TOSCA FEATURES

Version - 9.0

1

|  |  |  |
| --- | --- | --- |
| Created by | Reviewed by | Version |
| Ankush Pal | Tricentis Tosca Team | 1.0 |

2

TOPICS

1. Object Identification - Using anchor, Image and Index

2. Control Properties

3. Loops and Conditions

4. Table Steering

5. Action Mode CONSTRAINT

6. Recovery and Cleanup scenario

7. TOSCA Functions

8. Reusable Test Step Block (RTB)

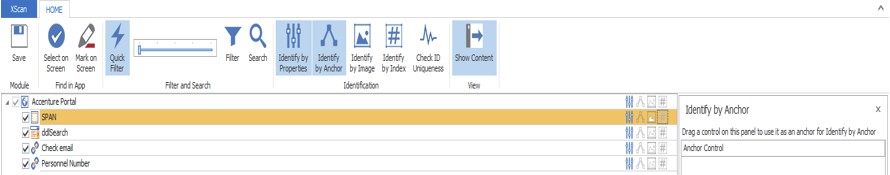
9. Input Parameters

10.Partial Buffer

11.Keywords

12.TOSCA Search (Simple & TQL)

3



Object Identification using advanced methods

1. IDENTIFY BY ANCHOR

INTRODUCTION:

In Tosca, we are able to use technical properties of controls in order to identify other controls, for instance if we have any control which cannot be uniquely identified using technical and representative properties, then it can be uniquely identified using ANCHOR.

For Example, “SP N”, as shown below, is not uniquely identifiable͘ Now with the help   
of “ddsearch” (which is uniquely identifiable), we use the option “Identify by nchor” to make SPAN   
unique.

Fig.1 Before Anchoring.

Steps to be followed to use “Identify by nchor”:

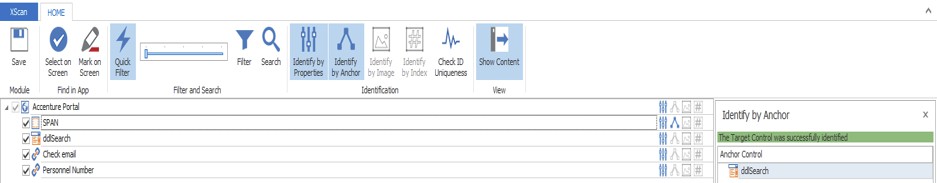
 Scan the required screen by using Tosca XScan.

 Open the “Identify by nchor” window by clicking on the corresponding button in the HOME   
 menu.

 Select the control which should be identified (“SP N”)͘

 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   
 (“ddsearch”)͘

4



 The Anchor Control Properties field shows the properties that were transferred along with the   
 control.

Fig.2 After Anchoring.

The above figure shows that “SP N” is now uniquely identifiable͘ The RelativeID of the control “SP N” shows that it is anchored with “ddsearch”͘

Note: We should try for all the different combinations of Technical and Representative Properties first to make a control unique, if not possible, then go for “Identify by nchor”͘

2. IDENTIFY BY IMAGE

INTRODUCTION:

We can use “Identify by Image” option to make any control unique in case, it is not possible to make it uniquely identifiable using technical and representative properties.

Screen objects can be found and steered by means of images. This requires a control to be created by capturing an image which is used for identification later on. This image is then searched and an   
according action is performed upon control steering.

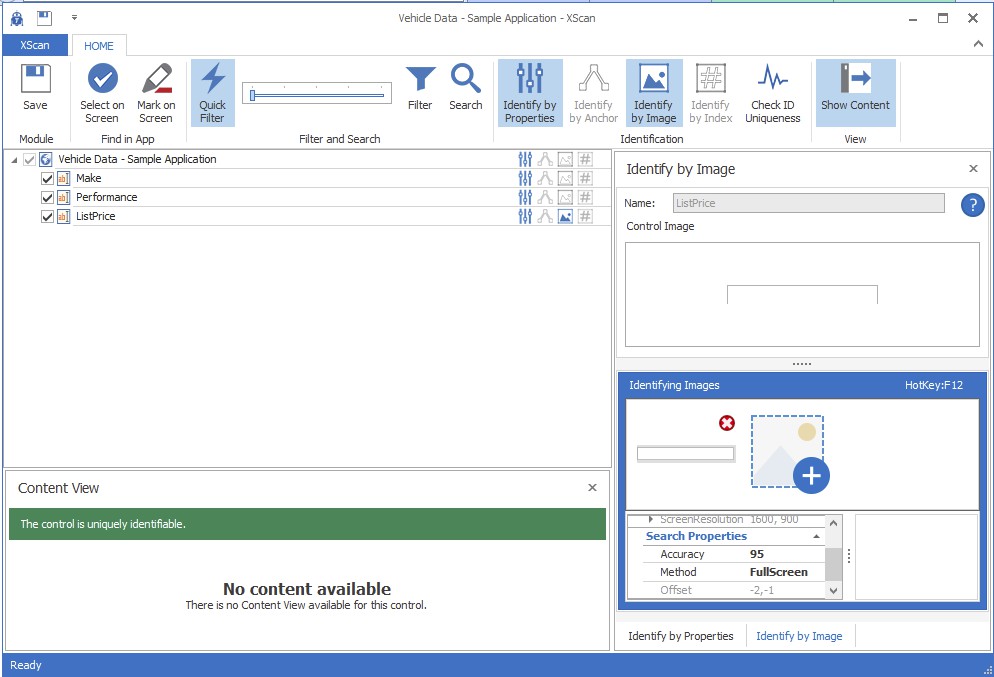
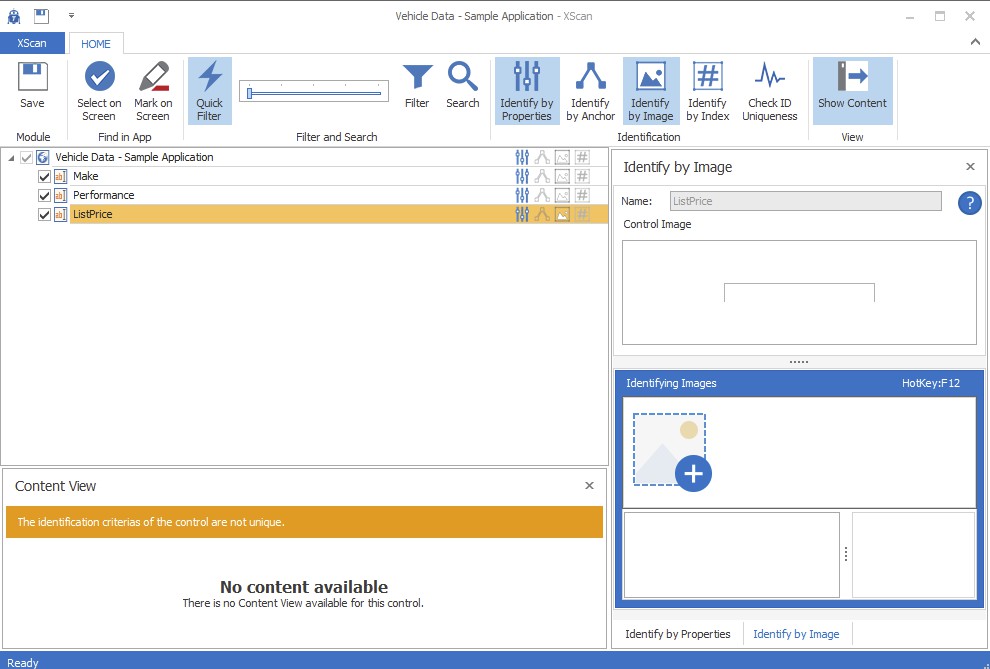
STEPS TO MAKE A CONTROL UNIQUE:

1. Scan the required screen by using Tosca XScan.

2. Open the “Identify by Image” window by clicking on the corresponding button in the HOME menu.

3. Select the control to be identified by means of a screenshot in the left section of the window (ListPrice).

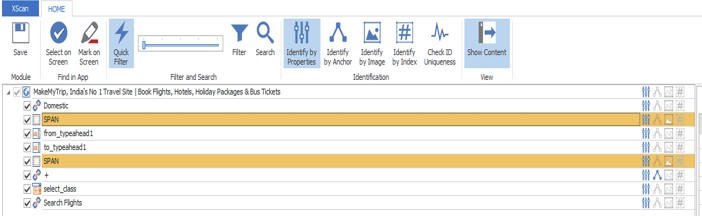
5



The below given figures explains how to make a control unique using “Identify by Image”͘

Fig. 1 Before using “Identify by Image”

6



Fig͘2 fter using “Identify by Image”, control (ListPrice) is now unique͘

Note:

This option is least preferred as changed screen resolution during execution might affect the identification of object.

3. IDENTIFY BY INDEX

INTRODUCTION:

The new XScan index based control identification helps tester to identify their control using a unique index defined by Tosca.

When c

Fig.1 Before using “Identify by Index”͘

Steps to be followed to use “Identify by Index”:

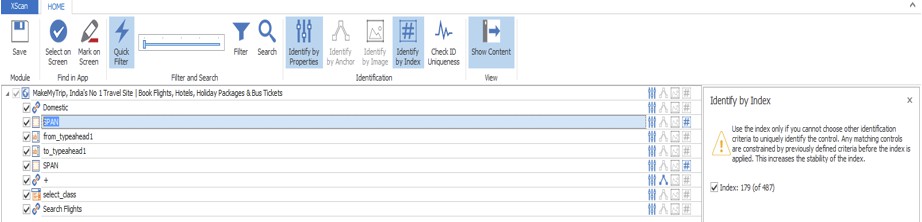
 Scan the required screen by using Tosca XScan.

 Open the “Identify by Index” window by clicking on the corresponding button in the HOME   
 menu.

 Select the control which should be uniquely identified (“SP N”)͘

 Mark the Index checkbox and Tosca will assign an index value to the respective control which   
 makes it uniquely identifiable.

7



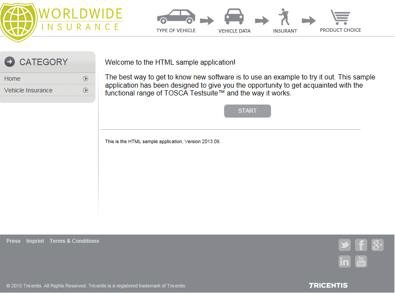
Fig͘2 fter using “Identify by Index”͘

The above figure shows that “SP N” is now uniquely identifiable͘ new option “Constraint index” is added to the properties of the control.

Any matching control are constrained by below defined criteria before the index is applied. This increases the stability of the index

Note: We should try for all the different combinations of Technical and Representative Properties first, to make a control unique, if not possible, then go for “Identify by Index”͘

8



Control properties

.Enabled:

 It is property which specifies if the object is active or not.  Enable property has the datatype Boolean.

 In the following example START is a button on which click action can be performed. In order to   
 perform that action START has to be active.

