

## STUDIO™ 4.3 Exercise Book

Training Material

---

Document: : StudioExercises.doc

Date : 11/25/2010

Author name : Shay CesaryShay Cesary

Shay Cesary

## Table of Contents

<b>Preface</b> .....	<b>4</b>
How can you use this exercises booklet: .....	4
<b>Exercise Structure</b> .....	<b>5</b>
Test Name .....	5
Short description .....	5
List of modules used .....	5
Exercise details .....	5
Test Details .....	5
<b>1. Analysis</b> .....	<b>6</b>
1.1 tAnalysis_GetData .....	6
1.2 tAnalysis_PassFail .....	8
1.3 tAnlysis_Transform .....	12
<b>2. Canvas</b> .....	<b>17</b>
2.1 t3a_delay .....	17
2.2 t3a_message .....	19
<b>3. CLI and Network Protocols</b> .....	<b>20</b>
3.1 tCLI_CommandShell .....	20
<b>4. Flow</b> .....	<b>22</b>
4.1 tDelay_1 .....	22
4.2 tEnd_1 .....	24
4.3 tFlow_EndTerminate .....	26
4.4 tFlow_Event .....	28
4.5 tFlow_LoopIP .....	30
4.6 tFlow_Parallel .....	31
4.7 tFlow_VarAssign .....	33
4.8 tFlow_While .....	35
4.9 tLoop_1 .....	37
4.10 tLoop_2 .....	38
4.11 tLoop_3 .....	39
<b>5. Assets</b> .....	<b>41</b>
5.1 tRsrc_Calc .....	41
5.2 tRsrc_Dll .....	43
5.3 tRsrc_TCL .....	45
<b>i. Runner</b> .....	<b>48</b>
5.4 tRunner_Report .....	48
<b>6. Utilities</b> .....	<b>51</b>

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

6.1	tUtil_FileEngine .....	51
6.2	tUtil_MatShell .....	54
6.3	tUtil_WriteToFile.....	56
<b>7.</b>	<b>Variables.....</b>	<b>59</b>
7.1	tVar_Expressions .....	59
7.2	fVar_Function.....	61
7.3	tVar_FunctionCall.....	63
7.4	tVar_Sessions .....	65

This material is confidential and proprietary.  
Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## Preface

This booklet is a part of the Studio 4.3 training kit. Use these exercises with the test solutions for the exercises (part of the kit).

Please notice, that the solutions have many of the steps (and hence, automatic variables) renamed.

### **How can you use this exercises booklet:**

- a. Pick a relevant exercise according to its description and list of modules used
- b. Read the Exercise Details
- c. Plan a course of action:
- d. What are the inputs (commands, user input, files, etc)
- e. What process needs to be done (Analysis, communication, using assets)
- f. Are there any decision making steps (a case structure)
- g. Are there repetitive actions (Loop, While)
- h. What are the outputs (messages, report, output files)
- i. Start building your test! (use the exercise name as the test name)

Do not have a clue?

No worries!

Follow the step-by-step description in **Test Details**

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## Exercise Structure

Every exercise in this booklet will have the following sections

### Test Name

Start every exercise by creating the test in your exercise folder (If you have not already, create a local folder named Exercises)

Rename the test to be identical to the exercise. The name consists of a first letter (**t** for a standalone test or **f** for a dependant function) and a camel casing short name.

### Short description

Use the description to decide the objective of the test and its relevance to your training.

### List of modules used

During the training, you will learn to work more and more modules. Apart from the basic message, start and end – use the module listing to match the exercise to your training phase.

### Exercise details

Use this section to create the test yourself. Please remember that there is usually more than one way to execute the exercise. You can use the modules mentioned above – you are encouraged to try other modules in other ways!

### Test Details

A step-by-step description of *one* solution to the exercise. It is similar to the built-in solution appended to the Studio Installation. The solutions might differ by step names (use function key F2 to rename steps).

