



*High performance. Delivered.*

# TOSCA Automation Tool Training

Accenture Confidential

1

Copyright © 2008 Accenture. All Rights Reserved. Accenture, its logo, and High Performance Delivered are trademarks of Accenture.

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

- Identify the TOSCA Test Suite and TOSCA Commander.
- Identify and define the different components of TOSCA.
- Create Modules, Test Cases and Execution List using the TOSCA Commander.



## Day 1:

- Installation and activation
- Introduction to Tosca Test Suite
- Tosca Commander
- Workspace
- Module
- Xscan
- Test Case
- Table Steering
- Control
- Action Mode
- Random Values
- Dynamic Dates



## Day 2:

- Verify
- Buffer
- Calculations
- Library
- Rescan, Value Range, Module Merge
- WaitOn
- Self Defined TCP
- Business Parameters



## Day 3:

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



- **Tosca Testsuite** is a software developed by Tricentis which allows automated, functional software to be tested.
- **Tosca Commander™** is the core of Tricentis Tosca Testsuite is.
- **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).



## 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.
- TestCases
  - Series of TestSteps that run from a defined starting point to a defined ending point.

Icon	Description
	TestCase folder
	TestCase
	TestStep, XTestStep





- 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



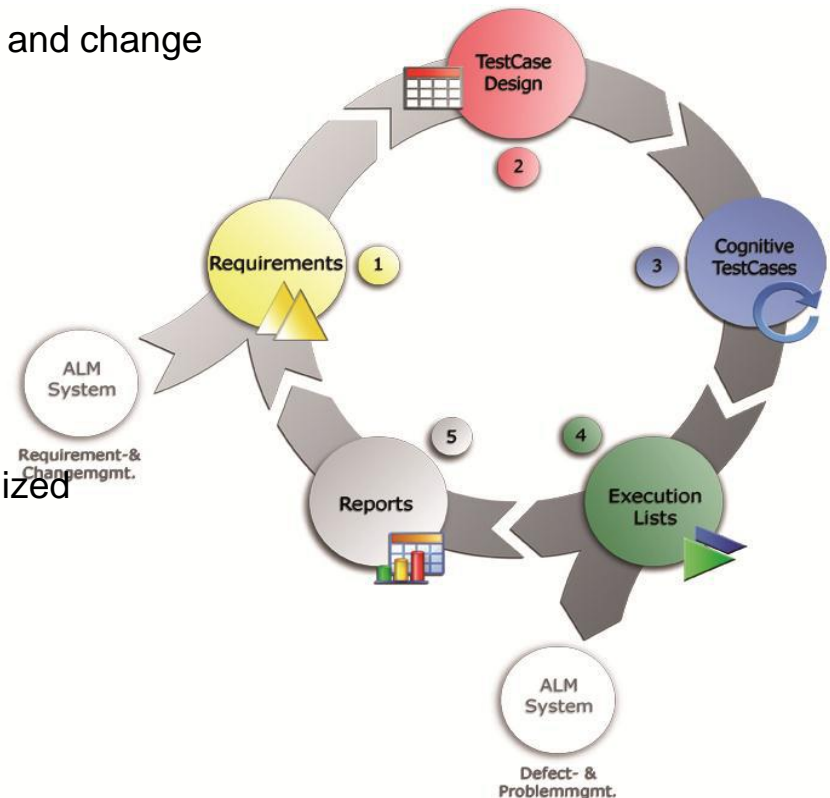
LogEntry



ExecutionEntryFolder

# 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

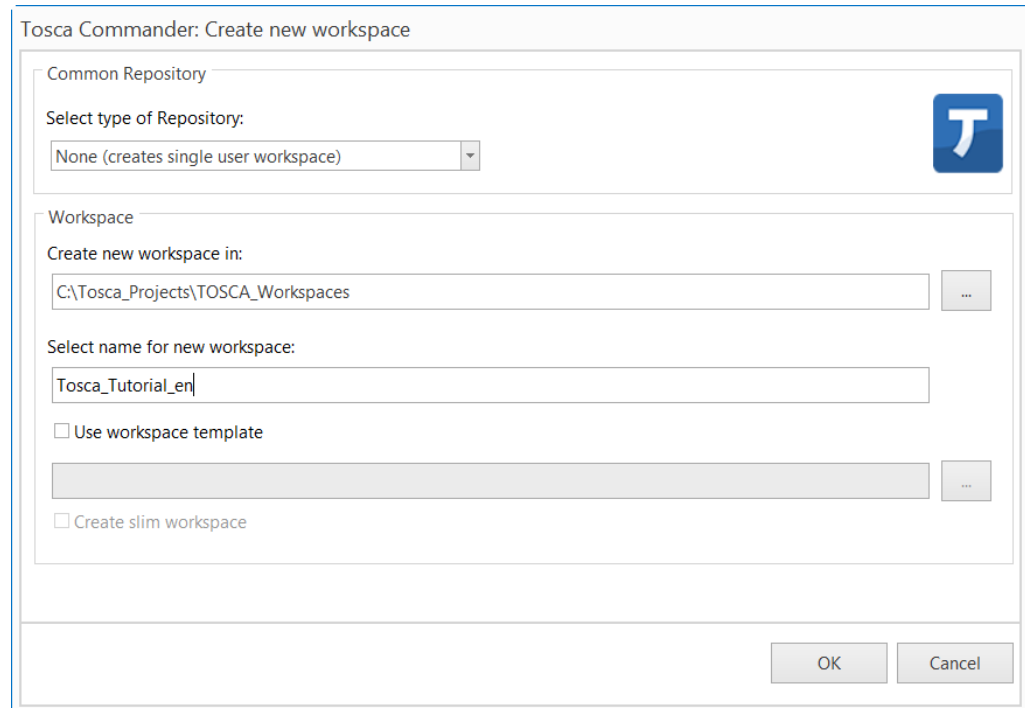




# TOSCA COMMANDER

## 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\_Tutorial\_en

☐ Use workspace template

☐ Create slim workspace

OK Cancel

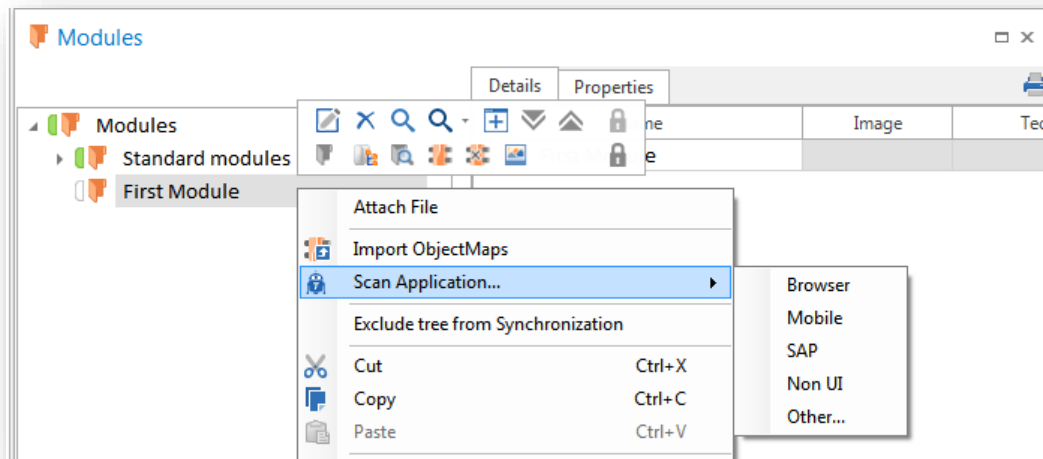


## Creating Module

- 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.

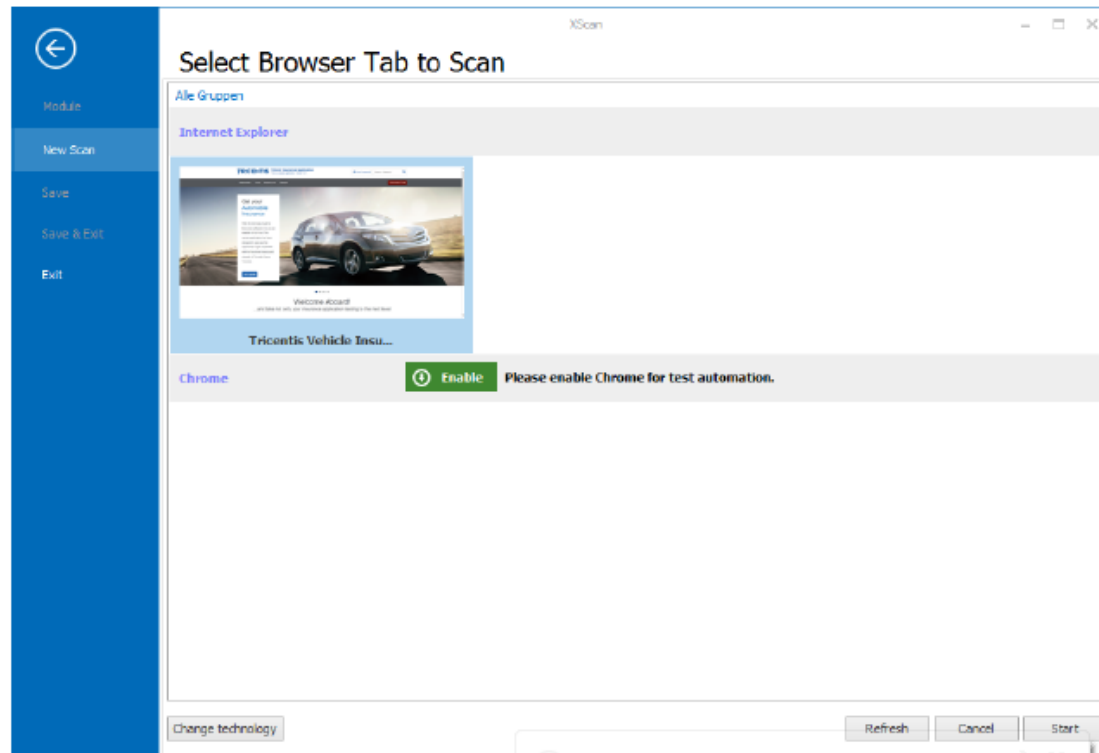
## Xscan

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...>Browser** from the context menu.