Steps to be followed:

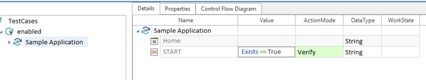
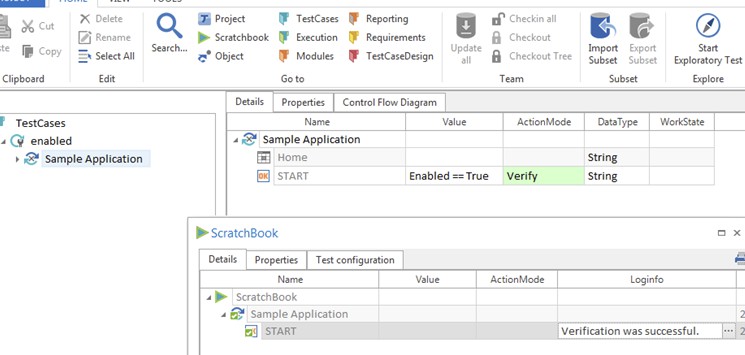
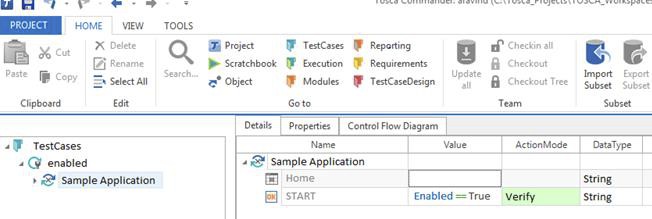
 Scan the object that has to be verified and drop it in the test case.

 The value of the selected object is given as .Enabled (True/False) can be verified using Action   
 Mode Verify.

 You can buffer the value of the property using “.Enabled -> <Buffer\_name>” and action mode as   
 Buffer.

 The control properties can also be directly selected by clicking in the value field of any object.  Run the test case to perform desired action.

9



.Exists:

 It specifies that the object exists on the page.

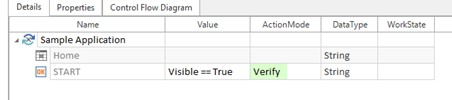
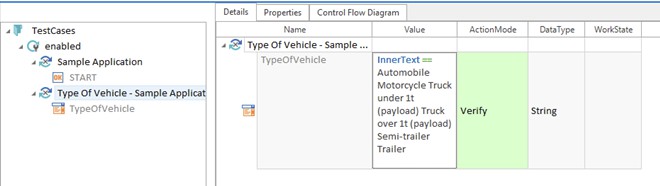
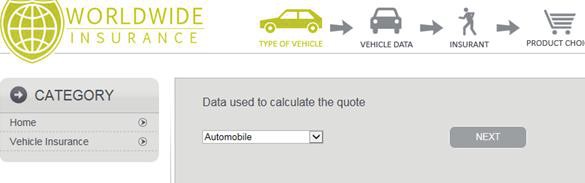
 Checking for exists property on the same start button in the previous example.

.Innertext :

 It checks the Innertext property of the object.

 In the example below, type of vehicle is an object with the inner text as automobile, motorcycle   
 etc.

10



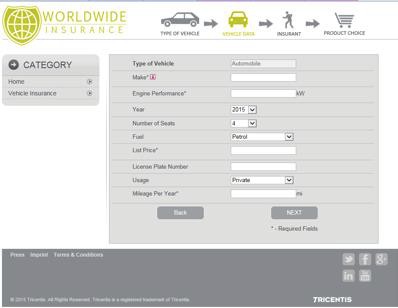
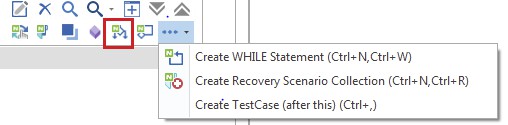
.Visible:

 It is a control property which specifies that the object is visible on the scanned page.

 There are some scenarios where the objects exist on the page but are not visible.

 In such cases this property helps us to know if the object is visible or not.

11



Loops and Conditions

IF ELSE:

 It is a feature to dynamically control the flow within individual test cases.  IF block has a condition followed by THEN block and ELSE block.

Steps to create:

 Right click test case and select Create if statement.

 On creating the IF statement the test step which has to be checked for the condition is added   
 into condition (only one test step can be added in condition).

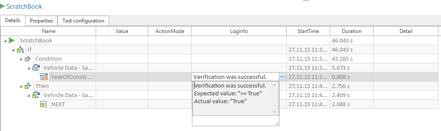
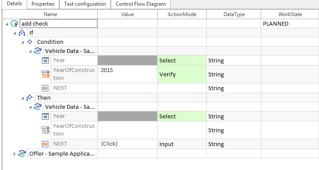
 Test steps which are to be run when the condition is true are dragged under THEN. If the   
 condition is false Else case is run.

 Multiple IF statements can be added inside THEN and ELSE block

Example:

 In the following example if year of construction is 2015 click next.

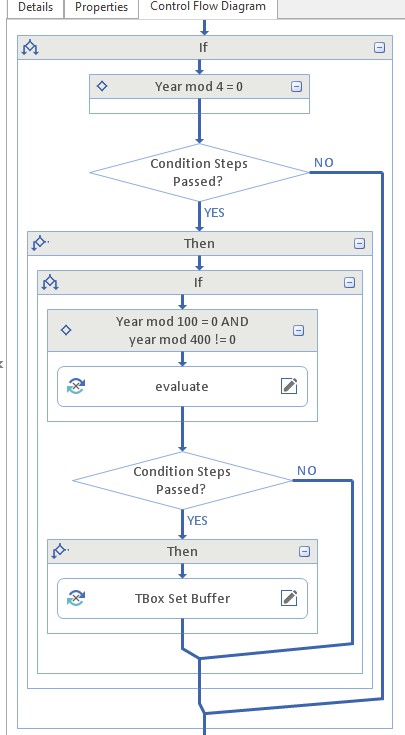
12



 Verify if the value in field “year of construction” is 2015͘

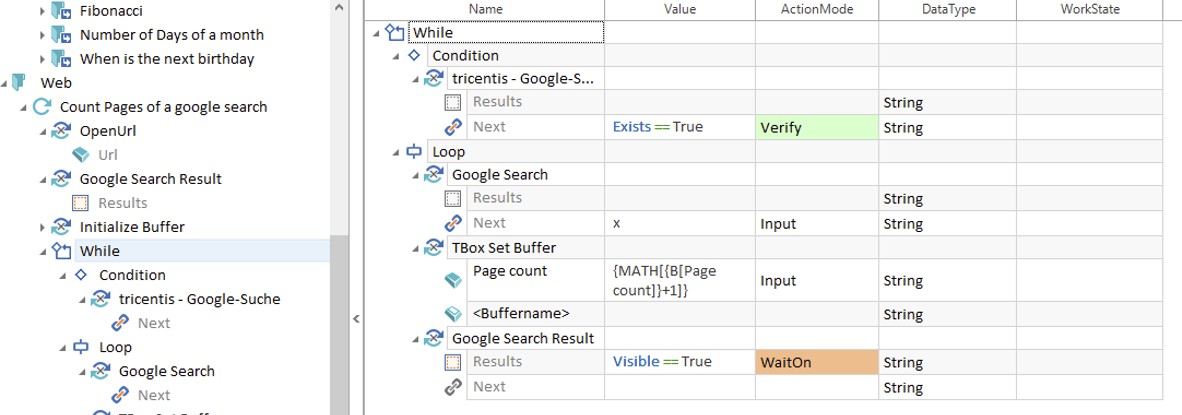
 Here object “next” is present in THEN block which is clicked if the condition is true͘  ELSE block is executed if condition is false.

13



Control flow diagram:

14



WHILE Loop:

A WHILE loop is repeatedly executed as long as a given condition is true. To use while loop below steps are followed:

 Right Click on test case and create a while loop.

 Add a teststep into the condition which has to be tested.

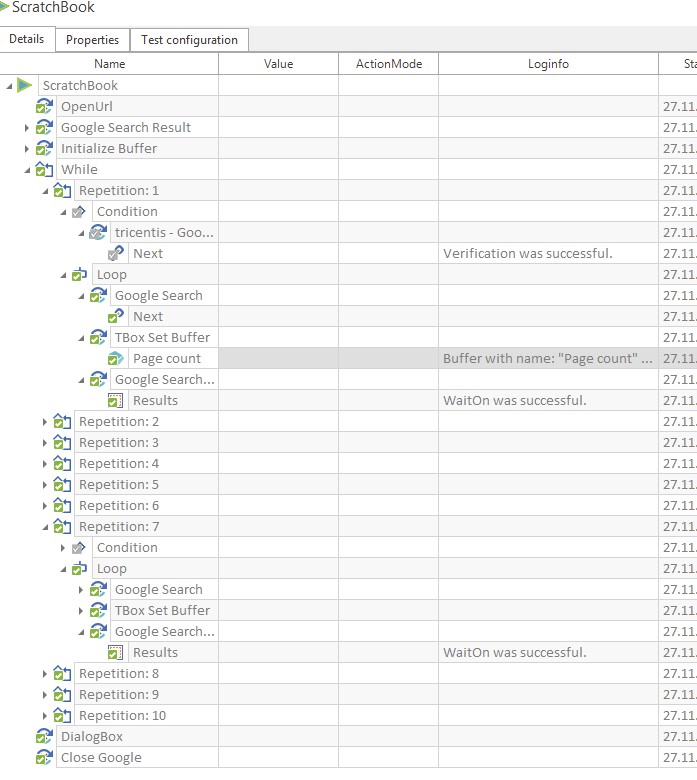
 Add test steps into the loop which are to be iterated until the condition fails.  You can add WHILE loop inside WHILE. (condition and loop)

Example:

 The loop is executed till “NEXT” button exists͘

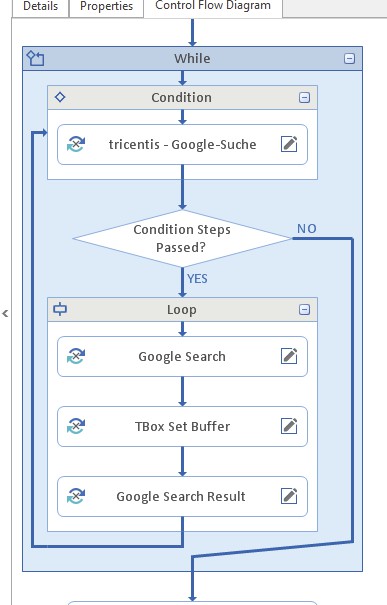
 If condition is true then next button is clicked else display a dialog box with the number of pages   
 searche

15



Output :

16



Control Flow Diagram:

DO WHILE Loop:

DO- WHILE loop is same as while loop. Only difference is it executes loop at least once despite condition is true or false.

It can be created in same way from Context menu.

17

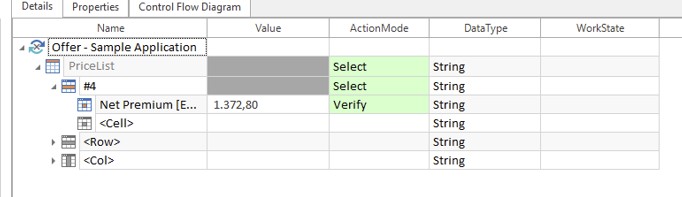
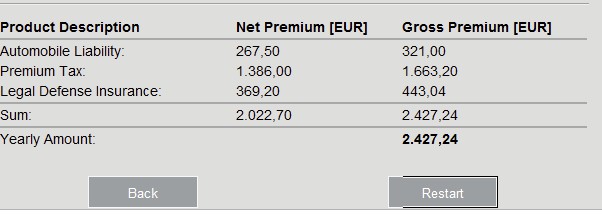


Table Steering

Consider the table below:

\

Steps to perform table steering:

1) In test cases section give the row number as #<row number> or the row name in name field.