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 1. Analysis

### 1.1 tAnalysis\_GetData

#### 1.1.1 Description

The test will extract data from a text output by using transformation

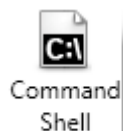
#### 1.1.2 Modules Used

- a. Command Shell
- b. Set Session
- c. Command Tool
- d. Transformation
- e. Message

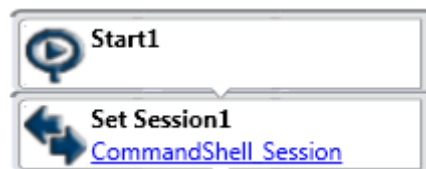
#### 1.1.3 Details

- a. Use the ipconfig command to get the Default Gateway IP address
- b. Display a pop-up window with the IP address

#### 1.1.4 Test Details:



- a. Add a command shell tool (from the Software stencil)
- b. Double click the command shell tool, a mini-canvas opens as a new document (Notice that the Start and Set Session steps are automatically added)

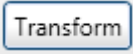


- c. Click on the Start Recording button and type the command ipconfig inside the terminal window. The Command step added to the canvas.

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.



- d. Edit the command step, click on the  button
- e. Highlight the default gateway IP address, right click and select “Extract” – the extraction pop up appears


- f. Change the extraction method to Before & After, and click OK

Extraction Method

Before & After 

- g.
- h. Two SPLIT transformations added to the transformation list.
- i. Notice: The transformation output variable appears in the variable pane



- j. Switch back to the main canvas, add a message  with the following text (use ctrl+insert to add the variable name instead of typing)

The default Gateway is:{Command1\_Transform1.Measurement1 }

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 1.2 tAnalysis\_PassFail

### 1.2.1 Description

The test will ping a user entered URL and check the sites availability

### 1.2.2 Modules Used

- a. Instruction
- b. Asset (Executable)
- c. Case
- d. Pass
- e. Fail
- f. Group
- g. Message
- h. End

### 1.2.3 Details

- a. Get a URL from the user
- b. Ping the address
- c. Set a PASS or FAIL status in the report
- d. If the ping was successful – pop a message ‘OK’
- e. If failed – pop a message with the exact reason (Unreachable, Timed out, or command not successful)

### 1.2.4 Test Details:



- a. Add an Instruction step **Instruction** to the canvas (from the Utilities stencil)
- b. Edit the Instruction (double click), set the text to be “Please enter a web address” and add a user input:



Type:	textbox
Label:	Web_Address

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.



Default Value: [www.qualisystems.com](http://www.qualisystems.com)  
 Output Name: webAddress  
 Output Type: String  
 Output Dimension: Scalar







Header:  

Instruction

Please enter a web address

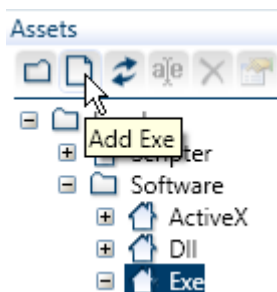
☐ Timeout  Sec

**User Input**

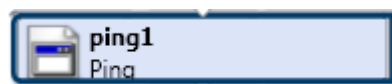
Type	Label	Data source	Default Value	Output Name	Output Type	Output Dimension
TextBox ▼	Web_Address		www.qualisystems.com	webAddress	String ▼	Scalar ▼
Buttons ▼		<[["OK"]]>	OK	Result	String ▼	Scalar ▼

- c. Add an *asset* to the Studio environment (from the Assets pane, select Local\Software\exe and click on the *Add Exe* button)



Asset Name: Ping  
 Path: C:\WINDOWS\system32\ping.exe

- d. Add the Ping asset to the canvas (drag it from the Assets pane), edit it (double click) and in the Arguments field enter the variable {Instruction1.webAddress} (variable can be entered from the variable pane by clicking CTRL+Insert)



- e. Activate the PING step (select it and click on the function key F7). Notice that two automatic variables were added to the variable pane:

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

Ping1.ExitCode (a numeric scalar) – 0 is command runs OK

Ping1.Result (A string scalar) – The text output

- f. Add a case structure to the canvas (from the Flow stencil) edit it and set the condition according to the ExitCode variable



- g. To the left branch of the case, add a PASS step *Pass* from the Analysis and Reports stencil

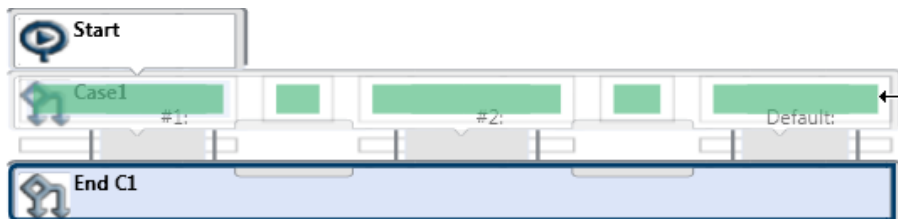


- h. Add a message step (from the utilities stencil) *Message*. Edit this step and type *OK*

