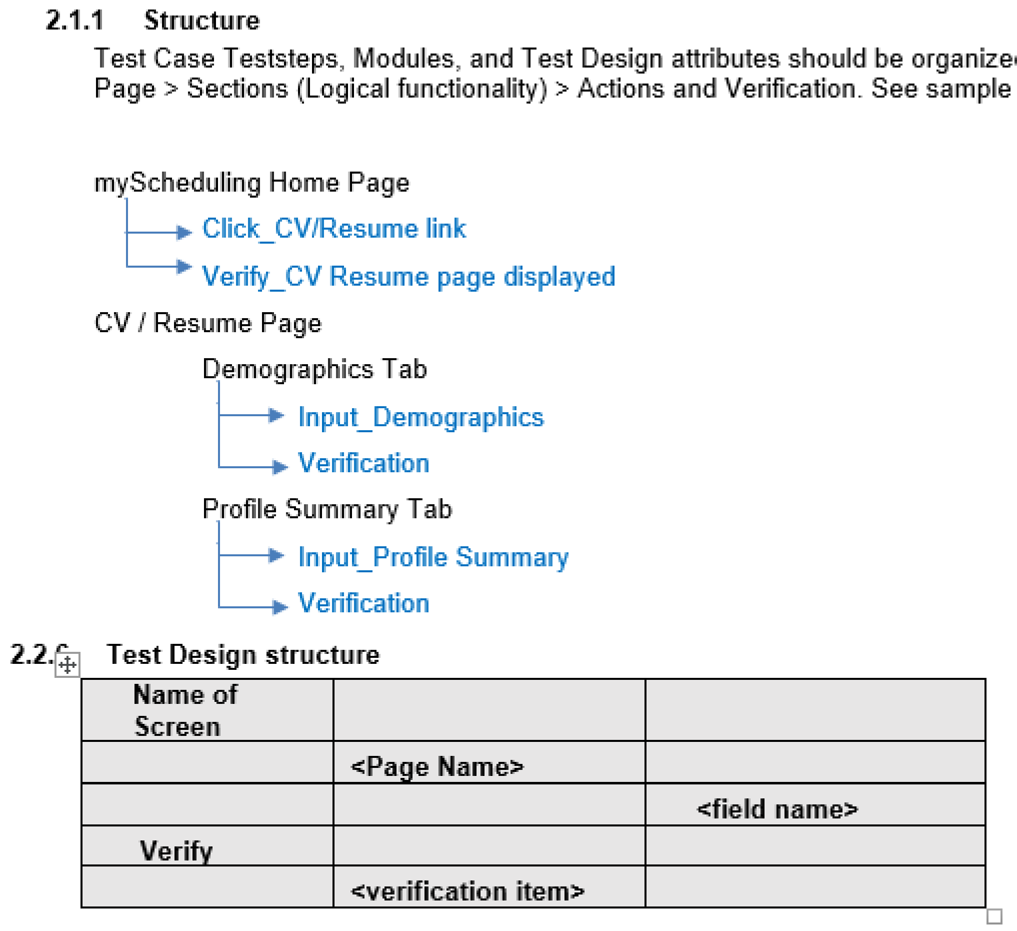
Export and Import of Subsets

- 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



Best Practices



2.1.1 Structure

Test Case Teststeps, Modules, and Test Design attributes should be organized on a folder according to Page > Sections (Logical functionality) > Actions and Verification. See sample illustration below:

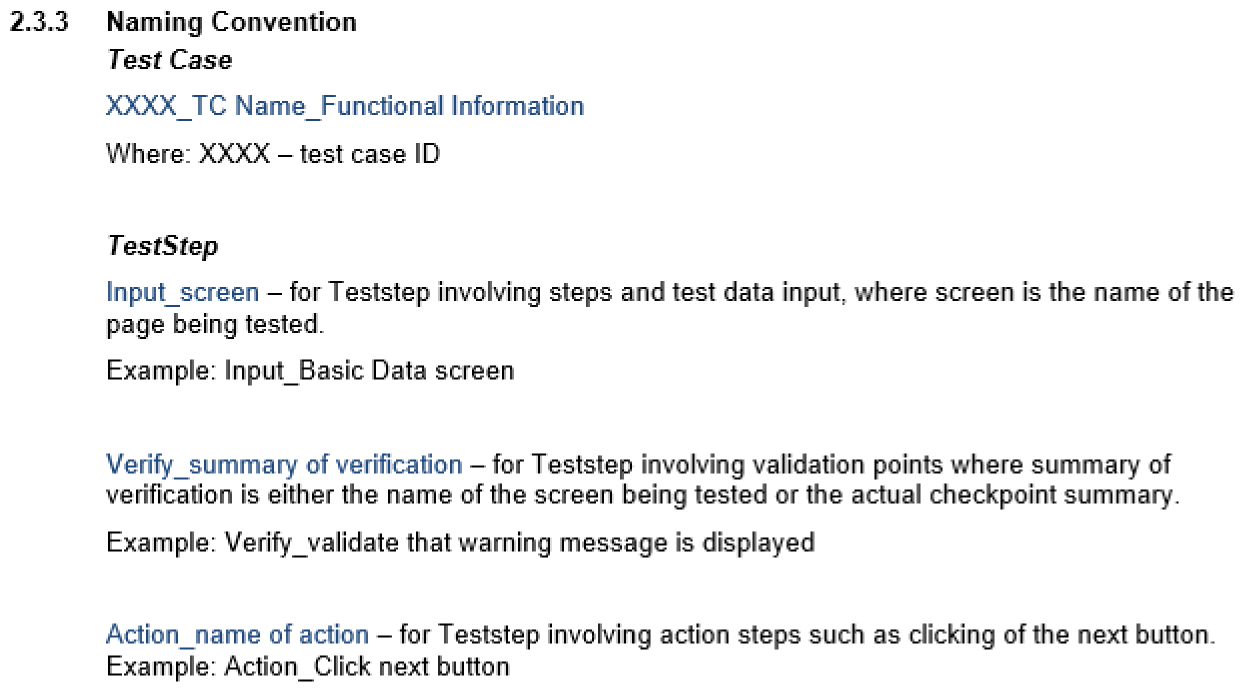
mySchedulling Home Page
- Click_CV/Resume link
- Verify_CV Resume page displayed

CV / Resume Page
- Demographics Tab
  - Input_Demographics
  - Verification
- Profile Summary Tab
  - Input_Profile Summary
  - Verification

2.2.6 Test Design structure

Name of Screen		
	<Page Name>	
		<field name>
Verify		
	<verification item>	

Best Practices



2.3.3 Naming Convention

Test Case

XXXX_TC Name_Functional Information

Where: XXXX – test case ID

TestStep

Input_screen – for Teststep involving steps and test data input, where screen is the name of the page being tested.

Example: Input_Basic Data screen

Verify_summary of verification – for Teststep involving validation points where summary of verification is either the name of the screen being tested or the actual checkpoint summary.

Example: Verify_validate that warning message is displayed

Action_name of action – for Teststep involving action steps such as clicking of the next button.
Example: Action_Click next button

accenture

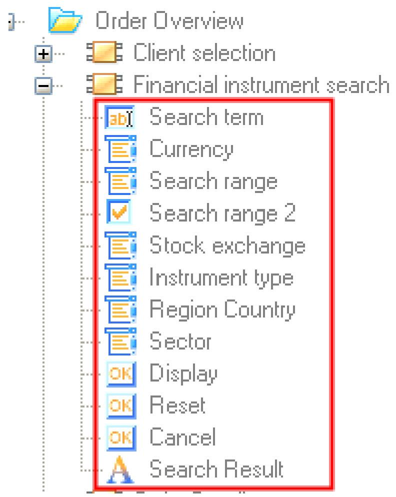
Best Practices

2.2.5 Test Data

Test Data should not be hard coded, use either the TOSCA Test Design component or Excel template, except if you are working on single test case (not template) or if teststep is part of a teststepblock.

2.4.1 Module Name

Rename scanned module name and attributes into its corresponding functional description.