2) Give the column name or column number in the name field below the row.

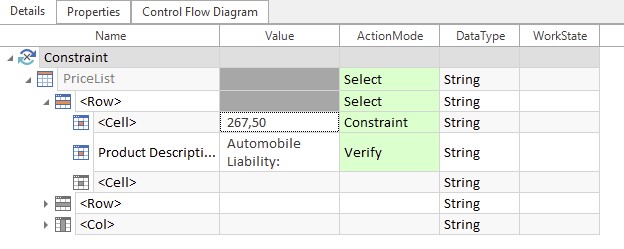
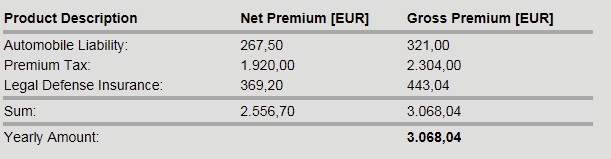
3) Same way you can start using column and then specify row under column.

4) Give the value and action mode according to the action to be performed.

i.e., specify value=<buffer name> and action mode =buffer if data from table is to be stored.

For example -

18



ACTION MODE CONSTRAINT

This Action Mode is used to find (Verify) an element in the tables. Consider the table below:

To find an element using constraint action mode, below steps are followed:

5) Scan the Table from the application and create the module.

6) Create the test case with the table.

7) In test cases section add the known element (reference), value (Value) with action mode as   
 Constraint in the table.

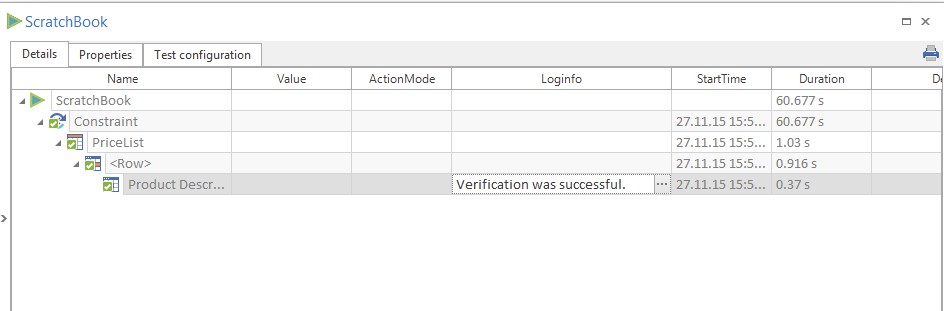
8) The column and row of the reference is identified.

9) You can find the elements in the same row, where the value in the constraint (e.g.- 267,50) was   
 present by specifying column name (Product Description). You can perform different actions -  
 Input/Verify/Buffer.

Please refer the screen below for reference:

On running the test case in scratch book, we can see the success message here. Following this the same test case can be run in ExecutionList.

19

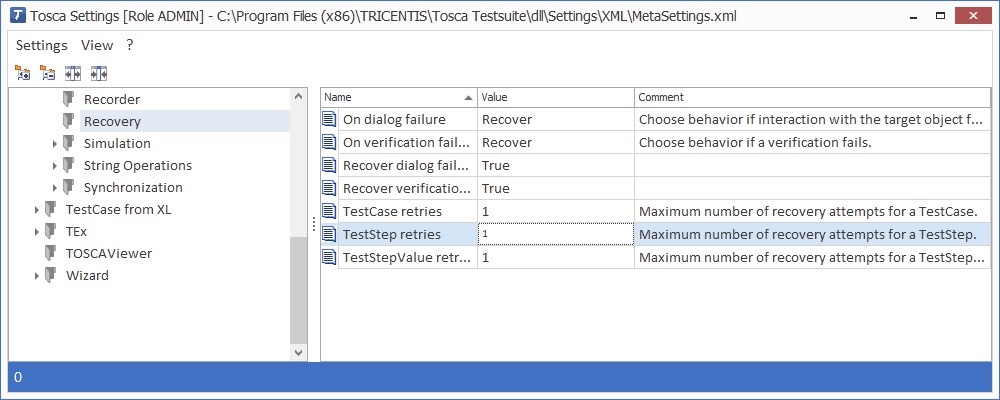
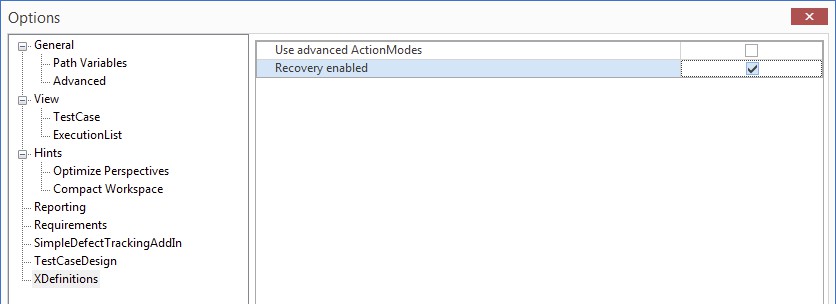


Note:

 Constraint action mode is used for identifying elements in tables only.

 This action mode is helpful, in case we don’t know the row and cloumn of reference element. In   
 this case superordinate element is found with reference to row and column of subordinate   
 element.

20



RECOVERY SCENARIO

Recovery scenarios are written in case there is a teststep fails and the user wants a recovery from the failure instead of retrying the case manually or halting the execution.

Recovery scenario can be created at test case or folder level.

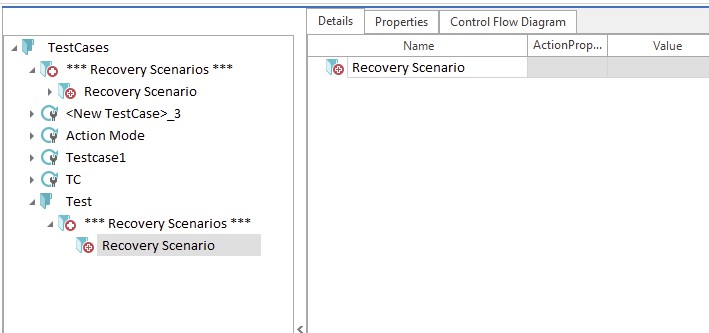
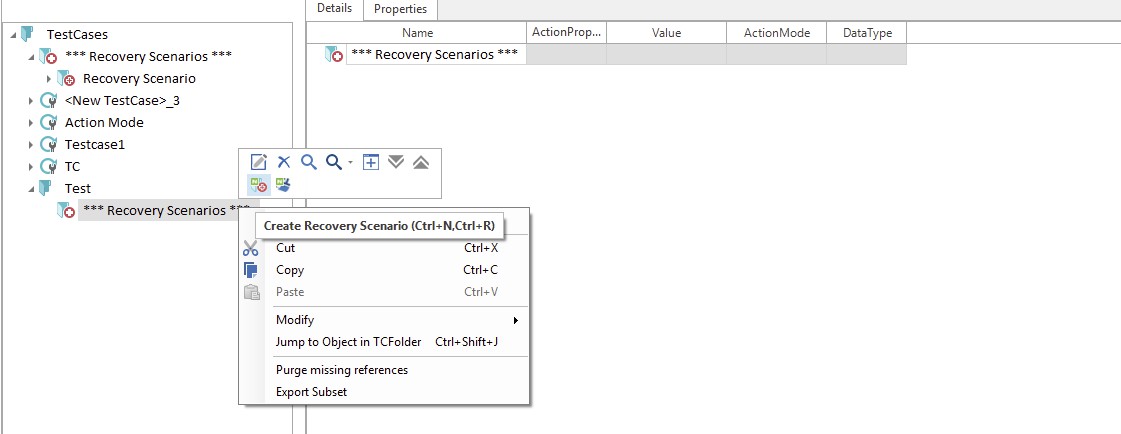
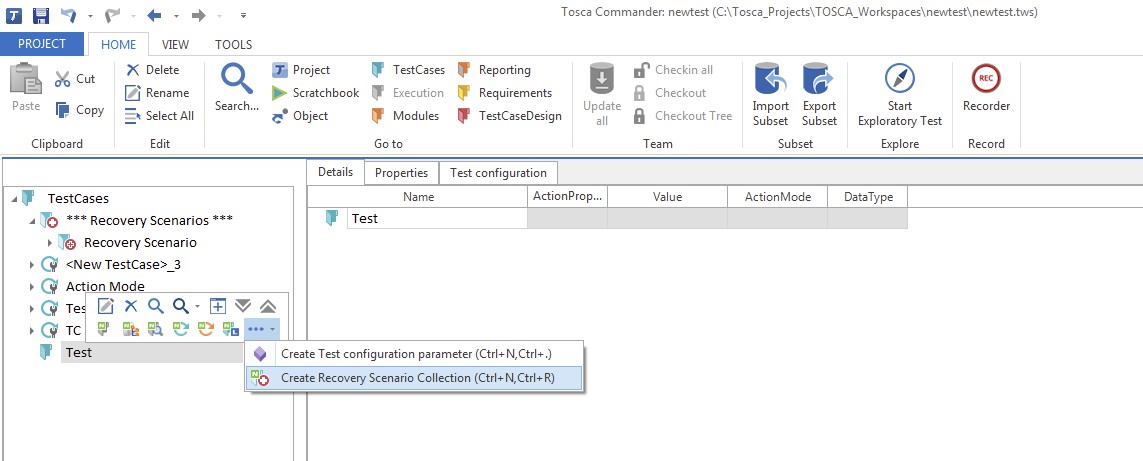
To call a recovery scenario, the Recovery option must be enabled in the Options dialog via the setting XDefinitions->Recovery.

 The TBox Recovery is configured via Settings dialog via the setting TBox->Recovery. It is also   
 possible to specify test configuration parameters for one or more objects.

Steps to create Recovery scenario:

1) The first step is to create a Recovery Scenario Collection in the TestCases section by using the   
 context menu entry Create Recovery Scenario Collection.