- i. At the right branch of the CASE structure, add a Group (from the Utilities stencil)



- j. Edit the group – this will open as a mini-canvas (a separate document named tAnalysis\_PassFail.Group1) already with a start step.
- k. Add a Case structure (from the Flow stencil). Drag the right edge of the case so it will include three branches



Edit the case conditions

Condition#1 {Ping1.Result} contains Unreachable

Condition#2 {Ping1.Result} contains Timed out



- l. To the each branch add a *Fail* step *Fail* from the Analysis and Reports stencil

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

- m. To the leftmost branch (#1 Unreachable) add a *Message* step from the *Utilities* stencil. Edit the message and type *Address is unreachable*
- n. To the middle branch (#2 Timed out) add a Message step from the Utilities stencil. Edit the message and type Time out on request
- o. To the rightmost branch (Default) add a Message step from the Utilities stencil. Edit the message and type Command Not Successful



- p. After the End Case step, add an END step *End* (from the Flow stencil)
- q. Back to the main test canvas – Add an END step

This material is confidential and proprietary.  
Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 1.3 tAnlysis\_Transform

### 1.3.1 Description

The test will get the user IP and the default gateway, and will create a list of ping results of all the addresses between the machine IP and the default gateway.

### 1.3.2 Modules Used

- a. Command Shell
- b. Set Session
- c. Command
- d. Transformation
- e. Assign Variable
- f. While
- g. Case

### 1.3.3 Details

- a. Use IPCONFIG command to get the machine IP and the default gateway address
- b. Ping all the addresses from the machine address up to the default gateway address
- c. Build a table of two rows: on the left row, type the pinged address. On the right row the average round trip time – if no such time is available (Timed out, address unreachable, etc.) write the word Timed Out on the right column

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

	0	1
0	192.168.42.1	0
1	192.168.42.2	0
2	192.168.42.3	0
3	192.168.42.4	0
4	192.168.42.5	Timeout
5	192.168.42.6	0
6	192.168.42.7	0
7	192.168.42.8	0
8	192.168.42.9	0
9	192.168.42.10	Timeout

## 1.3.4 Test Details:

- a. Create the following variables (CTRL+ALT+V)

BASE, A string scalar

Current\_IP, A numeric scalar

Result\_Matrix, A string matrix



Command  
Shell

- b. Add a Command Shell step from the Software stencil, edit it by double click on it

- a. The Command Shell opens as a mini-canvas, a separate document named *tAnlysis\_Transform.Command\_Shell1*. Click on the *Start Recording* button

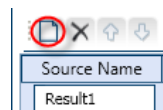
- b. In the terminal window, type the command *ipconfig* (and click Enter)



- c. From the Analysis stencil add a Transform step *Transform*. Click on the function key F2 and rename the transform step Get IP details

- d. To add inputs and measurements, click on the  button on the top right




- e. The source and measure panes reveal. On the source pane (left), click on the ADD SOURCE button



- f. Select the result of the previous step: {Command1.Result} and click OK





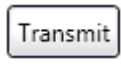


This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

- g. On the measurement pane (right) a measurement added automatically, named Measurement1 – select it and rename it BASE (this will be the base address for the ping command)
- h. In the INPUT pane, mark the text IP Address. . . . . : Right click on the marked text, and select Extract...
- i. From the extraction dialog, select the extraction method SPLIT AFTER
- j. The split transformation added to the left pane. Still on the INPUT pane (now updated after the split) Mark area right after the IP address (include the word Subnet). Right click on the marked area, and select Extract...
- k. From the extraction dialog, select the extraction method SPLIT BEFORE
- l. On the transformation pane, click ADD FROM LIBRARY  and select REVERSE (this will reverse the order of characters in the IP address text)
- m. Again, click ADD FROM LIBRARY on the transformation pane to add another split step (this step will remove the last number from the IP):
  - 1. Match String: . [a single dot]
  - 2. Regular Expression (not checked)
  - 3. Include Match String (Checked)
  - 4. Start at index: (0)
  - 5. Option: After match
- n. Add another REVERSE (click ADD FROM LIBRARY  and select REVERSE).
- o. Create another measurement: On the measurement pane (top right) click on the ADD MEASURE button . This will add a measurement, named Measurement1 – select it and rename it GW (this will hold the gateway last number in the address)
- p. Similar to the BASE measurement, add the following steps:
  - 1. SPLIT (After match, Match string=Default Gateway . . . . . :, match not included)
  - 2. SPLIT (Before match, Match string=PS C:\> match not included)
  - 3. REVERSE
  - 4. SPLIT (Before match, Match string=. match not included)
  - 5. REVERSE