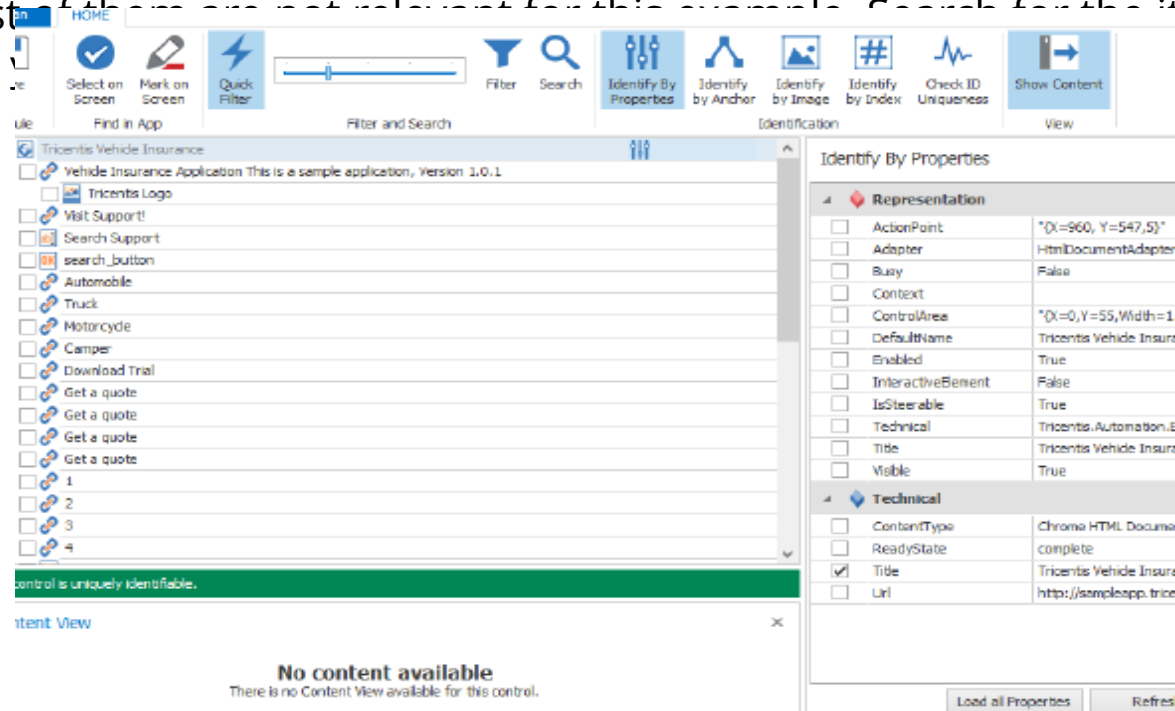
# TOSCA Commander - Module

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 **S**



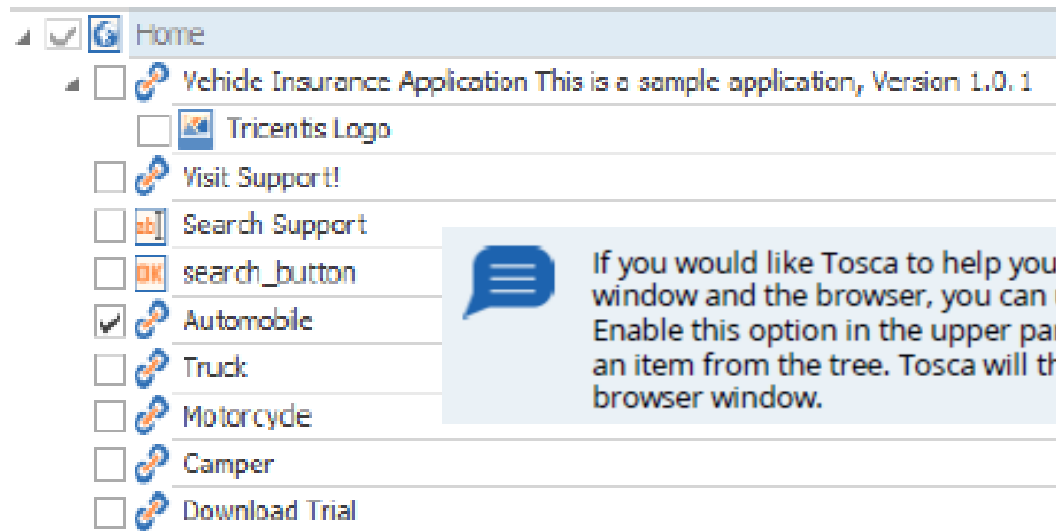
# TOSCA Commander - Module

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





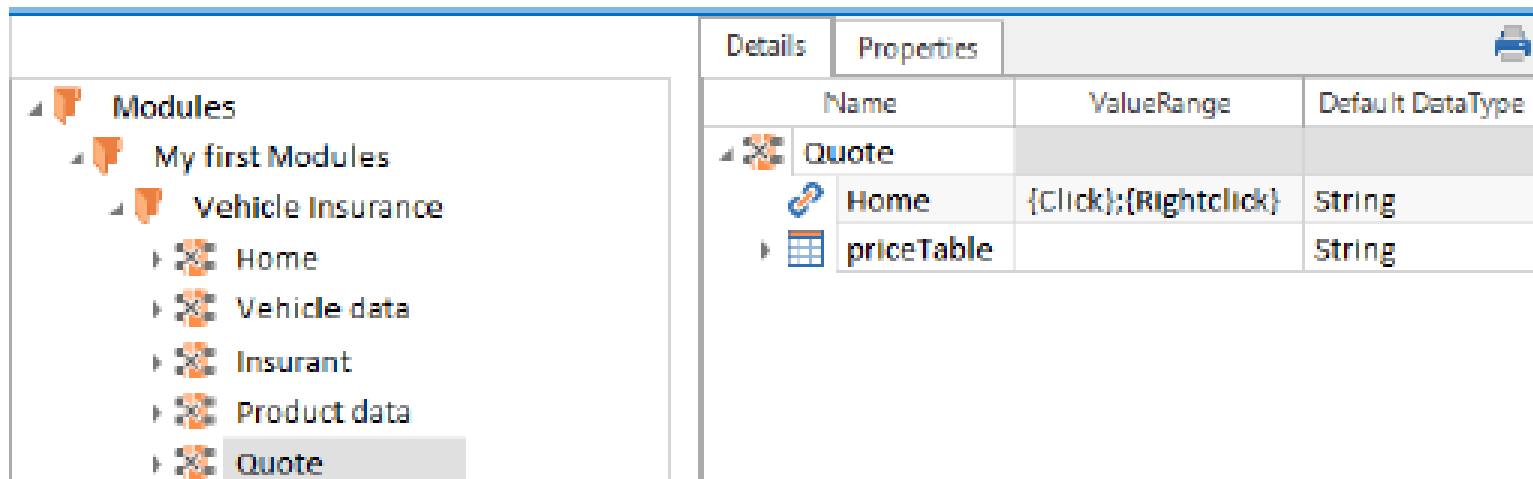
8. Enable the checkbox in front of the items that needs to be included in the test case



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.

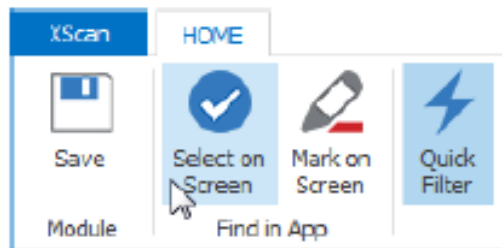
# TOSCA Commander - Module

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.