21



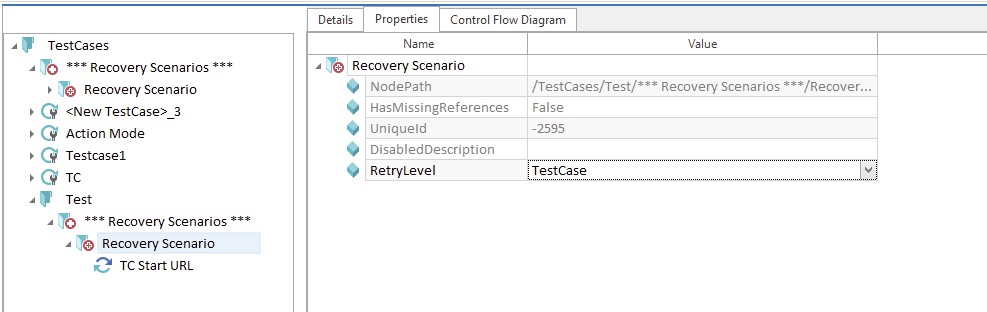
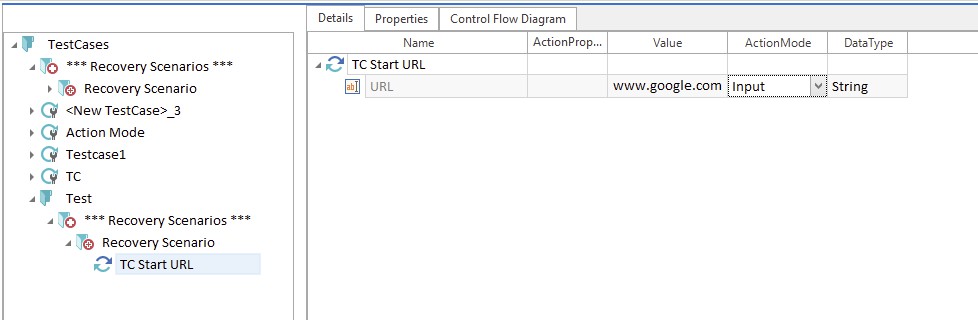
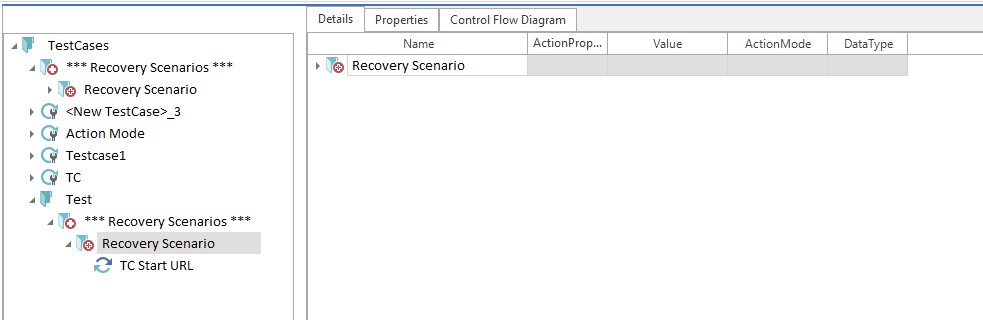
2) A Recovery Scenario can be created by selecting Create Recovery Scenario from the context   
 menu of a Recovery Collection.

 If several Recovery Scenarios exist in a Recovery Scenario Collection, they are processed one   
 after another in case of a recovery, until an applicable scenario is found.

 A Recovery Scenario is considered to be successful if all TestSteps are executed with a positive   
 result.

 The scenarios are processed from top to bottom.

22



3) The recovery test step is then created under recovery scenario.

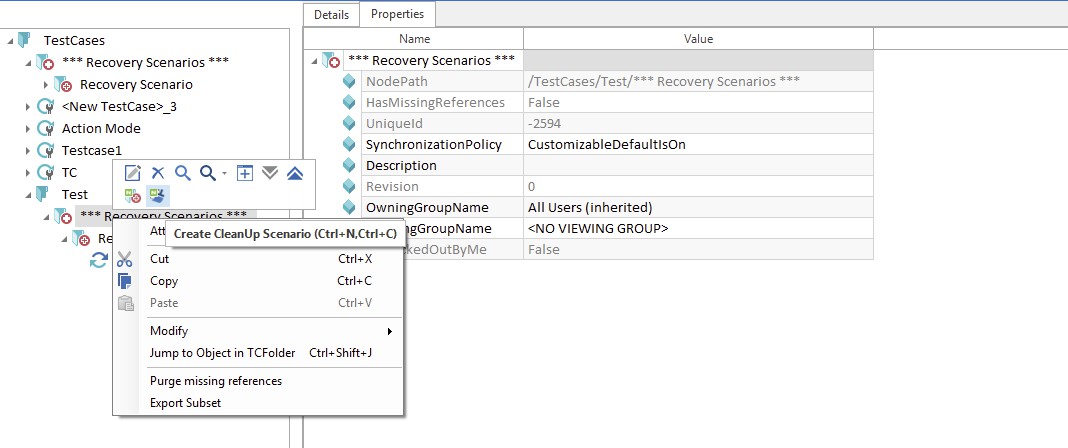
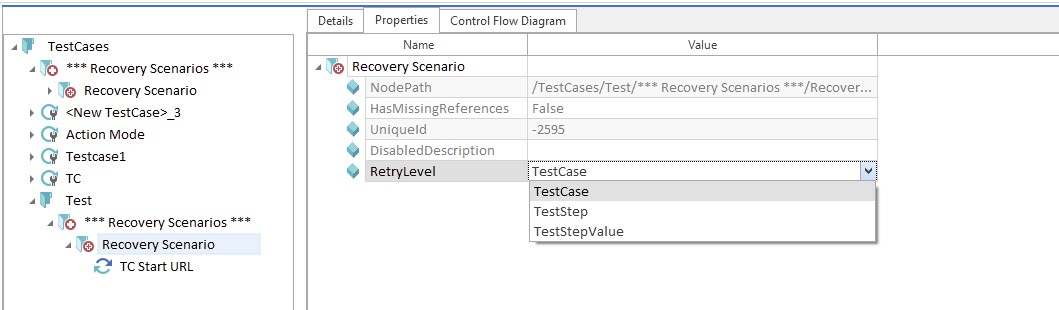
4) Retry Level must be set at recovery scenario level. We have below retry levels:   
  TestCase

 TestStep

 TestStepValue

 The RetryLevel can be set for each Recovery Scenario. The default value is TestCase.

23



5) The test case to be executed is dragged to Execution section to execute.

Example for recovery scenario:

In case while submitting a transaction there is some validation, and a dialog box appears where we need to click OK, we define a recovery scenario to click on OK, in case teststep fails.

Note: Recovery scenario can be executed only in execution lists.

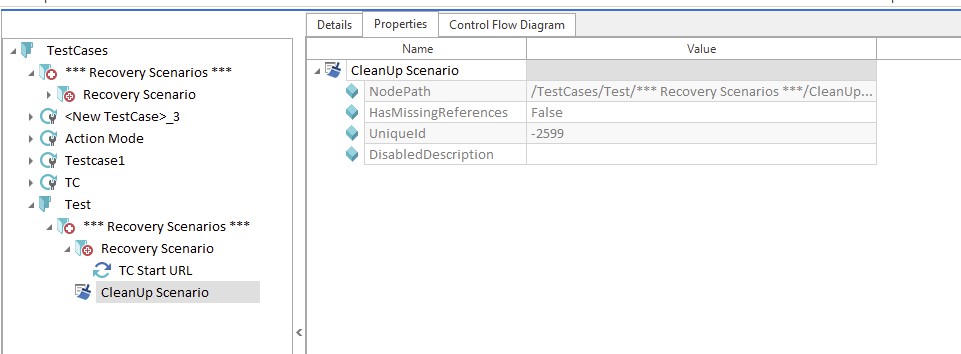
Clean Up Scenario:

In addition to recovery scenarios you can define CleanUp scenarios which adjust your test environment if the recovery has failed.

Steps to create Cleanup Scenario:

1) Use the option Create CleanUp Scenario from the context menu of a Recovery Scenario   
 Collection to create one or more CleanUp scenarios in addition to recovery scenarios.

24



2) Teststep is then created under cleanup scenario.

NOTE:

 CleanUp scenarios are triggered if none of the defined recovery scenarios returns a positive   
 result.

 If a Recovery Scenario Collection includes more than one CleanUp scenarios, these will be   
 processed one by one until an applicable scenario is found.

 If any problems should occur within the CleanUp scenario, no recovery will be triggered and the   
 result of the executed TestCase won't be modified.

Example for cleanUp scenario:

Kill the browser process.

25

TOSCA FUNCTIONS

DATE AND TIME

Date is a dynamic expression. In TOSCA dynamic expressions are used if values are needed for test specification that are generated only when the TestCases have been executed. Dynamic expressions can be used on both the Module and the TestCase level.

The expression can be specified with and without parameters.

Syntax:

{<EXPRESSION>[<Basedate>][<Offset>][<Format>]}   
  Possible <EXPRESSION> values are:

 DATE: current date(dd.MM.yyyy(default tosca date format))  TIME: current time(format: HH:mm)

 DATETIME: current timestamp(format:yyyyMMddHHmmss)  MONTHLAST: last day of the current month

 MONTHFIRST: first day of the current month

If the expression is used with parameters, all three parameters below must be specified:

 <Basedate> : A random date can be entered as the reference date which should be used   
 for the calculation.

 <Offset> : Deviation from the <Basedate>.(additions or manipulation needed should be   
 given here Eg :[+16d+2M+1y]). These operations can be performance on the {TIME}   
 expression.

 <Format>: The date or time format in which we want to view the result or output. Few descriptions:

 d- Day

 M-month   
 y-year

 HH-hours(h: in manipulations)  mm-Minutes

 ss-Seconds

26

FORMATS:

(TOD Y’S D TE: 25͘11͘2015)

|  |  |  |
| --- | --- | --- |
| FORMAT DESCRIPTION RESULT | | |
| dd | The day of the month i.e  thorough 1 to 31st | 25 |
| ddd | The first three letters of name  of the day of the week | Wed |
| dddd | The full name of the day of the  week | Wednesday |
| MM | The month through 1 to 12 | 11 |
| MMM | The abbreviated name of the  month | Dec |
| MMMM | The full name of the month | December |
| yy | The last two digits of the year | 15 |
| yyyy | The year as 4-digit number | 2015 |
| HH | The hour,using a 24 hour clock  from 00 t 23 | 16 |
| Mm | The minute, from 00 to 59 | 30 |
| Ss | The seconds from 00 to 59 | 23 |
| Tt | The AM/PM designator | PM |
| g(Datetime format) | MM/dd/yyyy  (12hrs format)HH:mm | 11/25/2015 4:51 PM |
| s(Datetime format) | yyyy-MM-ddTHH:mm:ss | 2015-11-25T16:46:26 |

NOTE: After giving the expression in TOSCA we can see the value of the expression without

executing the testcase. It can be done by right clicking on the test step and selecting translate   
value. It translates the expression and shows the value in the specified format in a dialog box.