This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

- q. The next measurement (My\_IP) is very similar to the GW measurement. Hence, please mark the GW measurement and click on the COPY MEASURE  button. Now, click on the PASTE MEASURE  and rename the newest measure My\_IP
- r. Keep the My\_IP measurement highlighted, and select the first transformation (named SPLIT) – change the MATCH STRING value to IP Address. . . . . :
- s. Back to the main test (you can close the command shell mini canvas) drag the Current\_IP variable from the Variables pane to the canvas. Double click to edit this step
- t. Set the module to ‘From Expression’ ☐ From Value ☒ From Expression and the expression to: {str2num(Get\_IP\_details.GW)}
- u. From the FLOW stencil, add a WHILE loop . Edit it by double clicking and set its condition to be while {Current\_IP} is less or equal to {str2num(Get\_IP\_details.My\_IP)}
- 
- v. Inside the while loop, add another command shell (from the software stencil). Double click to edit. A mini-canvas document will open at a new tab
- w. Click on the START RECORDING button, inside the WRITE tab (below the terminal) type ping {(Get\_IP\_details.Base) + num2str(Current\_IP)} and click on the Transmit button 
- x. Add a new transform step, from the Analysis and Report stencil. Edit it and click on the  button on the top right.
- y. On the source pane (left), click on the ADD SOURCE  button and add the previous step result {Command2.Result}. On the measurement pane (right) a measurement added automatically, named Measurement1 – select it and rename it ResponseTime
- z. In the transformation pane add the following steps:
1. SPLIT (After match, Match string= Average = ,match not included)
  2. SPLIT (Before match, Match string= ms ,match not included)
- aa. Click OK and switch back to the main test (you can close the mini canvas document)

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

- bb. Now, we will update the result table (Result\_Matrix). Drag the Result\_Matrix variable from the variable pane to the canvas. Double click to edit
- cc. Edit the target field to {Result\_Matrix(While1.Index,0)}. This will set the value of the first column, while the row will be according to the while loop index.
- dd. Set the module to 'From Expression' ☐ From Value ☒ From Expression and the expression to: {Get\_IP\_details.Base + num2str(Current\_IP)} which is the IP address we are pinging
- 
- ee. From the flow stencil, add a CASE structure Parallel . Double click to edit the case conditions. Set the condition to be {Command2\_Transform1.ResponseTime} is not null
- 
- ff. Drag the Result\_Matrix variable (from the variable pane) into the case structure left branch (#1). Double click to edit it,
- gg. Edit the Target: field to {Result\_Matrix(While1.Index,1)}. Set the module to 'From Expression' ☐ From Value ☒ From Expression and the expression to: {Command2\_Transform1.ResponseTime}
- hh. Copy the Result\_Matrix2 step you just created (use CTRL+C) and paste it (use CTRL+V) in the right branch of the case (Default). Edit the step (Result\_Matrix3) by double clicking.
- ii. Set the module to 'From Value' ☒ From Value ☐ From Expression and the value to: *Timeout*
- jj. Right after the End Case1 step (but before the End While), drag the Current\_IP variable from the variable pane. Double click to edit
- kk. Set the module to 'From Expression' and the expression to: {Current\_IP + 1 }

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.



## 2. Canvas

### 2.1 t3a\_delay

#### 2.1.1 Description

The test demonstrate working on the canvas with the Delay and Message modules

#### 2.1.2 Modules Used

- a. Message
- b. Delay
- c. End

#### 2.1.3 Details

- a. Display a message “Test has started”
- b. Wait for 5 seconds
- c. Display a message “Test in progress”
- d. Wait for 5 seconds
- e. Display a message “Test has ended”

#### 2.1.4 Test Details:

- a. Create a test named t3a\_delay in your exercise folder



- b. From the Utilities stencil, drag the Message Message module to the canvas (right below the Start step). Double click to edit the module and type: Test has started