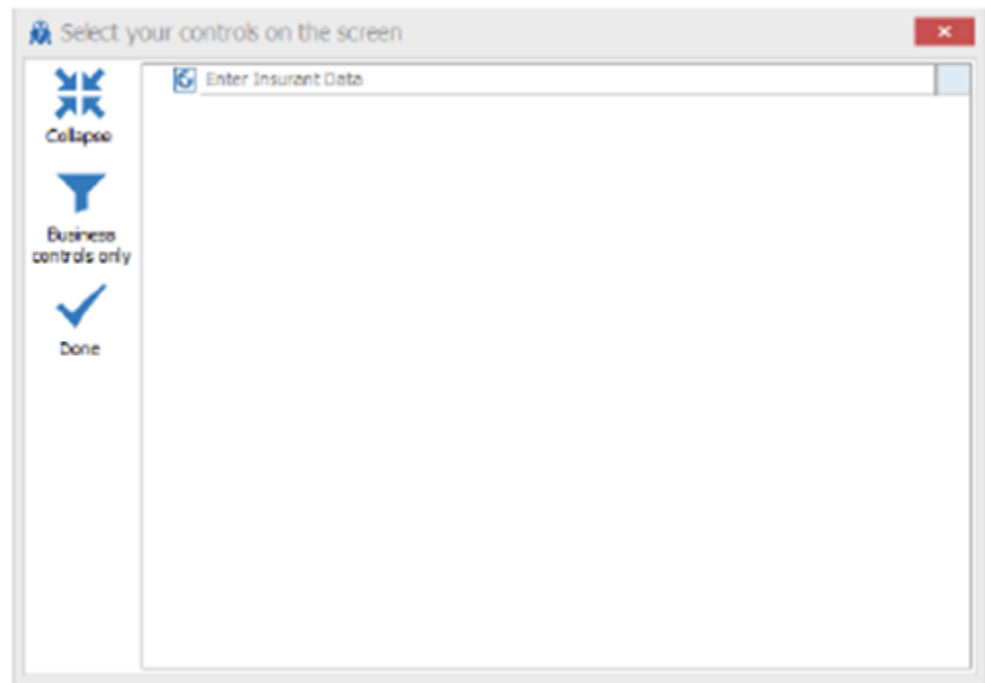


## Xscan Using Select on Screen

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**



## ***Exercise Getting Started***

- *Goal: Familiarization of the SUT and Workspace Creation*
- *Task*



Adobe Acrobat  
Document

## *Exercise 1 Module Creation*

- *Goal*
  - **Create Module through Xscan**
- *Task*



Adobe Acrobat  
Document








## 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.

## ***Exercise 2 Test Case Structure and Browser TCP***





- *Goal*
  - **Familiarize yourself with the Test Case Structure and Browser TCP**
- *Task*



Adobe Acrobat  
Document





- **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.

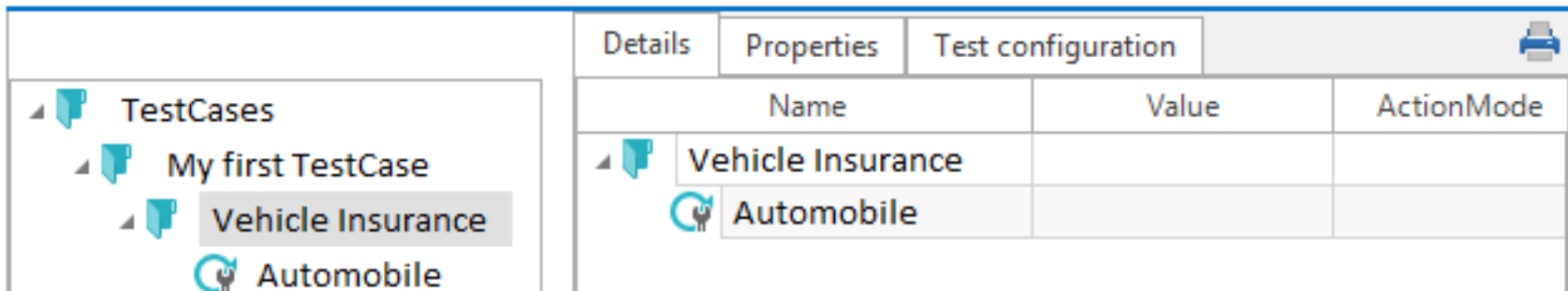


TestStepValues which are grayed out due to lacking values can either be hidden from view or shown via the **F9** key.



## Creating TestCase

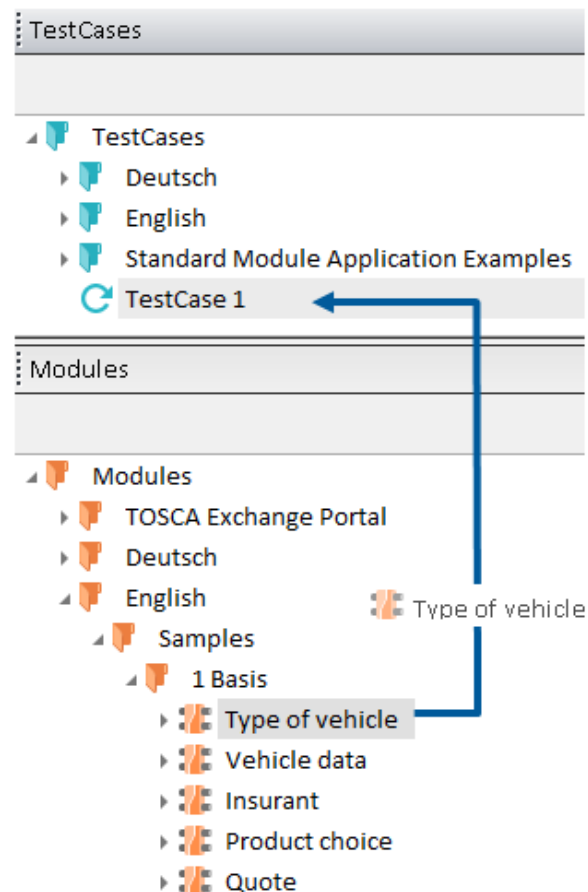
- Right-click on the TestCase folder **Vehicle Insurance** and select **Create TestCase** from the mini toolbar.
- Assign the name (e.g. **Automobile**) to the new TestCase.



Details		Properties	Test configuration	
Name		Value	ActionMode	
Vehicle Insurance				
Automobile				

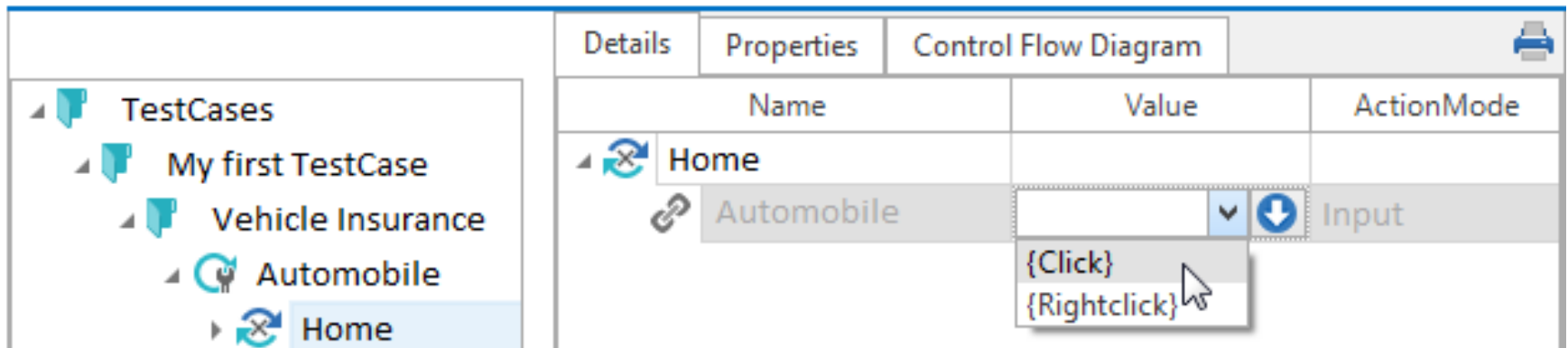
## Creating Automated Test Steps

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



## 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,



Details			Properties	Control Flow Diagram	
Name	Value	ActionMode			
Home					
Automobile	<input type="text"/> <div> {Click} {Rightclick} </div>	Input			



- 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.

## *Exercise 3 Open and Close the Application*

- **Goal**
  - Familiarization with the Tbox Engine, test case creation and the Scratchbook.
- **Task**



Adobe Acrobat  
Document



- 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>

- 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: `"="` `"!="` `">"`





- 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.