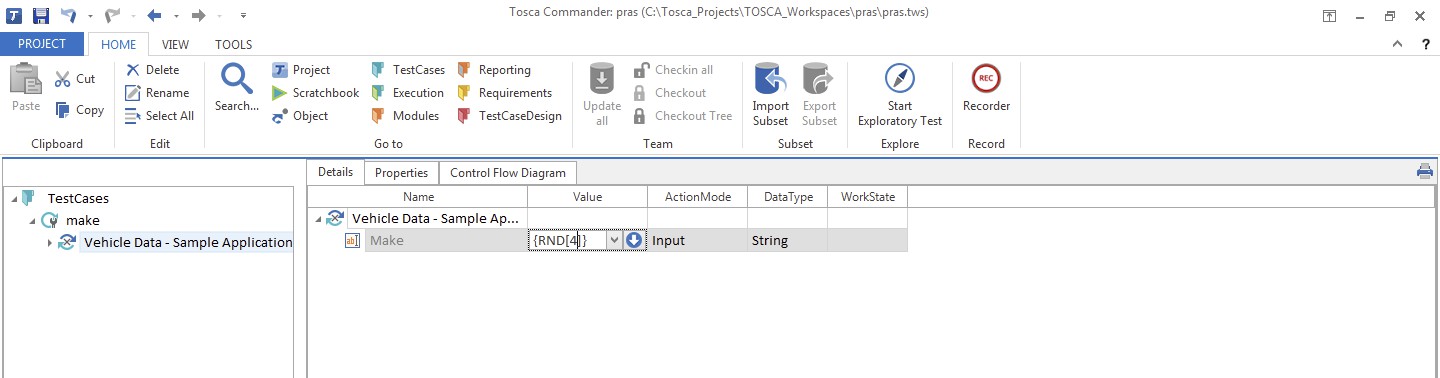
27

FEW EXAMPLES:

TOD Y’S D TE:25͘11͘2015

|  |  |
| --- | --- |
| EXPRESSIONS RESULT | |
| {DATE} | 25.11.2015 |
| {DATE[][+16d][]} | 11.12.2015 |
| {DATE[][+1y-1M+2d][]} | 27.10.2016 |
| {MONTHFIRST} | 01.11.2015 |
| {MONTHLAST} | 30.11.2015 |
| {MONTHFIRST[][+2M][]} | 01.01.2016 |
| {MONTHFIRST[][+2y+2M][]} | 01.01.2018 |
| {DATE[11.08.2006][+2M+9D][]} | 20.10.2006 |
| {DATE[][][y]} | November, 2015 |
| {DATE[][][M]} | November 25 |
| {DATE[][][d]} | 11/25/2015 |
| {DATE[][][ddd]} | Wed |
| {DATETIME} | 20151125144613 |
| {TIME} | 14:44 |
| {TIME[][+2h][HH]} | 16 |
| {TIME[][+2h+2m][HH:mm:ss tt]} | 18:43:27 PM |
| {DATETIME[][][s]} | 2015-11-25T16:46:26 |
| {DATETIME[][][g]} | 11/25/2015 4:51 PM |

28



Random Number:

This function is used to generate any random number   
CASE 1:

{RND[no of digits]}

Input value: {RND[4]}, this is to assign any random number . 4 represents number of digits of a random   
number.

Output is 4 digit random value: 5336

CASE 2:

Generate a random number between two numbers. SYNTAX: {RND[lowerlimit][upperlimit]}

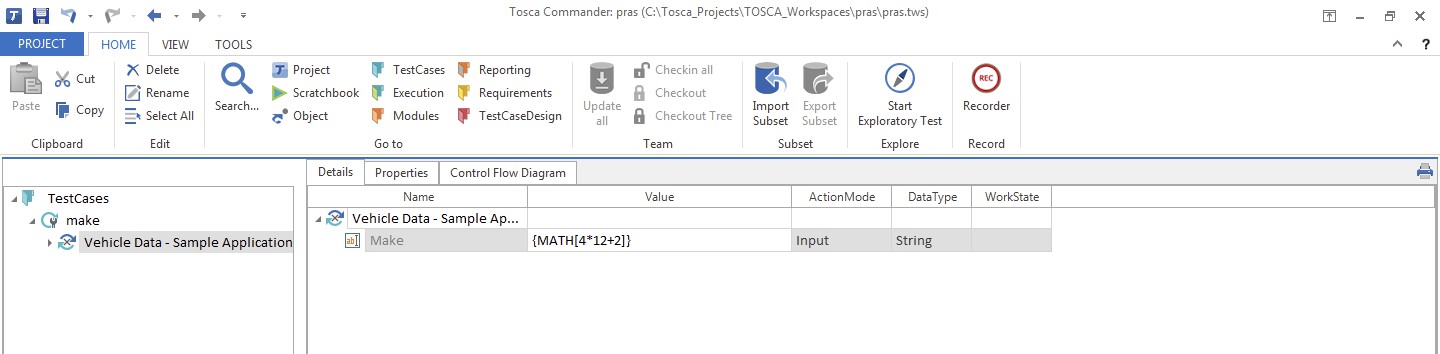
Example :

Input value: {RND[10][230]

Output is a random value between 10 and 230 : 147

Random Decimal:

29



Any random decimal value can be given using a syntax given below

SYNTAX: {RNDDECIMAL[no. of decimal values][lowerlimit][upperlimit]} Input value: {RNDDECIMAL[3][128][567]}

3 represents required no. of decimal values, 128 and 567 are the limits of a random decimal number Output is 358.703

MATH:

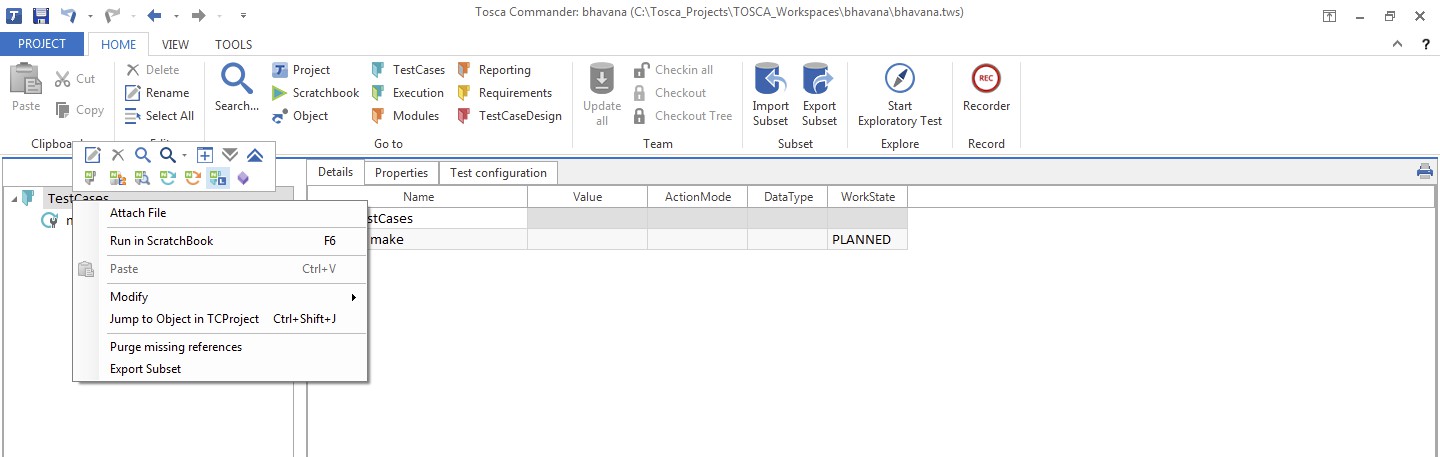
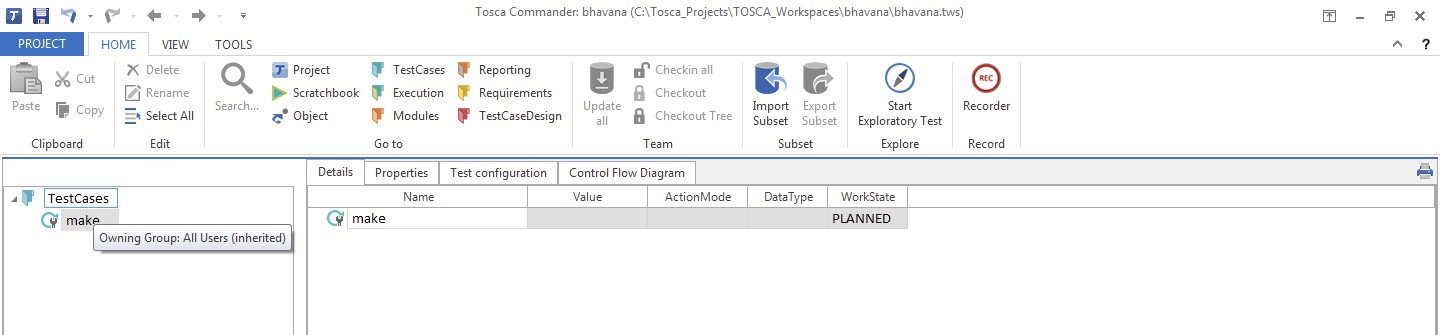
MATH function is used to calculate required arithmetic calculations using a syntax. SYNTAX: {MATH[arithmetic calculations]}

Input value: {MATH[4\*12+2]} .

Output is 50

You can use buffered value inside math function as below -  
e.g. -{MATH[{B[next]}+5]}

30



Reusable TestStep Block

Reusable TestStepBlocks (RTB) and their references contain Test Steps which can be managed centrally. They are created and managed within TestStepLibraries.

Reusable TestStepBlocks must be created to be able to use TestStepLibraries, which are then filled with   
TestSteps.

1. Create a new test case.

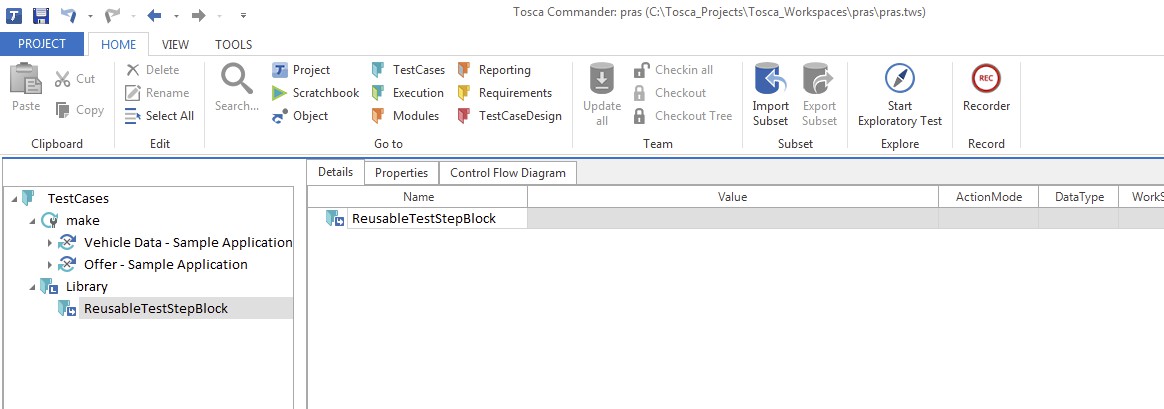
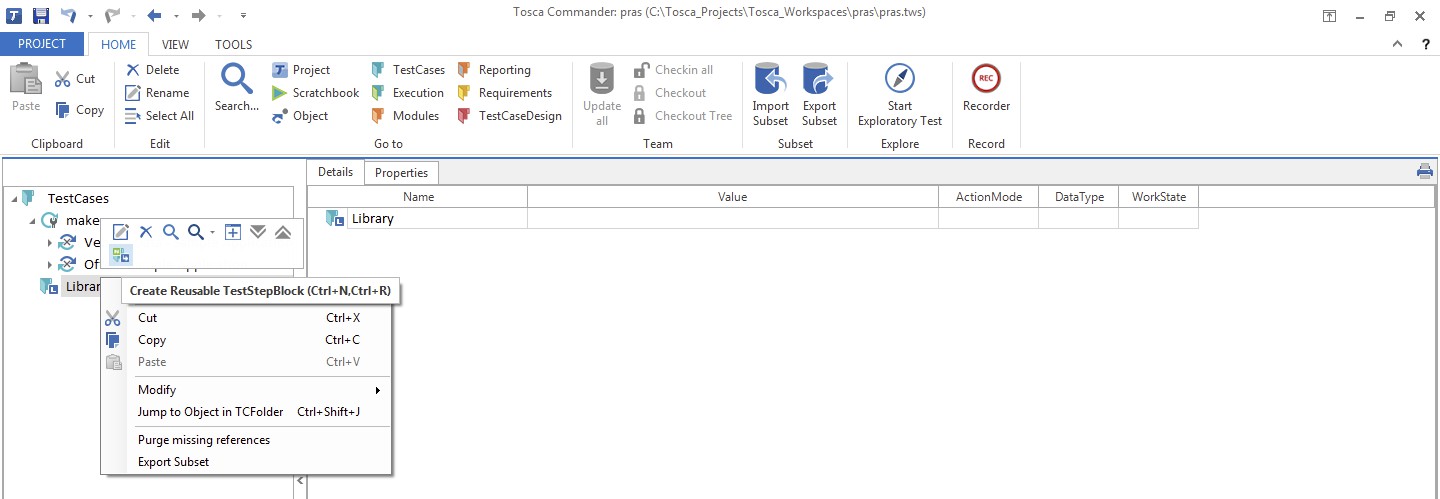
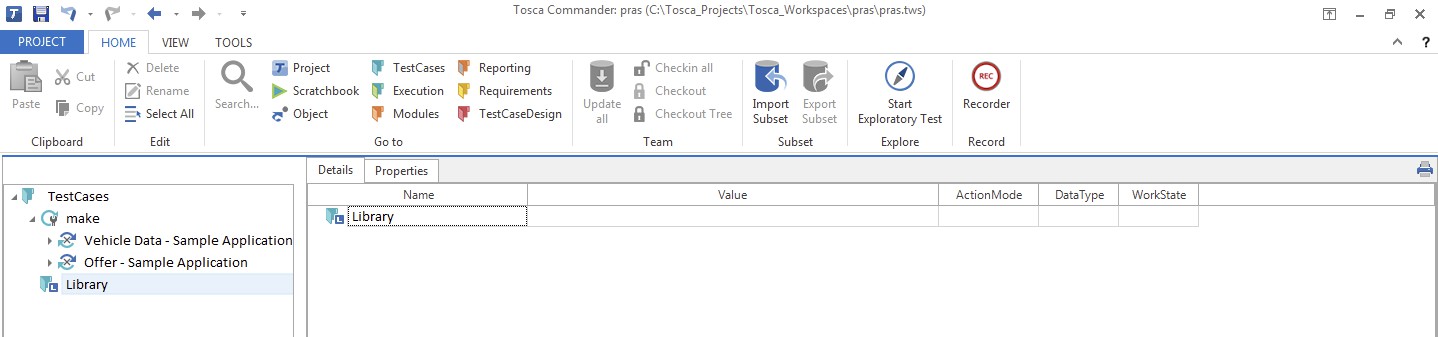
2. Right Click on the Testcase folder , Create a new testStepLibrary.