- c. From the Flow stencil, drag the Delay Delay module. Double click to edit the module, change the Delay to 5 seconds and check the “Show Progress” checkmark
- d. Select the two modules (Message1 and Delay1) by dragging a rectangle around them

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

- e. Once selected, copy them (either by CTRL+C or by right clicking, and selecting copy from the right click menu) and paste them right below the last step (CTRL+V or by right clicking, and selecting paste from the right click menu)
- f. Edit the new message (Message2) and change the text to Test in progress
- g. Mark the space below the last step (it should turn light blue). From the Utilities stencil, double click on the message module – this will add a message step to the canvas. Edit it and type in it Test has ended



- h. Add an END step End from the Flow stencil

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 2.2 t3a\_message

### 2.2.1 Description

The test demonstrate working on the canvas and the Message module






### 2.2.2 Modules Used

- a. Message
- b. End

### 2.2.3 Details

- a. Display a pop up to the user, with the text “Welcome to TestShell Studio course! – This is the message tool”

### 2.2.4 Test Details:

- a. Create an exercise folder to store your solutions: In the Tests pane, click on the Add Folder button . By default it is named Folder1 – rename it to My\_Exercises
- b. Create a test named t3a\_delay in your exercise folder: While highlighting the folder created in the previous step, click on the Add Test button  - by default, it will create a test named Test1. Rename it to t3a\_delay
- c. From the Utilities stencil in the Tools pane, drag a Message step  to the canvas – right below the Start step. Double click to edit the step
- d. In the message field, type the message, Welcome to TestShell Studio course! – This is the message tool. Click OK to close the module
- e. From the Flow stencil, add an END module, right below the message.
- f. Save ( in the ribbon) and close  the test

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 3. CLI and Network Protocols

### 3.1 tCLI\_CommandShell

#### 3.1.1 Description

Working with User interaction, Command Shell and user variables


#### 3.1.2 Modules Used

- a. Instruction
- b. Command Shell
- c. Command tool
- d. End

#### 3.1.3 Details

- a. Get from the user a command and parameter (default ipconfig /all)
- b. Send the command and parameter on a DOS prompt
- c. Get the windows version
- d. Display the response to the user command

#### 3.1.4 Test Details:


- a. Create two variables (by clicking CTRL+ALT+V):
  - 1. CommandParam (String Scalar, Initial Value: /all)
  - 2. VarCommand (String Scalar, Initial Value: ipconfig)
- b. Add an Instruction step **Instruction** (from the Utilities stencil)
- c. In the header field, type “Get the command and parameter”
- d. In the “User Input” section, add inputs by clicking on the  button. Add two user inputs (Note: in the default value, click CTRL+INS to insert the variables):
  - 1. Type:TextBox, Label:Command, Default value:{ VarCommand}, Output Name: Command, Output Type: String, Output Dimension: Scalar

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

2. Type:TextBox, Label: Parameter, Default value:{CommandParam}, Output Name: Parameter, Output Type: String, Output Dimension: Scalar



- e. Add a Command Shell (from the Software Stencil)  . Edit it by double click on it. The Command Shell opens as a mini-canvas, a separate document named tCLI\_CommandShell.Comand. Click on the Start Recording button
- f. Type, in the terminal pane, the command cmd ver (click enter) – A command step is added to the mini-canvas flow
- g. From the Command Shell Tools stencil (This stencil appears ONLY within the Command Shell



mini-canvas) add the Command tool  . Edit it by double click

- h. Set the command shell panel with:
  1. Command: {VarCommand} {CommandParam} /Note: use Ctrl+INS to select the variable
  2. Termination string: C:\>
  3. Command Timeout: 5
- i. Close the tCLI\_CommandShell.Comand document, and switch back to the main test.
- j. Add Another Instruction module (Utilities module). In the text area type (or use CTRL+INS):  
This is the command's output:  
{Variable\_Command.Result}
- k. Switch to the advanced tab of the Instruction module. Set the window size to 500X800 pixels

Window size (pixels)

☐ Auto
 ☒ Manual
 Width  Height

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 4. Flow

### 4.1 tDelay\_1

#### 4.1.1 Description

The test demonstrates the “Show progress” option in delay

#### 4.1.2 Modules Used

- a. Start
- b. Instruction
- c. End (Test option)
- d. Loop
- e. Delay

#### 4.1.3 Details

- a. Display a message “The two kinds of delay will alternate - with or without showing progress”
- b. Allow the user to end the test by clicking on an “End Test” button
- c. While the message appears, run two consecutive delays, one with and one without ‘show progress’