Details	Properties	Control Flow Diagram	
Name	Value	ActionMode	
Vehicle data			
Make	Mazda	Input	
[kW]	65	Input	
Date of Manufacture	1/11/2012	Input	
Number of Seats	5	Input	
Fuel Type	Petrol	Input	
List Price	19000	Input	
Annual Mileage	21000	Input	
Next »	{Click}	Input	

ActionMode	Color	Description
DoNothing	white	With the ActionMode <b>DoNothing</b> no action is performed. The content of the field <b>Value</b> has no influence. The test object specified by the Module is steered, but not the control.
Input	white	<b>Input</b> is used for steering individual controls. This includes the input of a value, the operation of a control panel or a <b>ComboBox</b> , etc. The specific input mode is determined by the field <b>Value</b> .
Verify	green	Allows the value of a control provided by the application to be verified as specified in the field <b>Value</b> . Control properties can also be verified.
Buffer	yellow	If input values are repeatedly needed they can be saved with the ActionMode <b>Buffer</b> .
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™ <ul style="list-style-type: none"> <li>if the TestCase was executed in an ExecutionList.</li> <li>in the column <b>Used Value</b> in the details view of the ExecutionList.</li> </ul>
WaitOn	orange	The ActionMode <b>WaitOn</b> interrupts the execution of the TestCase until the respective control has the value or property that has been specified in the <b>Value</b> field.



## 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			
ab	Prefix	01	Input	String
ab	Number	{RND[12]}	Input	String

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

## 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

## *Exercise 4 Test Step Values*

- **Goal**
  - Familiarization with object steering and random values in a test case.
- **Task**



Adobe Acrobat  
Document

## 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 milliseconds)	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



## 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)

## *Exercise 5 Dynamic Dates*

- **Goal**
  - Familiarization on inserting dynamic dates in the test case.
- **Task**







Adobe Acrobat  
Document





- 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.

 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	<code>.enabled=true</code>	Verify

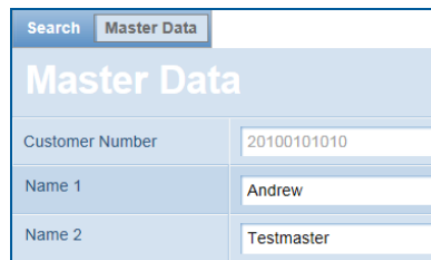
The following control properties are available by default for verify operations:

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

The following control properties are available for list box

Name	Description
<code>.contains</code>	This verifies whether a specific content exists in a ComboBox or a ListView.
<code>.list</code>	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 ;.

- 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.



Search Master Data

### Master Data

Customer Number	20100101010
Name 1	Andrew
Name 2	Testmaster

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	DataType
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	1000000007		
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





Tosca allows calculations to be performed with the expressions MATH.

## 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

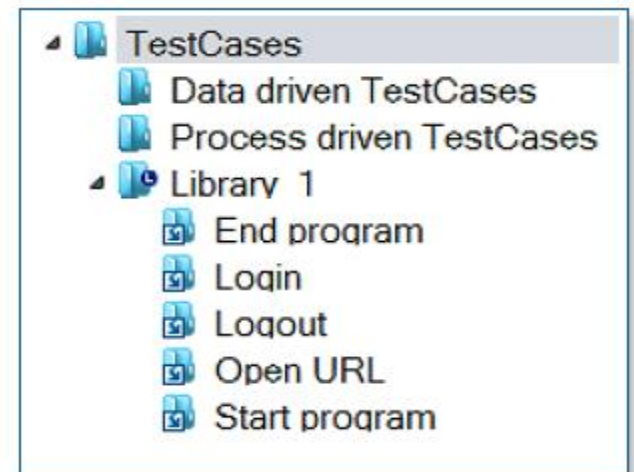
## *Exercise 6*

- *Goal*
  - Familiarization of Buffer, Verify and Calculated Value.
- *Task*



Adobe Acrobat  
Document

- 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**).



## *Exercise 7*

- *Goal*
  - Create Library and reusable test step blocks
- *Task*













Adobe Acrobat  
Document

# 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.

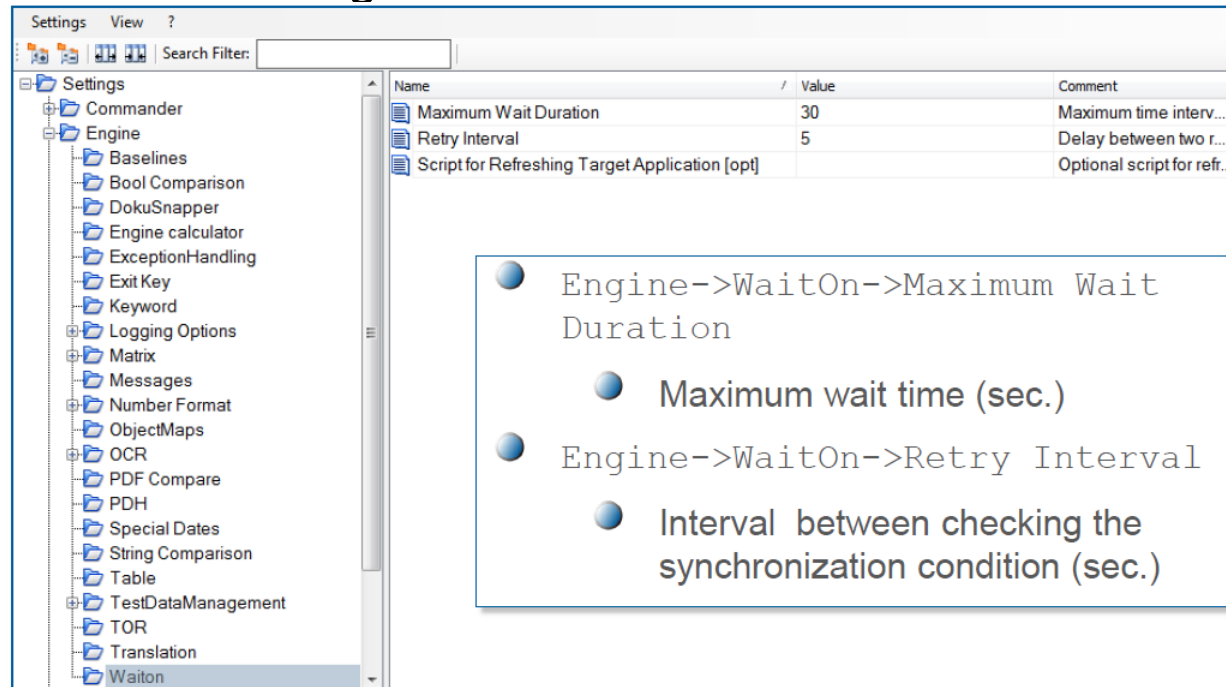
## Value Range

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



- 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



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

Illustration: TOSCA Commander™ Settings

Accenture Confidential

## ***Exercise 8***

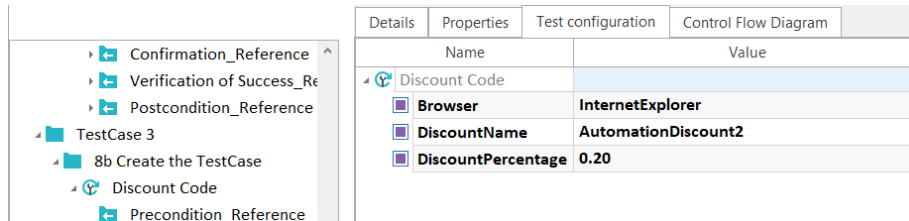
- *Goal*
  - Rescan, Value Range, Module Merge, WaitOn
- *Task*



Adobe Acrobat  
Document

## 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.



Name	Value
Discount Code	
Browser	InternetExplorer
DiscountName	AutomationDiscount2
DiscountPercentage	0.20

Shopping Cart Procedures				
Shopping cart products			String	
Update shopping cart			String	
Enter discount code	{CP[DiscountName]}	Input	String	