3. Right Click on the TestStepLibrary and create new ReusableTestStepBlock.

4. Select Reusable TestStepBlock and Reusuable TestStepBlock is created.

5. Now Drag Test Step to the Reusable TestStepBlock.

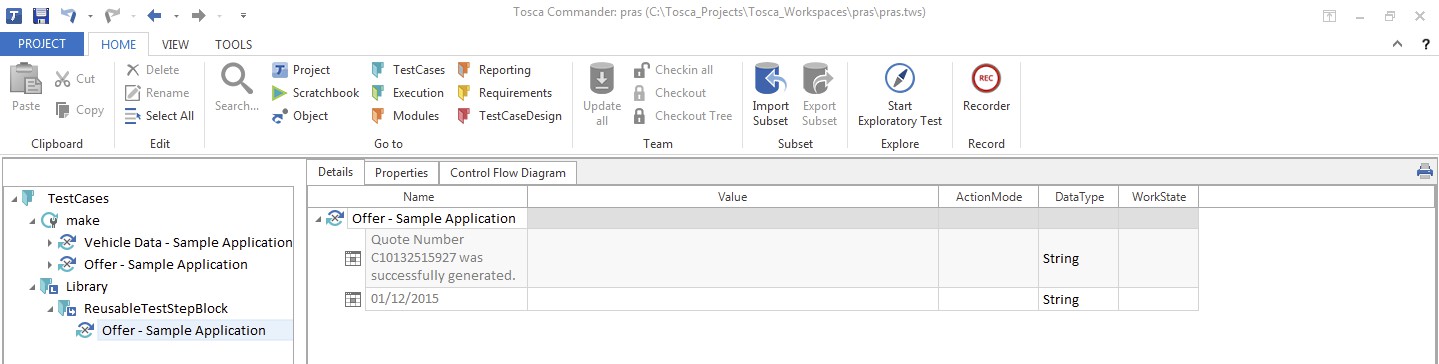
31



Select Reusable TestStepBlock and Reusuable TestStepBlock is created.

Drag the required Test Steps to the Reusable TestStepBlock.

32



This Reusable TestStepBlock can be used directly whenever the TestStep is required there is no need to Create New TestStep.

If the test Step need any change the change can be directly made into the TestStepBlock and same will be reflected into all the Test cases where the TestStepBlock is used.

Resolve Reference

You can use resolve reference when you do not want to refer to the RTB but want to manage it individually inside the test case.

You can do this by selecting option “Resolve reference” in context menu of RTB in a test case͘

Working with Reusable TestStepBlock:

With drag & drop, existing Reusable TestStepBlocks can be added to TestCases. As a result, a reference to the Reusable TestStepBlock is created in the TestCase.

Merging Reusable TestStepBlock:

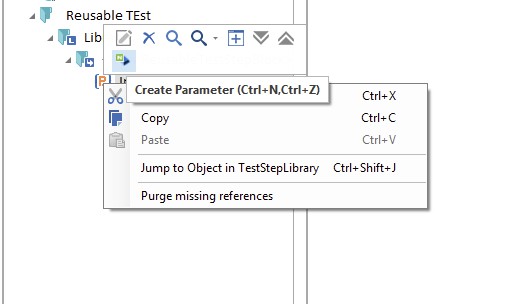
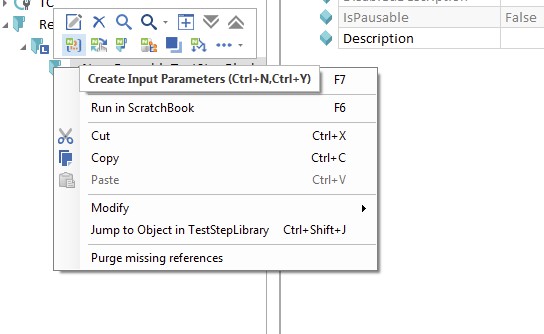
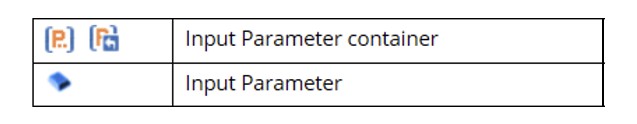
1. Left click on the Reusable TestStepBlock to be merged and keep the mouse button pressed.

2. Drag the selected Reusable TestStepBlock onto the TestStepBlock to be merged with.

3. Release the left mouse button.

4. Select the option Merge ReusableTestStepBlocks in the window that appear

33



INPUT PARAMETERS

Input parameters enable us to display and manage business-relevant dynamic values separated from the TestStepValues. Input parameters are only created in Reusable teststepblocks. Input parameter values are defined in the appropriate reference.

Symbols:

METHOD1:

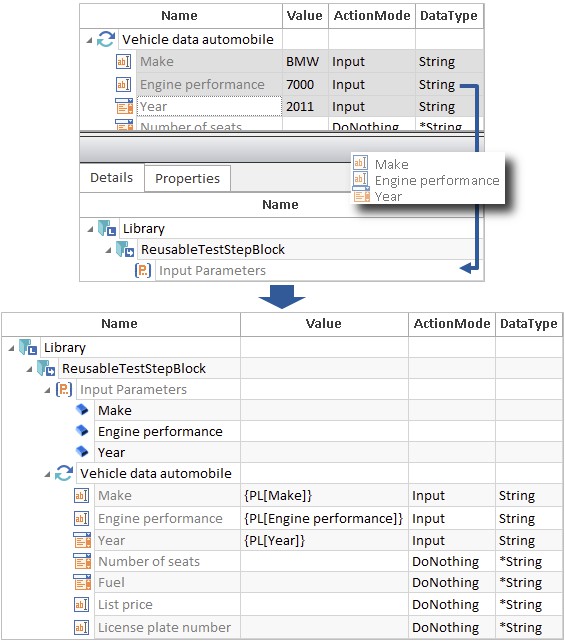
Steps to create Input Parameters: (Manual creation)

1. Select the option Create Input Parameters from the context menu of a Reusable TestStepBlock.   
 All input parameters are defined in this newly created input parameter container.

2. Select the option Create Parameter from the context menu of the input parameter container.

3. Define a name for the new input parameter.

34



4. Drag the newly created input parameter onto the chosen TestStepValue and drop it there. A link   
 is created. The link to the input parameter can also be created by entering the syntax into the   
 required TestStep Value.

Syntax: {PL[<Name of input parameter]} METHOD 2:

Steps to create input parameters via drag and drop

1. Select the option Create Input Parameters from the context menu of a Reusable TestStepBlock.   
 All input parameters are defined in the newly created Input Parameter container.

2. Drag the chosen TestStepValues onto the container with the name Input Parameter.

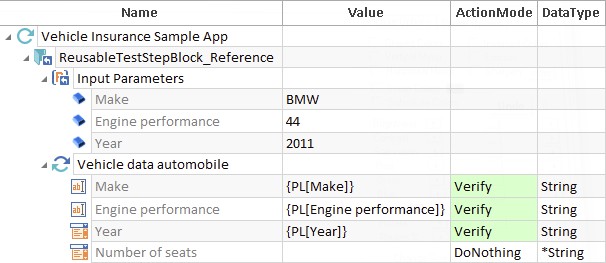
Modifying input parameters:

If input parameter names are changed or if an input parameter is moved within an input parameter container, the corresponding parameter links and conditions are automatically adapted.