Order Overview
- Client selection
- Financial instrument search
  - Search term
  - Currency
  - Search range
  - Search range 2
  - Stock exchange
  - Instrument type
  - Region Country
  - Sector
  - Display
  - Reset
  - Cancel
  - Search Result

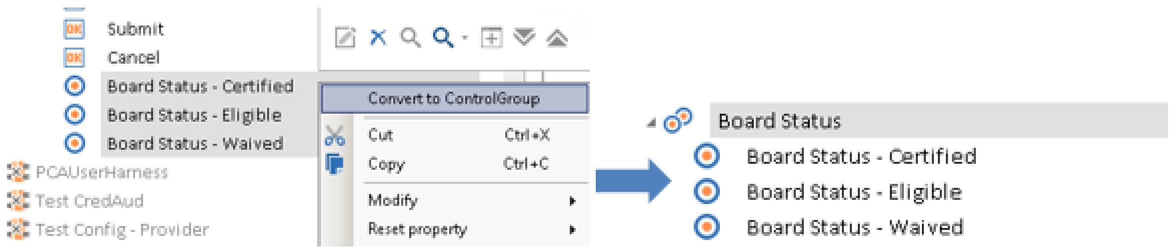
2.4.2 Module Maintenance

- Maintain only one module per page being tested. Do not duplicate modules of the same page. (smaller sections can be combined together and module)

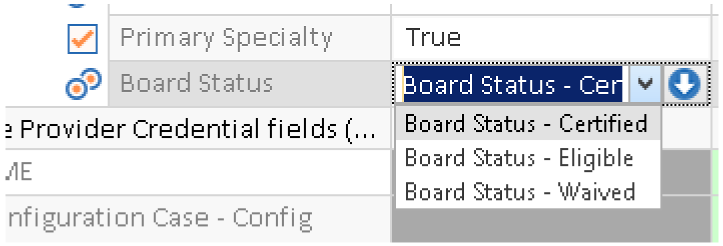
Best Practices

2.4.3 Control Groups

- Controls of the same type can be grouped into ControlGroups. To do this, right click on the controls that you want to be grouped and then select **Convert to ControlGroup** on the context menu.

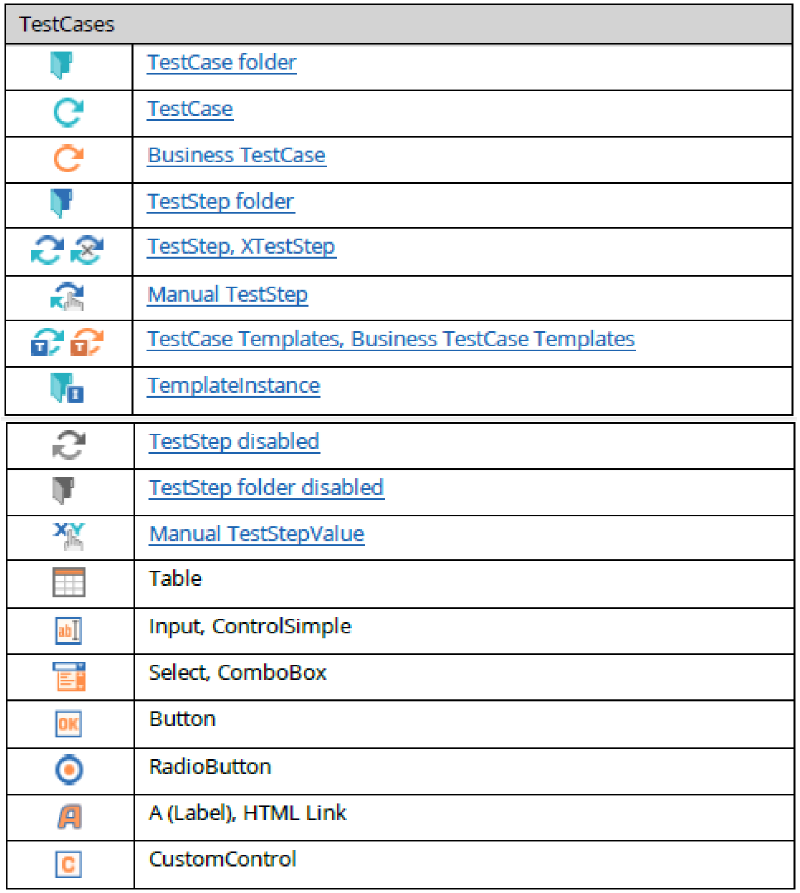


- Once several controls were combined in a single ControlGroup, the individual group items can now be selected from a drop-down menu.