## ***Exercise 9***

- *Goal*
  - Familiarization of TCP
- *Task*

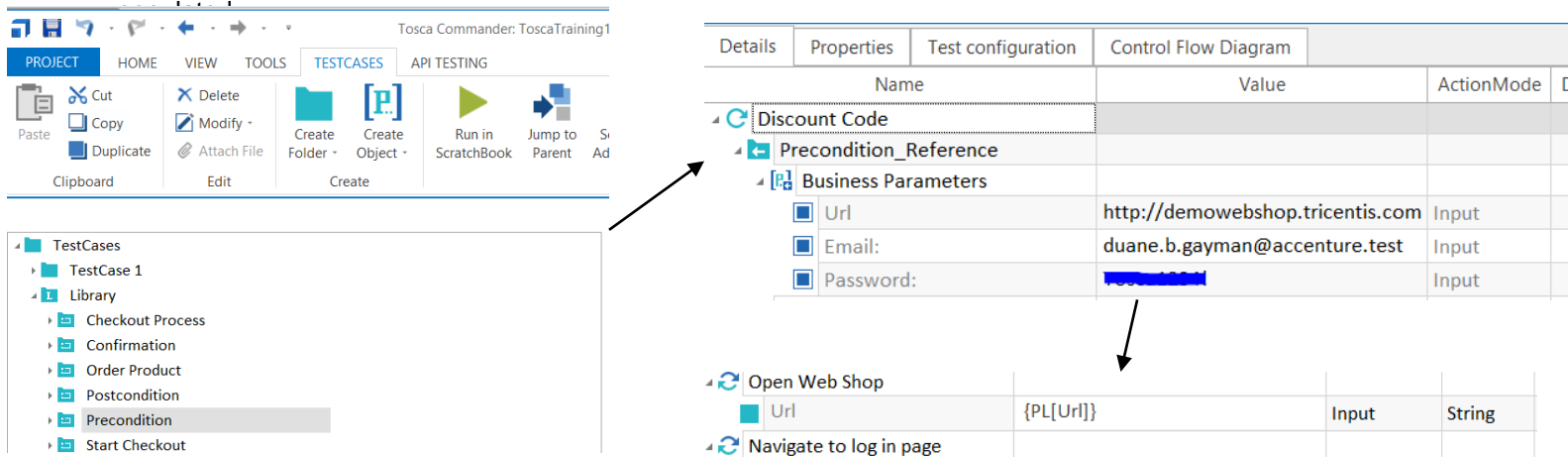


Adobe Acrobat  
Document

## 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



The screenshot shows the Tosca Commander interface with the 'TESTCASES' tab selected. The left pane shows a tree view of TestCases, including 'Test Case 1' and a 'Library' containing 'Checkout Process', 'Confirmation', 'Order Product', 'Postcondition', 'Precondition', and 'Start Checkout'. The 'Precondition' item is selected. The right pane shows the 'Details' tab for the 'Precondition' item, which is a 'Precondition\_Reference' block. Below this, the 'Business Parameters' section is expanded, showing a table with the following data:

Name	Value	ActionMode	Control Flow Diagram
Discount Code			
Precondition_Reference			
Business Parameters			
Url	http://demowebshop.tricentis.com	Input	
Email:	duane.b.gayman@accenture.test	Input	
Password:	*****	Input	

Below the Business Parameters table, there are two more rows in the table:

Open Web Shop			
Url	{PL[Url]}	Input	String
Navigate to log in page			

Arrows indicate the flow from the 'Precondition' item in the left pane to the 'Business Parameters' table in the right pane, and from the 'Business Parameters' table to the 'Open Web Shop' row in the table below.



## ***Exercise 10***

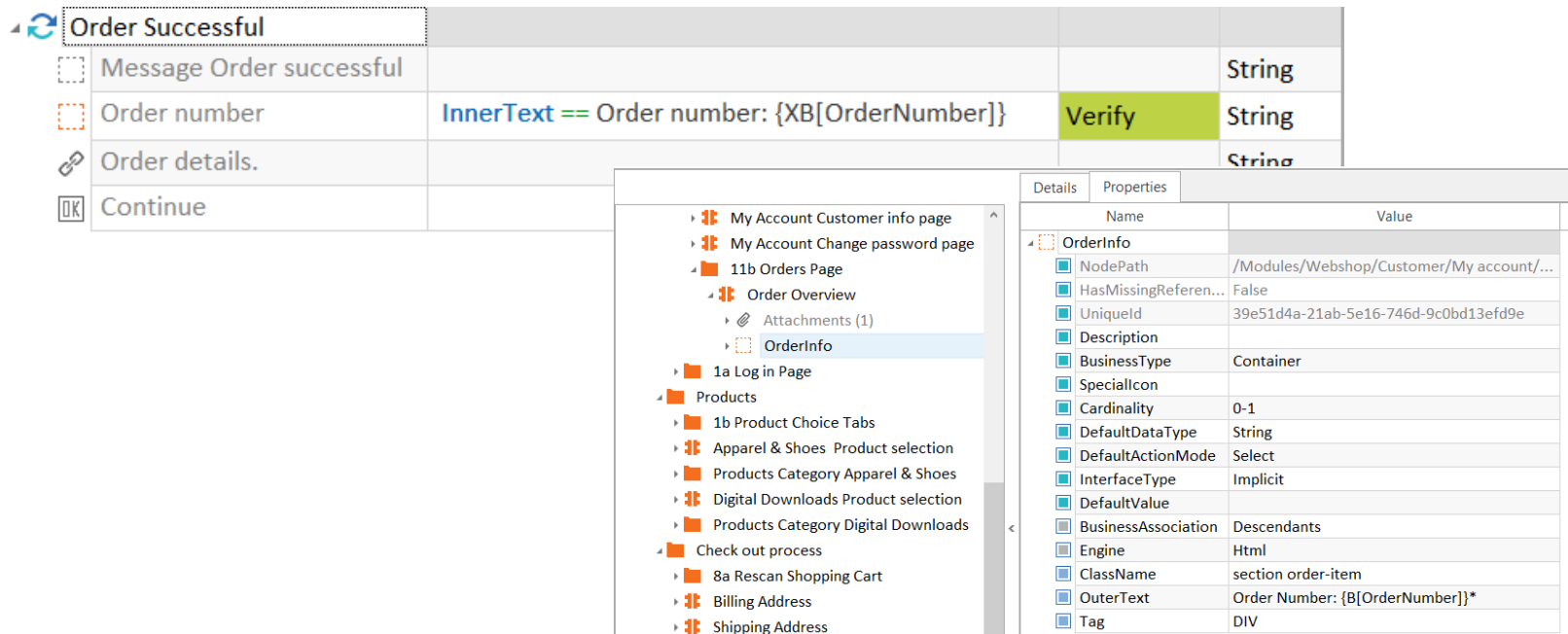
- *Goal*
  - Business Parameters
- *Task*



Adobe Acrobat  
Document

## 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.



The screenshot displays a test automation interface with a table of test steps and a detailed view of a specific step.

Icon	Step Name	Expression	Action	Data Type
Refresh	Order Successful			
Message	Message Order successful			String
Text	Order number	InnerText == Order number: {XB[OrderNumber]}	Verify	String
Link	Order details.			String
OK	Continue			

The detailed view shows the 'OrderInfo' object structure and its properties.

OrderInfo	
Name	Value
NodePath	/Modules/Webshop/Customer/My account/...
HasMissingReferen...	False
Uniqueld	39e51d4a-21ab-5e16-746d-9c0bd13efd9e
Description	
BusinessType	Container
SpecialIcon	
Cardinality	0-1
DefaultDataType	String
DefaultActionMode	Select
InterfaceType	Implicit
DefaultValue	
BusinessAssociation	Descendants
Engine	Html
ClassName	section order-item
OuterText	Order Number: {B[OrderNumber]}*
Tag	DIV

## ***Exercise 11***

- *Goal*
  - Familiarization of dynamic comparisons, parent control and dynamic ID
- *Task*



Adobe Acrobat  
Document

# 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

Details	Properties	Control Flow Diagram	
Name	Value	ActionMode	
Count Number of Or...			
OrderInfo		Select	
OrderTotal	ResultCount → NumberofOrders	Buffer	
Details			

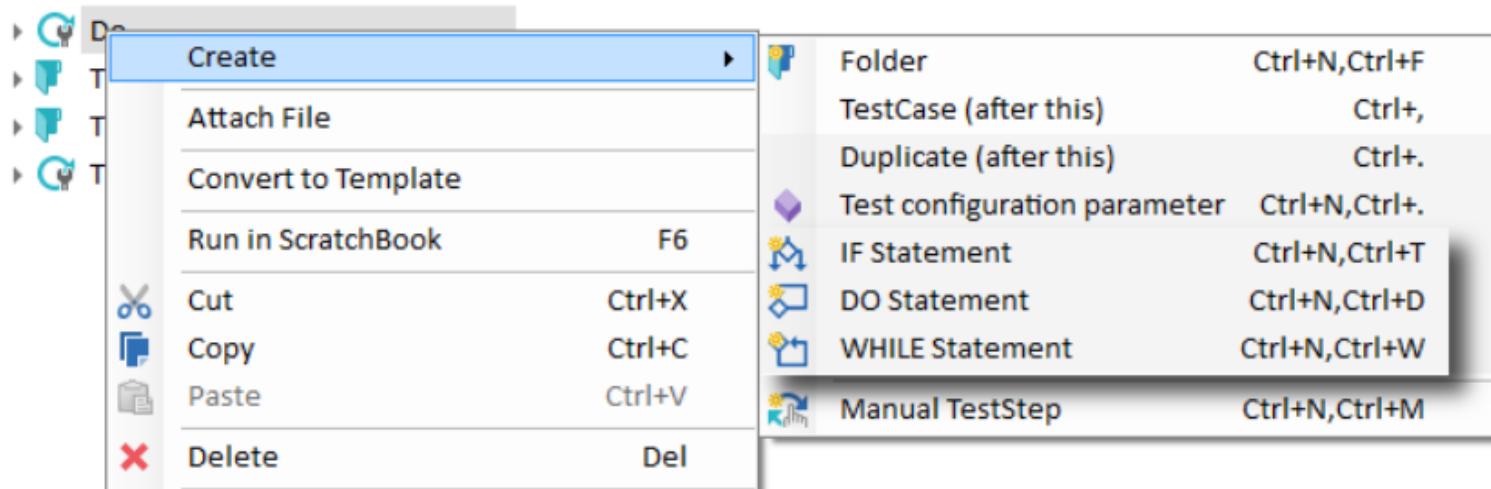
- *Goal*
  - *Explicit Name, Buffer Tbox, ResultCount*
- *Task*



Adobe Acrobat  
Document

# Conditional Statements and Loops

- In Tosca, you can define IF, DO and WHILE statements if you would like to run TestSteps with branches or repeatedly. 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
  - DO Statement
  - WHILE 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)

1. In the first TestStep the buffer **A** including the value 1 is created (Module [TBox Set Buffer](#)).
2. The **Repetition** column defines that the TestCase should be run twice.
3. The condition set in the **Condition** object defines that the value of the buffer **A** must be greater than 0.
4. Since the buffer value is greater than 0, Tosca runs the **Then** TestStep in the first run. As a result, the value of the buffer **A** is changed to 0.
5. In the second run, the same verification is performed. Since the buffer value includes the value 0 after the first run, the TestStep is then run in the **Else** statement. The value 1 is written to buffer **A**.