Using input parameters in TestCases:

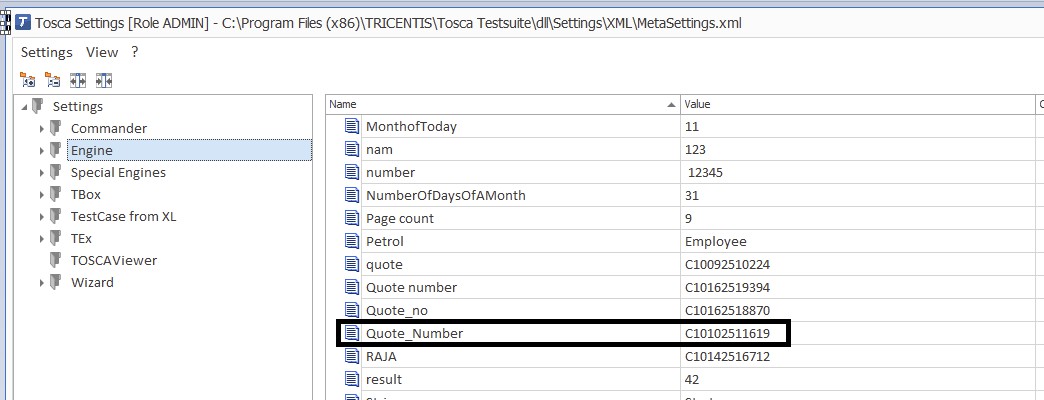
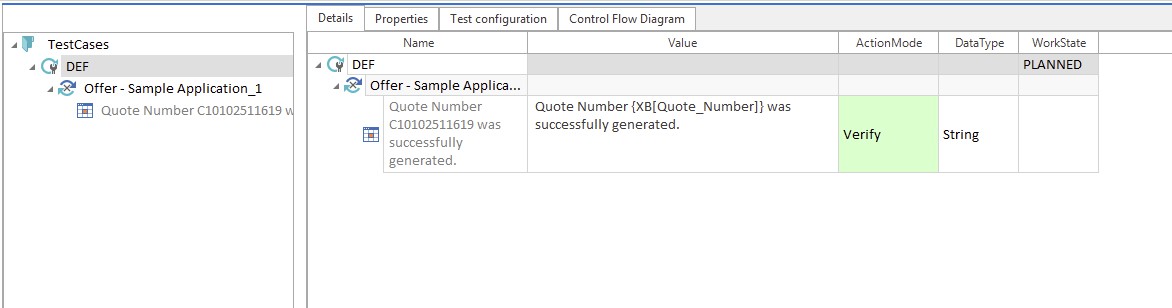
If a Reusable TestStepBlock is for instance used in a TestCase (referenced), test case specific data can be   
defined by using the input parameter reference. Input parameters in a TestStepBlock reference are

35



referred to as input parameter references since these also represent a reference to the input parameters of the referenced TestStepBlock.

36



Partial Buffer

This function is used to partially buffer a part of string from entire string. This partially buffered value can be used when required.

{XB[Buffer\_name]} is the syntax to partially buffer a value. Any Buffer name can be specified in which you want to store your buffer value.

For Example -

We need to buffer “C10132515927” from string - “Quote Number C10132515927 was successfully generated”.

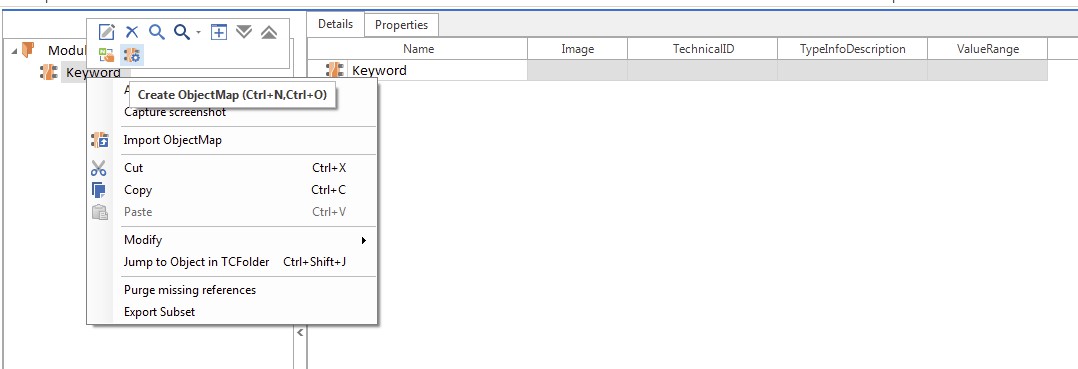
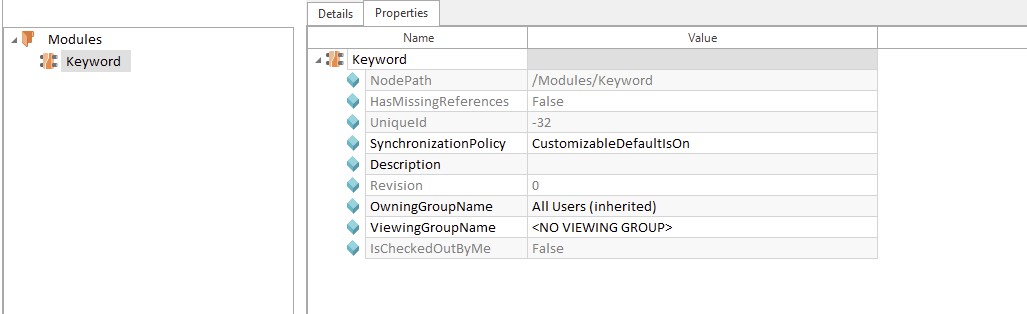
To do that we would write - “Quote Number {XB[Quote\_Number]} was successfully generated” C10132515927 is replaced by by {XB[Quote\_Number]}

Action mode is always Verify

Execution sets Buffer “Quote\_Number = C10132515927”

To Check the Value of stored buffer Goto Settings > Engine > <Buffer name>. List of saved buffer values are below -

37



KEYWORDS

Keywords allow individual steering requirements to be easily integrated into an engine. Specific

keywords are mainly used for steering screens. From a technical point of view, these are VB script files which are executed by the engine upon test execution.

For performing specific action that we are not able to perform directly in Tosca we need to write Vbscript code.

 The keyword files must be available in the directory of the Tosca Commander™ project that is used for   
 test execution.

In Tosca Commander™ standard installations, this directory is by default located at %TRICENTIS\_PROJECTS%\ToscaCommander\Keywords.

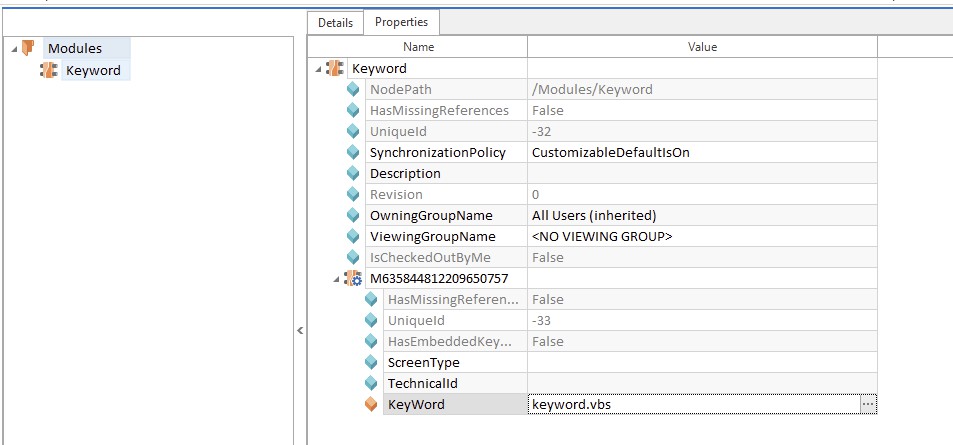
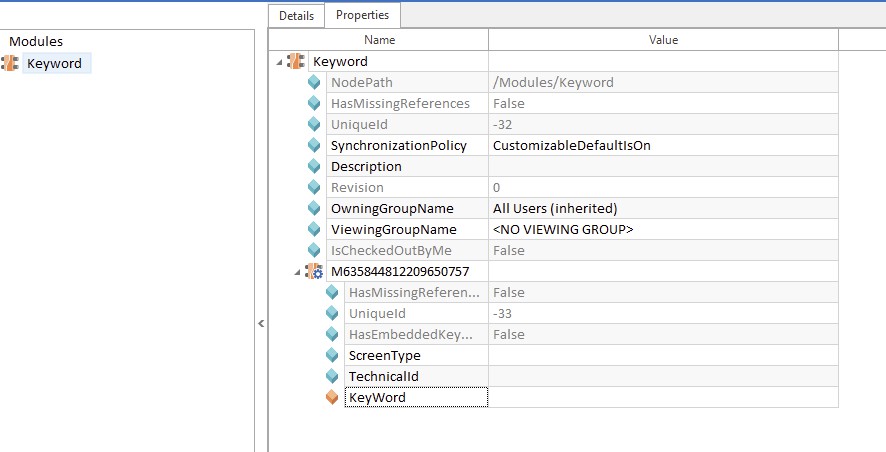
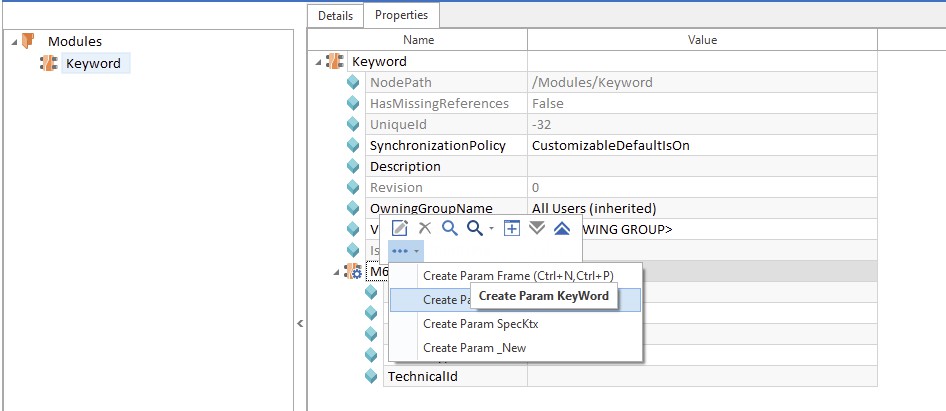
 Keywords are used in the Modules section in Tosca Commander™

Steps to create a keyword :

Create a new Module. You can attach keyword to existing module as well.

Right Click on the Module and Create object Map.

38

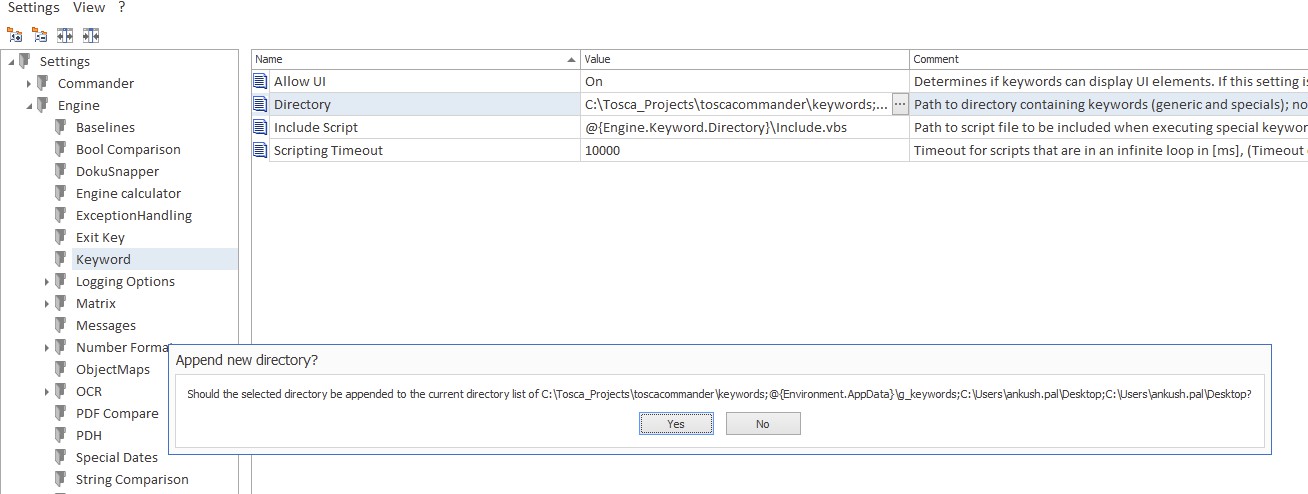


Go to Properties Right Click on the object map and Select Create Param Keyword.