#### 4.1.4 Test Details:

- a. Create a test named tDelay\_1 in your exercise folder

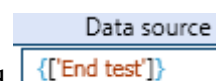


- b. Add to the canvas another Start step **Start** (from the FLOW stencil) – place it to the right of the original Start step (at the same level)



- c. To the new Start, add an Instruction step **Instruction** (from the Utilities stencil). Double click to edit it.
- d. Set the text to be “The two kinds of delay will alternate - with or without showing progress”

- e. Rename the OK button to read “End Test”. To do so, rename the Data Source field



- f. Switch to the Instruction ‘Advanced’ tab, set the windows location to 50x50

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

Window location (% percentage)

☐ Auto

☒ Manual

X

50

Y

50

- g. From the Flow stencil, add an END step. Double click to edit it. Change its setting to ‘Stop this test’  
 “ ☒ Stop this test (by default its ‘Stop This branch’).



- h. Now, back to the original flow (the left side Start) – Add a loop <sup>Loop</sup> (from the Flow stencil).  
 Double click to edit it – and set its iterations to 100.



- i. Within the loop add two delays <sup>Delay</sup> (from the Flow stencil). Double click to edit them, set them both to 5 Seconds
- j. Set the first to display progress, by checking ☒ Show progress
- k. Make sure the second on will not show progress ☐ Show progress

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 4.2 tEnd\_1

### 4.2.1 Description

The test demonstrates the End module options 'Stop this Branch' VS. 'Stop this Test'

### 4.2.2 Modules Used

- a. Start
- b. Delay
- c. Instruction
- d. End (Branch)
- e. End (Test)


### 4.2.3 Details

- a. Run 5 delay modules in parallel (10, 20, 50, 100 and 200 seconds)
- b. End each branch with an END (stop this branch)
- c. Display a pop-up to the user 'Click to stop all branches'


### 4.2.4 Test Details:

- a. Create a test named tEnd\_1 in your exercise folder



- b. Add a Delay step  (from the Flow stencil)

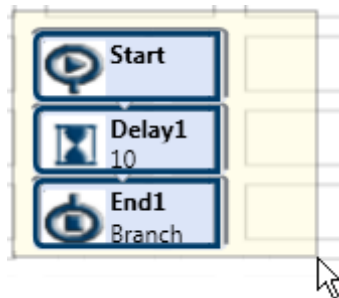


- c. Add an End step  (from the flow stencil). By default, a new End module is set to 'Stop This Branch'
- d. Drag a rectangle around the flow you created. Copy (CTRL+C) and paste it (CTRL+V) four times on the canvas

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.





- e. Edit (double click) each of the delay steps to be 10, 20, 50, 100 and 200 seconds respectively
- f. Add another Start module (from the Flow stencil)



- g. Add an Instruction module *Instruction* (From the Utilities stencil). Double click to edit it
  1. Set its text to : Click to stop all branches
  2. Change the OK button writing to be *STOP*. Do it by editing the Data Source

## User Input



Type	Label	Data source
Buttons ▼		{['Stop']}

- h. Right below the instruction step add an END step (From the Flow stencil). Double click to edit it:  
Set it to “Stop this test” ☒ Stop this test

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 4.3 tFlow\_EndTerminate

### 4.3.1 Description

The test demonstrates the difference between the End execution and Terminate modules

### 4.3.2 Modules Used

- a. Message
- b. Delay
- c. End (Execution)
- d. Terminate


### 4.3.3 Details

- a. In order to see the actual difference – it's best to view the execution log (enabled from the View button on the ribbon) – the status will be 'Finished' for the END step and 'Terminated' for the Terminate step.
- b. Allow the user to terminate the test by clicking OK on a message 'Click to Terminate'
- c. End the execution after 15 seconds (three consecutive delays of 5 seconds each)


### 4.3.4 Test Details:

- a. Create a test named tFlow\_EndTerminate in your exercise folder



- b. Add a Delay step  (from the Flow stencil). Double click to edit it – set it to 5 seconds and make sure the 'Show progress' checkmark is not checked ☐ Show progress
- c. Highlight the Delay step you created, Copy (CTRL+C) and paste (CTRL+V) the step twice (So you would have three consecutive delay steps)



- d. Right after the last (third) delay, add an END step  (from the Flow stencil)

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.