Name	Value	ActionMode	Data Type	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	

Name	Value	ActionMode	Data Type	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"



## 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. The result of the last repetition is thus negative.

DO statements 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.

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

Order of conditions:

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



- In the first TestStep the buffer **A** including the value **0** is created (Module [TBox Set Buffer](#)).
- The TestStep within the **Loop** object uses a dynamic expression which defines that the value of the buffer **A** should be increased by 1 ([see chapter "Calculations"](#)).
- The condition set in the **Condition** object defines that the value of the buffer **A** must be less than 10.

The TestStep **Set Buffer for tests** is run repeatedly until the buffer **A** has the value 10.

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	

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'



## Creating WHILE statements

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

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

WHILE statements 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 ([see Illustration "MaximumRepetitions property"](#)).

Order of conditions:

- WHILE
- Condition (within the WHILE statement)
- Loop (within the WHILE statement)

1. In the first TestStep the buffer **A** including the value **0** is created (Module [TBox Set Buffer](#)).
2. The condition set in the **Condition** object defines that the value of the buffer **A** must be less than 10.  
Since the buffer **A** contains the value **0**, Tosca runs the TestStep in the **Loop** object.
3. The TestCase within the **Loop** object uses a dynamic expression which defines that the value of the buffer **A** should be increased by 1 ([see chapter "Calculations"](#)).  
The TestStep is run repeatedly until the buffer **A** has the value **10**.

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



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'








Accenture Confidential

- *Goal*
  - *While Statement*
- *Task*



Adobe Acrobat  
Document

- **ExecutionListFolder** 
  - ExecutionList folders are used to structure and manage ExecutionLists.
- **ExecutionList** 
  - ExecutionLists offer a flexible possibility 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

- **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 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'				



# 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

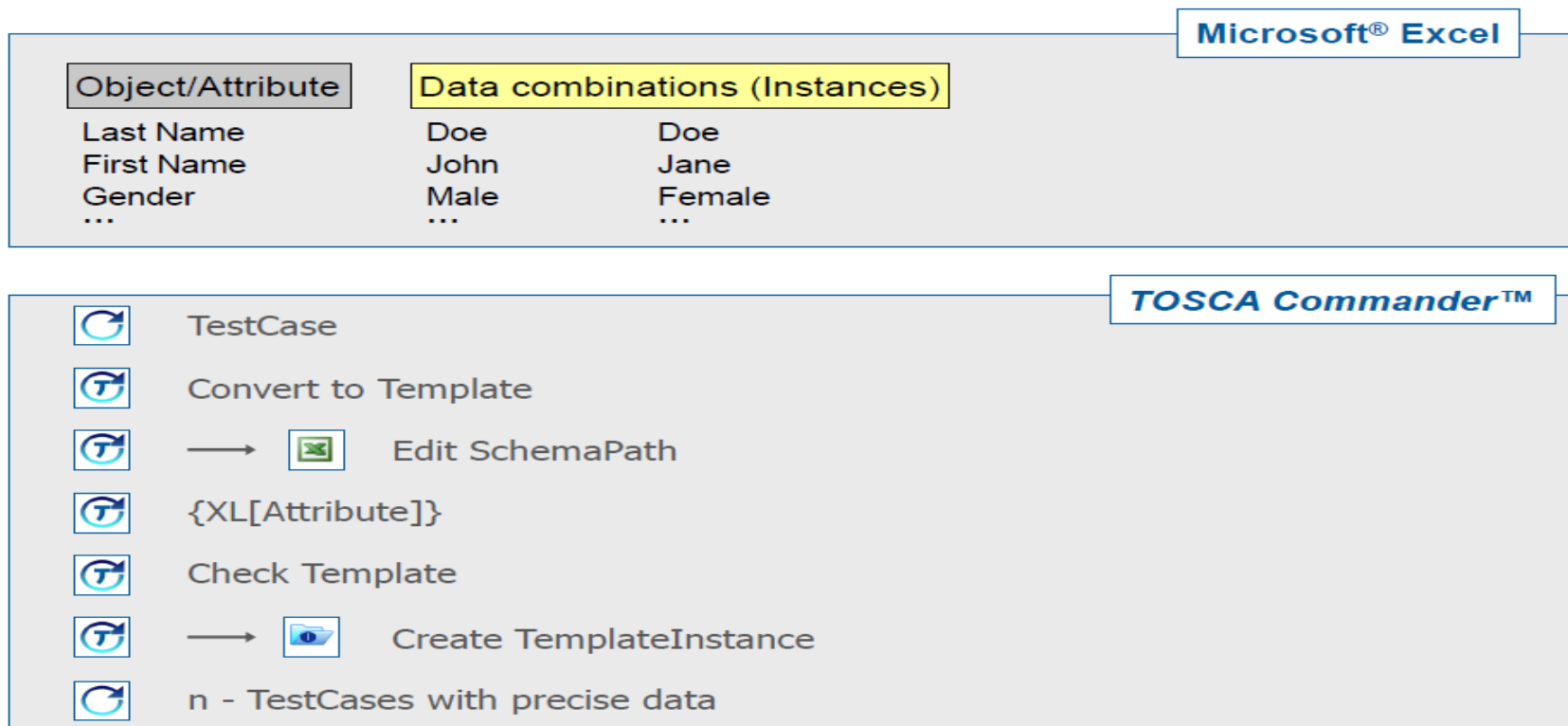
- The TestCase data is collected in **Microsoft® Excel**. The attributes and test data are organized in rows and columns in a **Worksheet**.

	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.



- *Goal*
  - Create Dynamic Test Cases using Microsoft Excel as source of Test Data
- *Task*



- 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*<sup>TM</sup>:
  - {<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.



- 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 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



Exception Handling (all failures are traced)

On Verification Failure ...	<input type="radio"/> Recover	<input type="radio"/> Ignore	<input checked="" type="radio"/> Require manual input
On Dialog Failure ...	<input type="radio"/> Recover	<input type="radio"/> Ignore	<input checked="" type="radio"/> Require manual input
On User Abort ...		<input type="radio"/> Ignore	<input checked="" type="radio"/> Halt execution



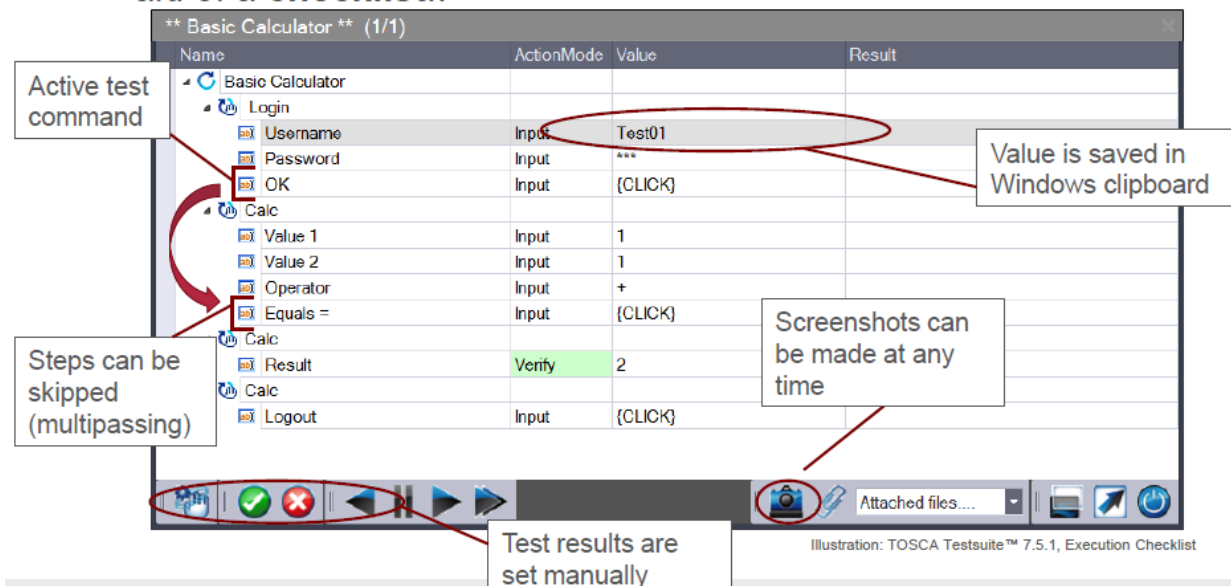
- The response to errors can be specifically set for each project. For every situation, a log will be produced documenting any errors.
- It's possible to define at runtime how each of these error types is handled:
  - **Recover:** Test execution is set to continue without error dialog
  - **Ignore:** 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**





- Should *TOSCA Commander*<sup>™</sup> 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

- *TOSCA Testsuite™* offers the possibility 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**..