Param Keyword is created.

Specify the name of “͘vbs” file you have created.

39

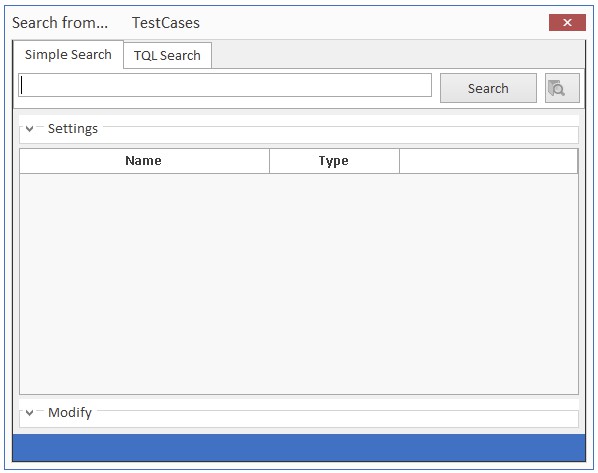


Keyword is attached to module.

We can use this vbscript file based on the requirement provided. Keyword Path as - Settings> Engine> Keyword> Directory.

NOTE : Keywords work for normal modules only not Xmodules (created using TBox).

40



TOSCA SEARCH

(SIMPLE SEARCH AND TOSCA QUERY LANGUAGE)

INTRODUCTION:

Tosca Commander allows any objects to be searched by providing two different search   
functions: [Simple Search](https://documentation.tricentis.com/en/900/content/tchb/simple_search.htm) and [TQL Search](https://documentation.tricentis.com/en/900/content/tchb/tql_search.htm) (TQL = Tosca Query Language).

When the search dialog is opened, the search function that has been most recently used is selected (Simple Search or TQL Search).

SIMPLE SEARCH:

We can start the search function with Home->Search..., by selecting Search from the context menu of an object. Fig 1. Shows the simple search dialog box.

41

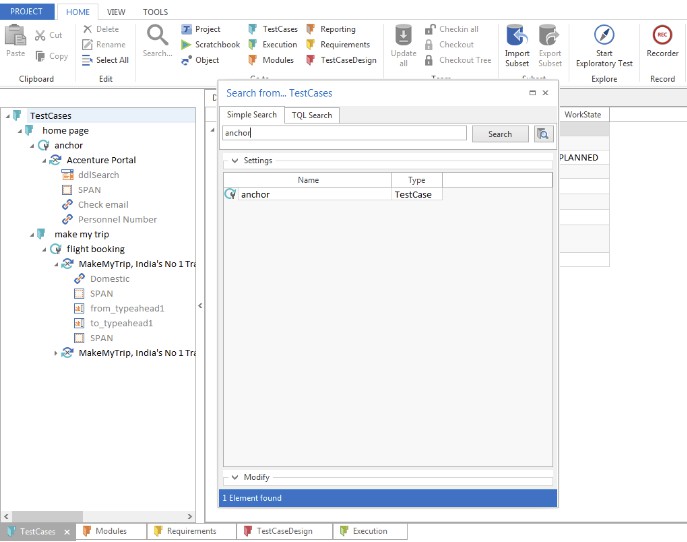


Fig 1. SIMPLE SEARCH DIALOG BOX

Description:

 In Tosca Commander, the search starts from the object that has been selected.

 The starting point of the search can be modified by dragging a new element over the   
 Search button

 The number of search results to be shown can be specified via the Options dialog.  The status bar can be used to scroll through the search results.

 Enable the simple search function via the Simple Search button. In addition, settings and   
 modify options can be shown by clicking on the left button of Settings and Modify.

 In the search field enter the term to be searched. This can for instance be the name of a   
 TestCase or the value of a TestStepValue.

PERRFORMING A SEARCH QUERY:

42

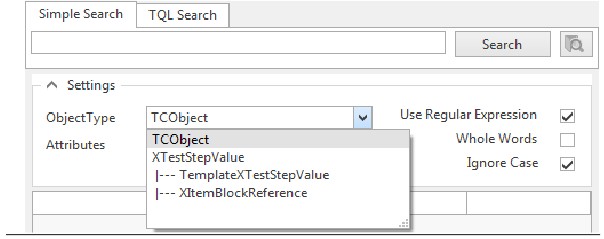


Fig 2. PERFORMING A SEARCH QUERY

STEPS TO BE FOLLOWED:

1. Select a starting point in a Tree View.

2. Start the search via Edit->Search... or with the help of the keyboard shortcut”Ctrl+F”͘

3. The search window opens.

4. Enable any additional options by clicking on the button.

5. Enter the search term into the input field and click on Search (“anchor”, as shown in Fig 2)͘

6. The search function will display all found elements in the results window.

SIMPLE SEARCH SETTINGS:

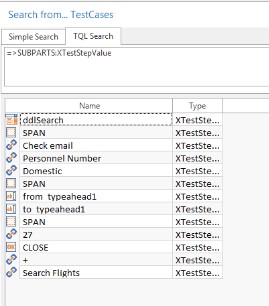
These settings helps users to simplify their search.

Fig. Simple Search settings.

 Selecting an option in Object Type navigates users to the respective level.

 Special Characters and wild cards can be used when we check “Use Regular Expression” option͘  Search field will accept case sensitive values when we check “Ignore Case” option͘

43



TQL (TOSCA QUERY LANGUAGE)

The TQL search is the other search function provided by TOSCA Commander. TQL (TOSCA Query   
Language), name itself suggests that a query is to be given to perform any search operation.

It is context-dependent. This means that the starting point has an effect on the search to be carried out.

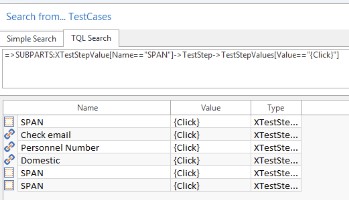
A search, which is performed on the basis of a project root element, produces different results from one based on the topmost Module folder.

TQL is deeply rooted in TOSCA. Even the normal, Simple Search dialog is based on it.

To understand what this means, we'll conduct a search operation for the TestStepValues (as shown below in the Fig 3.) under the TQL Search tab in the search dialog:

Fig 3. SEARCHING ALL TestStepValues

44



Some of the basic operators used in TQL search:

|  |  |
| --- | --- |
| Arrow operators | |
| -> | Examines the level beneath the starting point, |
| => | Examines all levels beneath the starting point, |

|  |  |
| --- | --- |
| Relational operators | |
| == | Equal,  =>SUBPARTS:XTestStepValue [Name=="SPAN"] returns the TestStep with value SPAN(Fig.4) |
| =i= | Equal, case insensitive. |
| != | Unequal. |
| !i= | Unequal, case insensitive |
| < | Less than,  =>SUBPARTS:XTestStepValue[(Name=="DateOfBirth")AND(Value<"01/01/1980")] returns  the TestStepValue DateOfBirth(For Example). |
| > | Greater than,  =>SUBPARTS:XTestStepValue[(Name=="DateOfBirth")AND(Value>"12/12/1970")] returns  no result |
| <= | Less than or equal,  =>SUBPARTS:XTestStepValue[(Name=="DateOfBirth")AND(Value<="01/01/1980")] returns  the TestStepValue DateOfBirth |
| >= | Greater than or equal,  =>SUBPARTS:XTestStepValue[(Name=="DateOfBirth")AND(Value>="12/12/1970")] returns  the TestStepValue DateOfBirth |

45

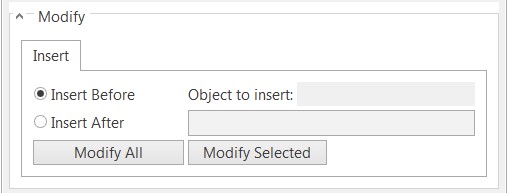


Fig. 4.TQL Query using Operators

NOTE:

 Each query starts with the arrowOperator => or ->.

 The expression ->SUBPARTS returns the subelements as results, whereas =>SUBPARTS returns   
 the subelements of the subelements as well as the hierarchies beneath.

[Modifying objects:](javascript:void(0);)

If the search was successful, you can use the option Modify to insert objects either before or after the search results - if the conditions mentioned below are met.

In order to do so, mark all results to be modified in this manner. Drag the required object onto the

Modify All or Modify Selected button in order to add it there. Only one object can be added at a time. Click on the Modify All button to confirm your selected action.

Conditions:

 All objects, which can be added to a TestCase, can be edited in the search results via Modify.   
 Except for: Reusable TestStepBlocks

 All objects which can be added to a TestCase, and all TestCases which can be added to an   
 ExecutionList, are insertable objects.

The below figure shows and explains the different options for modifying.

Modify options:

Attribute Description

Insert The selected object is inserted before the object in the search result.

Before

Insert The selected object is inserted after the object in the search result.

After

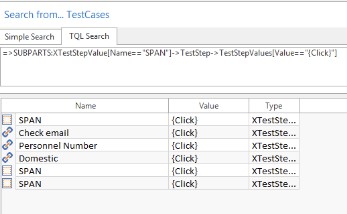
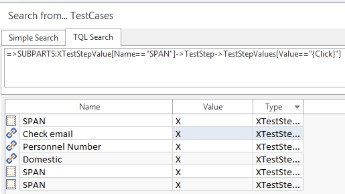
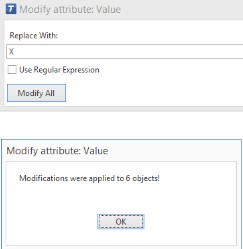
Modify The selected object is added to all objects in the search results that meet the specified

All conditions.

Modify The selected object is only added to the objects that are marked and meet the conditions.

Selected

46

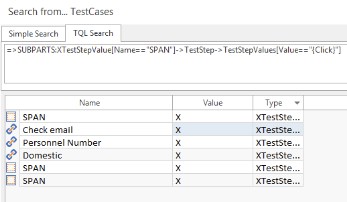


For instance, modifying all the TestStepValues having value as “{Click΃” to “X” is shown below͘

Fig.5 before Modification

Fig.6 after Modification

47



Virtual Folder:

Virtual folders are folders with no assigned objects, but an assigned search function. When a virtual

folder is opened, the stored search is performed and the results are displayed. The name ‘virtual’ points   
out that this folder only displays search results. The actual position of the shown objects remains   
unchanged.

Each TQL query can be saved in the workspace using virtual folders. The most recent search results can be accessed any time, even if Tosca Commander was closed in the meantime.

Since virtual folders represent a query they can be created in every folder and may display every query as search result, depending on the query.

When an object in the virtual folder is edited or deleted, it is also edited and deleted at its actual

position. When a virtual folder is deleted, the objects persist, as the virtual folder only represents one search and one view.

Steps to create Virtual folder:

1. Select the folder in which a new virtual folder should be created.

2. In the context menu choose Create Virtual Folder.

3. Tosca Commander creates a new virtual folder at the selected position and automatically offers to   
 rename it. You can either assign a name to the new virtual folder or press Enter to use the default   
 name Virtual Folder.

Virtual folders can also be created in the search dialog by clicking on the Virtual folder button. The TQL   
query, which has been performed in the search dialog, is then saved as virtual folder as shown below.

We can edit and even modify queries as and when required.

Fig.7 Creating Virtual Folder.

48