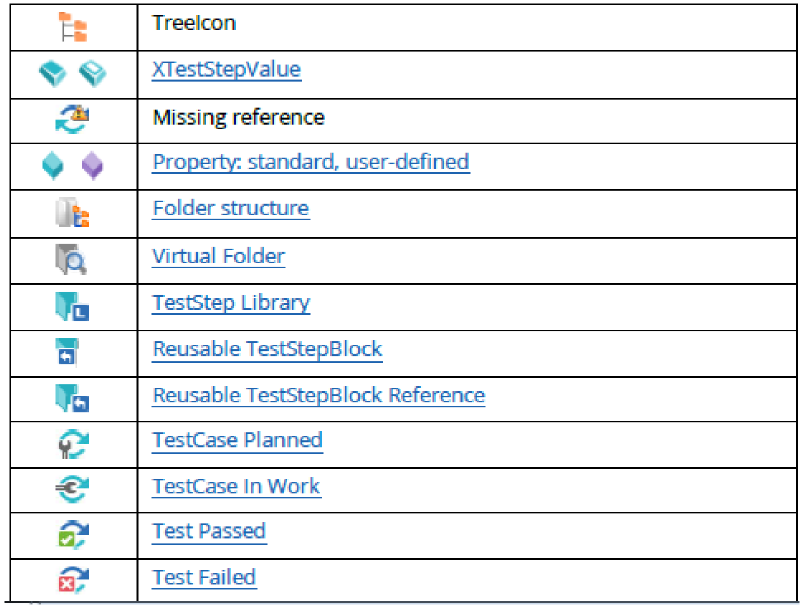


accenture

Icons – Test Cases

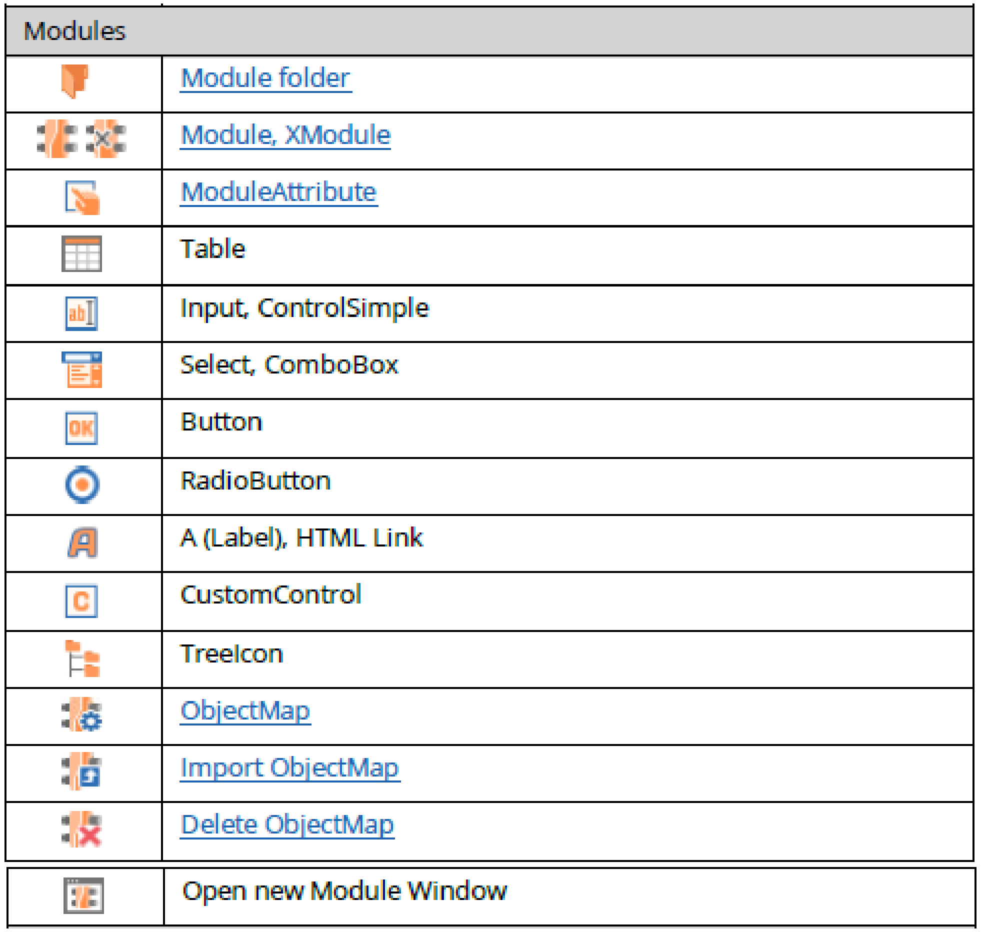


	TestCases
	TestCase folder
	TestCase
	Business TestCase
	TestStep folder
	TestStep, XTestStep
	Manual TestStep
	TestCase Templates, Business TestCase Templates
	TemplateInstance
	TestStep disabled
	TestStep folder disabled
	Manual TestStepValue
	Table
	Input, ControlSimple
	Select, ComboBox
	Button
	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