**Active test command**

**Steps can be skipped (multipassing)**

**Value is saved in Windows clipboard**

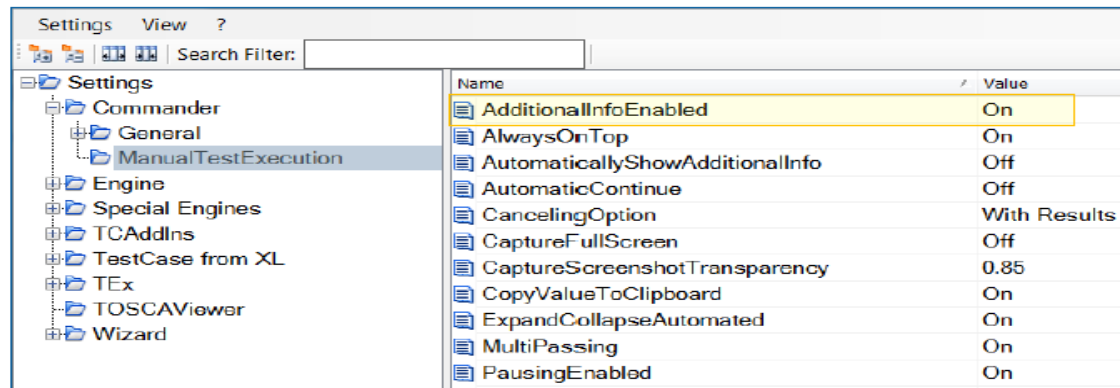
**Screenshots can be made at any time**

**Test results are set manually**

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

- The following settings are recommended based on best practices:



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



- 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 can 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.



Status of objects in the Multuser 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 Multuser workspace.

- *TOSCA Commander*<sup>™</sup> 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*<sup>™</sup>.

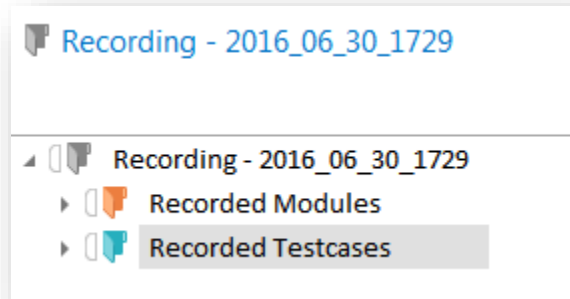
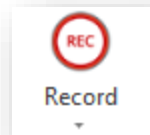


- *TOSCA Testsuite*<sup>™</sup> supports the administration of various test projects with different configuration requirements.
- TOSCA Engine specific settings can be configured by going to **Settings** in the menu bar.
  - Menu: Tools -> Settings -> Engine
  - Menu: Tools -> Settings -> Special Engines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



## 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.

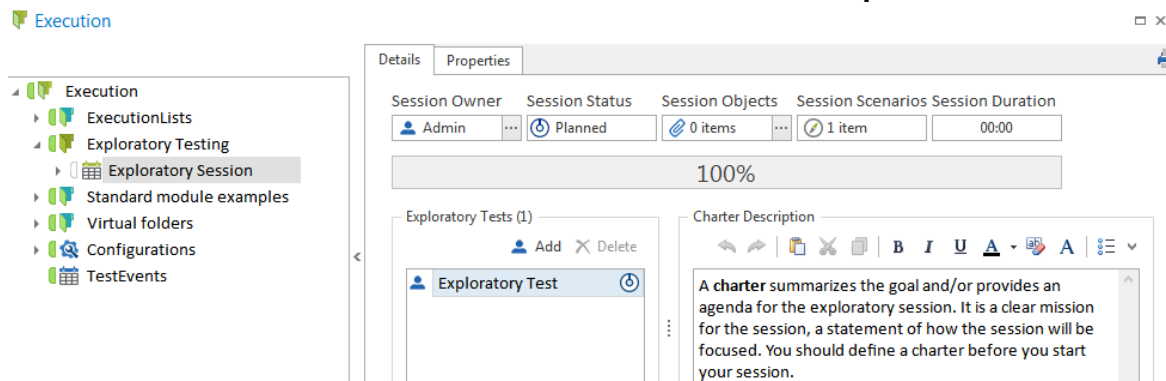


## Exploratory Testing

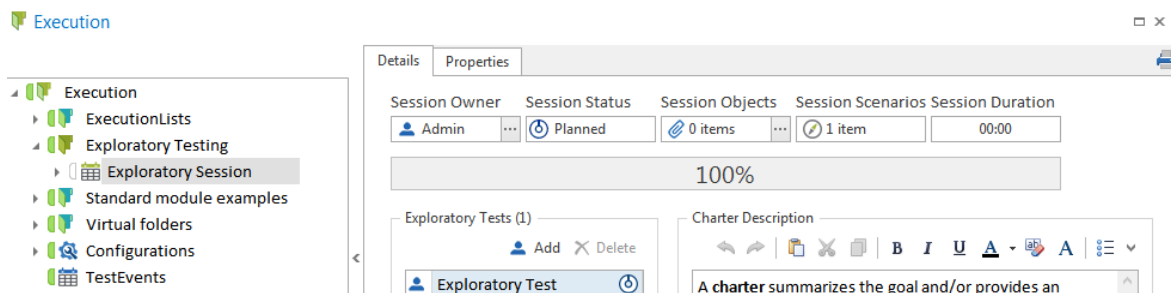
Testers create scenarios, they make use of videos, screenshots and steps (TestSteps) to document both the scenarios and any found errors in the test object.

### 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:** 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.





## 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.

- **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.

Execution

- ExecutionLists
- Exploratory Testing
  - Exploratory Session
    - Exploratory Test
- Standard module examples
- Virtual folders
- Configurations
- TestEvents

Details Properties

Tester

Admin

Test Status

Planned

Test Objects

0 items

Test Scenarios

1 item

Test Duration

00:00

100%

Captured Scenarios

Test Summary

Additional Instructions

Start New Scenario

Continue/Edit Scenario

Create Manual Test Case

Export Scenario

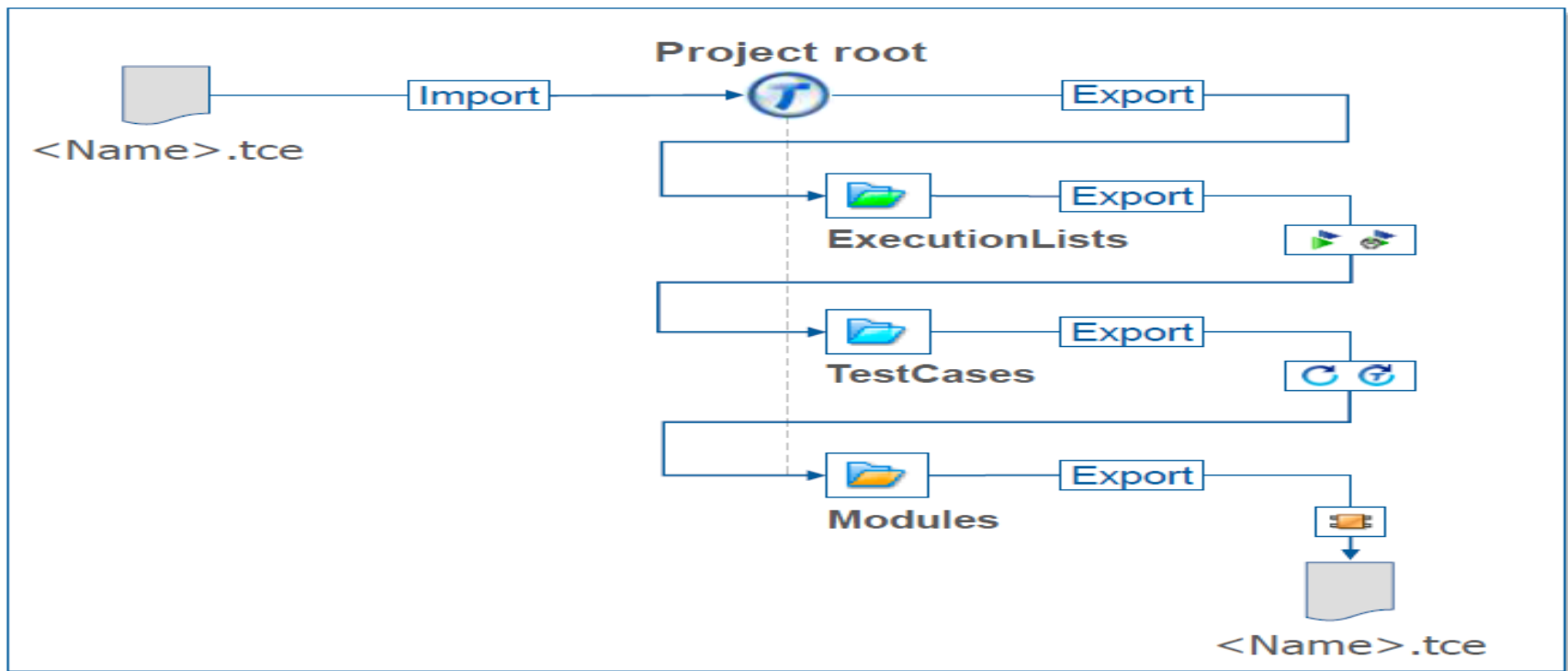
Name	StartTime	EndTime	Duration	Actual Result
Click name	30.06.16 18:38:17	30.06.16 18:39:23	66.37 s	