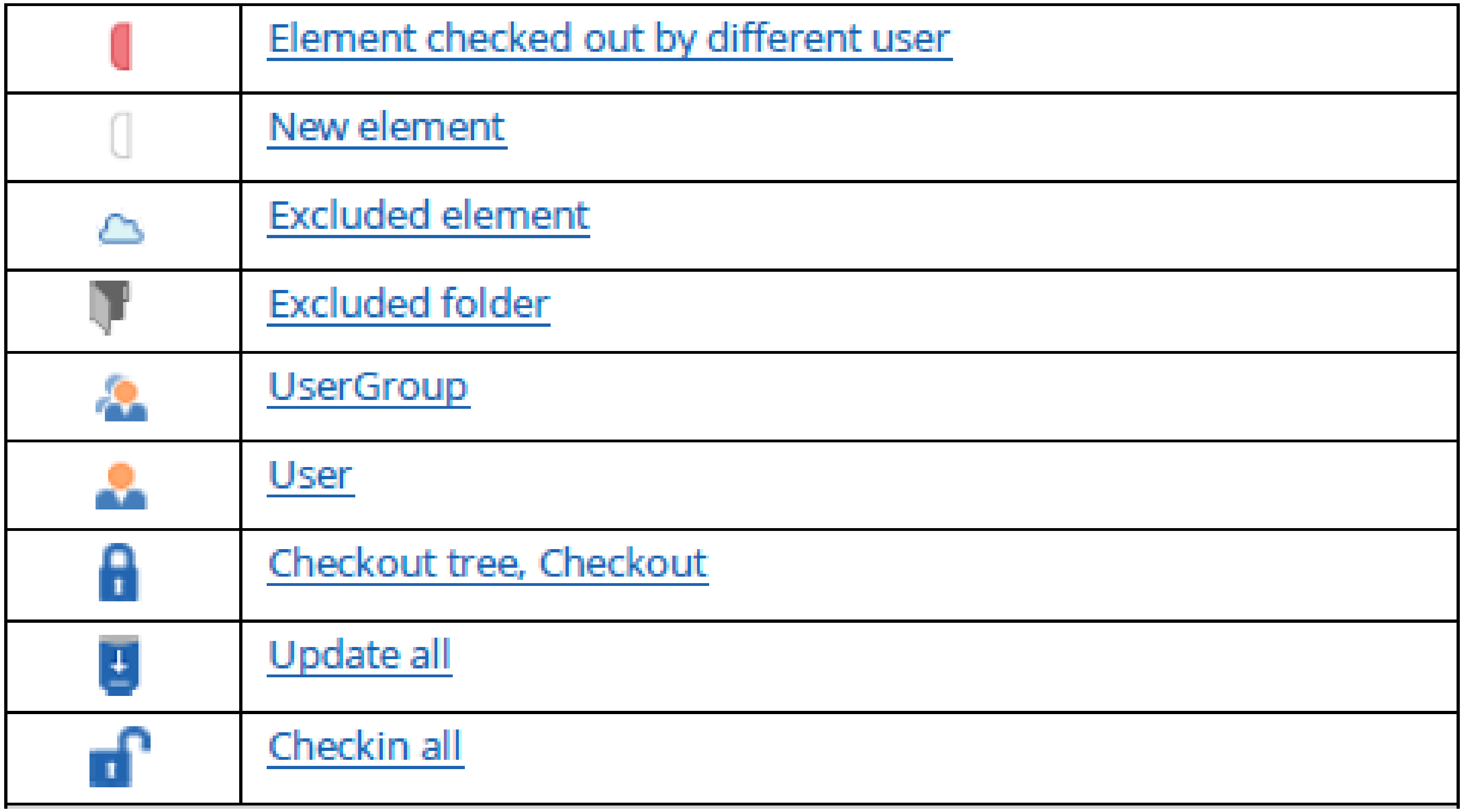
Icons – Modules



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

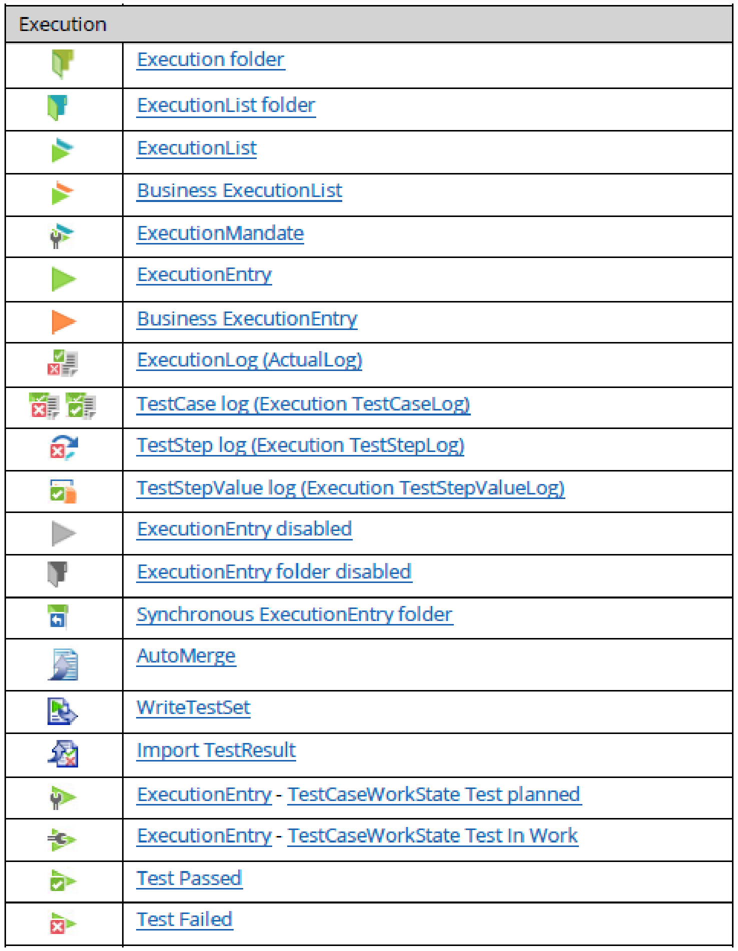
accenture

Icons – Multi User Workspace

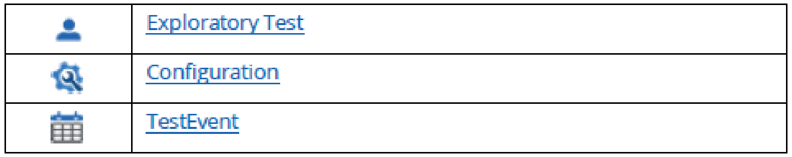


🟥	Element checked out by different user
▯	New element
☁	Excluded element
▮	Excluded folder
👥	UserGroup
👤	User
🔒	Checkout tree, Checkout
⬇	Update all
🔓	Checkin all

Icons – Execution



Execution	
	Execution folder
	ExecutionList folder
	ExecutionList
	Business ExecutionList
	ExecutionMandate
	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
	WriteTestSet
	Import TestResult
	ExecutionEntry - TestCaseWorkState Test planned
	ExecutionEntry - TestCaseWorkState Test In Work
	Test Passed
	Test Failed



	Exploratory Test
	Configuration
	TestEvent