# 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





**ON THE BASIS OF OBJECT**

Object Type	Shortcut	Action
F	Ctrl + N	Create folder
S	Ctrl + Shift + N	Create directory structure
V	Ctrl + Shift + V	Create virtual folder
A	Ctrl + A	Create Module Attribute
E	Ctrl + E	Create TCD Attribute
C	Ctrl + C	Create class
E	Ctrl + E	Create ExecutionList
G	Ctrl + G	Create user group
I	Ctrl + I	Create group of control elements
I	Ctrl + I	Create control element as part of group of control elements
I	Ctrl + I	Create Template Instance
I	Ctrl + I	Create Instance
L	Ctrl + L	Create Library
L	Ctrl + L	Create TestCase Link
M	Ctrl + M	Create Module
M	Ctrl + M	Create TestStep
M	Ctrl + M	Create manual TestStep Value
M	Ctrl + M	Create TestMandate
O	Ctrl + O	Create ObjectMap
P	Ctrl + P	Create ObjectMap param
R	Ctrl + R	Create new reusable TestStep Block
R	Ctrl + R	Create RequirementSet
R	Ctrl + R	Create Requirement
S	Ctrl + S	Create simple control element
S	Ctrl + S	Create TCD Step
T	Ctrl + T	Create TestCase
T	Ctrl + T	Create TestSheet
U	Ctrl + U	Create user

**Update all**: Ctrl + U

**Checkin all**: Ctrl + Shift + I

**Checkout tree**: Ctrl + Shift + O

**Copy table to clipboard**: Ctrl + Shift + C


**Create window with selected object and connect as tab**: Ctrl + W + Ctrl + N

**F8**: Show/Hide DoNothing

**F9**: Show/Hide Log

**F10**: Show/Hide linked Execution List

**F11**: Show/Hide Instance



Esc F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12

~ 1 2 3 4 5 6 7 8 9 0 Fold all + Expand all Del

Tab Q W E R T Y U I O P { } [ ] Enter

Caps Lock A S D F G H J K L ; ' " \ / ~ `

Shift Z X C V B N M < > ? / Shift

Ctrl Alt Space Alt Gr Ctrl

**TestCase Design**

**Ctrl + E**: Fill in empty values orthogonally

**Ctrl + E**: Fill in empty values specifically

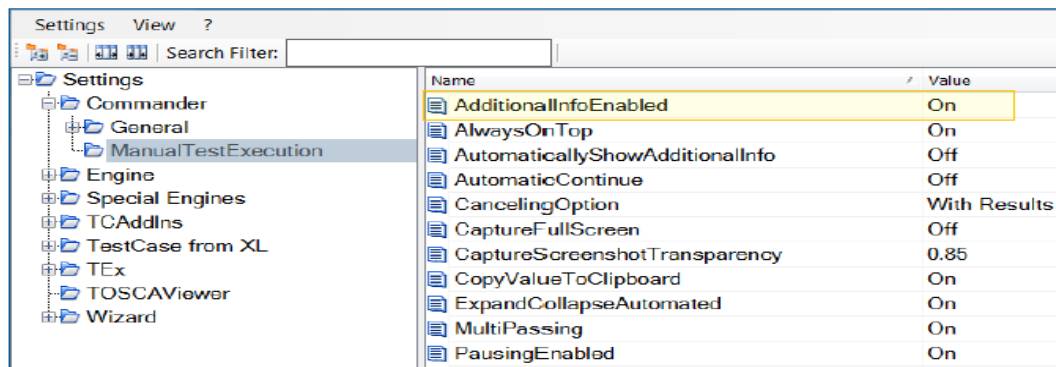
**Ctrl + Alt + E**: extract class

**Ctrl + .**: Create element subsequently

**Ctrl + .**: Duplicate element



- 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


















- 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	
	<a href="#">TestCase folder</a>
	<a href="#">TestCase</a>
	<a href="#">Business TestCase</a>
	<a href="#">TestStep folder</a>
	<a href="#">TestStep, XTestStep</a>
	<a href="#">Manual TestStep</a>
	<a href="#">TestCase Templates, Business TestCase Templates</a>
	<a href="#">TemplateInstance</a>
	<a href="#">TestStep disabled</a>
	<a href="#">TestStep folder disabled</a>
	<a href="#">Manual TestStepValue</a>
	Table
	Input, ControlSimple
	Select, ComboBox
	Button
	RadioButton
	A (Label), HTML Link
	CustomControl

	TreeIcon
	<a href="#">XTestStepValue</a>
	Missing reference
	<a href="#">Property: standard, user-defined</a>
	<a href="#">Folder structure</a>
	<a href="#">Virtual Folder</a>
	<a href="#">TestStep Library</a>
	<a href="#">Reusable TestStepBlock</a>
	<a href="#">Reusable TestStepBlock Reference</a>
	<a href="#">TestCase Planned</a>
	<a href="#">TestCase In Work</a>
	<a href="#">Test Passed</a>
	<a href="#">Test Failed</a>






















Modules	
	<a href="#">Module folder</a>
	<a href="#">Module, XModule</a>
	<a href="#">ModuleAttribute</a>
	Table
	Input, ControlSimple
	Select, ComboBox
	Button
	RadioButton
	A (Label), HTML Link
	CustomControl
	TreeIcon
	<a href="#">ObjectMap</a>
	<a href="#">Import ObjectMap</a>
	<a href="#">Delete ObjectMap</a>
	Open new Module Window




# Icons – Multi User Workspace



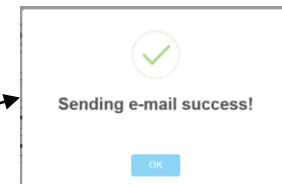
	<a href="#">Element checked out by different user</a>
	<a href="#">New element</a>
	<a href="#">Excluded element</a>
	<a href="#">Excluded folder</a>
	<a href="#">UserGroup</a>
	<a href="#">User</a>
	<a href="#">Checkout tree, Checkout</a>
	<a href="#">Update all</a>
	<a href="#">Checkin all</a>



Execution	
	<a href="#">Execution folder</a>
	<a href="#">ExecutionList folder</a>
	<a href="#">ExecutionList</a>
	<a href="#">Business ExecutionList</a>
	<a href="#">ExecutionMandate</a>
	<a href="#">ExecutionEntry</a>
	<a href="#">Business ExecutionEntry</a>
	<a href="#">ExecutionLog (ActualLog)</a>
	<a href="#">TestCase log (Execution TestCaseLog)</a>
	<a href="#">TestStep log (Execution TestStepLog)</a>
	<a href="#">TestStepValue log (Execution TestStepValueLog)</a>
	<a href="#">ExecutionEntry disabled</a>
	<a href="#">ExecutionEntry folder disabled</a>
	<a href="#">Synchronous ExecutionEntry folder</a>
	<a href="#">AutoMerge</a>
	<a href="#">WriteTestSet</a>
	<a href="#">Import TestResult</a>
	<a href="#">ExecutionEntry - TestCaseWorkState Test planned</a>
	<a href="#">ExecutionEntry - TestCaseWorkState Test In Work</a>
	<a href="#">Test Passed</a>
	<a href="#">Test Failed</a>

	<a href="#">Exploratory Test</a>
	<a href="#">Configuration</a>
	<a href="#">TestEvent</a>

- 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.



Enter Vehicle Data **7** Enter Insurant Data Enter Product

Make

Engine Performance [kW]

Enter Vehicle Data **0** Enter Insurant Data Enter Product

Make

Engine Performance [kW]

Date of Manufacture

Number of Seats

Fuel Type