- e. Add another Start step **Start** (from the FLOW stencil) – place it to the right of the original Start step (at the same level)



- f. Add a message step **Message** (from the Utilities stencil) right below the new Start. Double click to edit it to display ‘Click to terminate’



- g. Right below the message, add a Terminate step **Terminate** (form the Flow stencil)

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 4.4 tFlow\_Event

### 4.4.1 Description

The test demonstrates the Event variable usage: The Set Event and Wait For Event modules

### 4.4.2 Modules Used

- a. Parallel
- b. Delay
- c. Message
- d. Set Event
- e. Wait For Event
- f. End (Branch)

### 4.4.3 Details

- a. Display two messages “Message 1” and “Message 2” in parallel (each one for 5 seconds)
- b. Each message should appear after a random delay (between 0 to 10 seconds)
- c. Run the test a few times
- d. Now, change the test so message 1 will ALWAYS appear before message 2

### 4.4.4 Test Details:

- a. Create a test named tFlow\_Event in your exercise folder



- b. Add a parallel structure **Parallel** (from the Flow stencil)

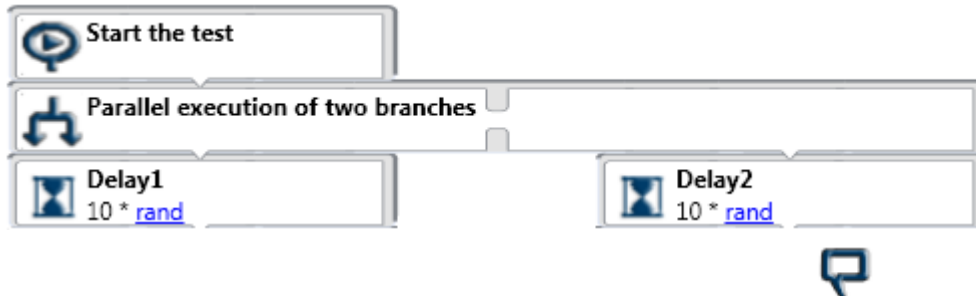


- c. Add a Delay step **Delay** (from the Flow stencil) into one of the branches of the Parallel. Double click to edit it – in the delay field, type `{10 * rand}`

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

- d. Copy the delay module (highlight it and click CTRL+C) to the other branch



- e. Add to the left branch (right below the Delay) a Message step **Message** (from the Utilities stencil).

Double click to edit it to display “Message 1”. Check the time-out check mark ☒ Timeout  Sec

- f. Copy (CTRL+C) the message to the second branch (CTRL+V) – edit it to display “Message 2”

- g. Run the test: it will display the messages after random delays, in random order.

- h. Create a variable (Ctrl+Alt+V)

1. Name: MsgOneDone
2. Type: Event
3. Dimension: Scalar



- i. Add a Set Event step **Set Event** (From the Variable Stencil) right below Message 1. Double click to edit it. Set the Event field to { MsgOneDone} - you can use (CTRL+INS) to insert the variable name



- j. Add a Wait For Event step **Wait For Event** (From the Variable Stencil) right above Message 2. Double click to edit it. Set the Event field to {MsgOneDone} - you can use (CTRL+INS) to insert the variable name



- k. Add an END step **End** (from the Flow stencil) just below the End Parallel

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 4.5 tFlow\_LoopIP

### 4.5.1 Description

Pinging six consecutive IP addresses

### 4.5.2 Modules Used

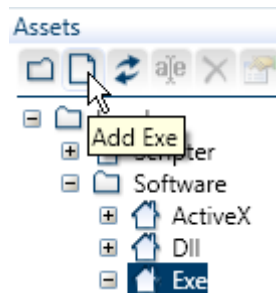
- a. Loop
- b. Asset (Executable – PING)
- c. End

### 4.5.3 Details

- a. Activate a Ping command, addressing 192.168.1.0 through 192.168.1.5

### 4.5.4 Test Details:

- a. Create a test named tFlow\_LoopIP in your exercise folder
- b. Add an asset to the Studio environment (from the Assets pane, select Local\Software\exe and click on the Add Exe button)



- 1. Asset Name: Ping
- 2. Path: C:\WINDOWS\system32\ping.exe



- c. Add a loop structure **Loop** (from the Flow stencil). Double click to edit it to iterate 6 times.
- d. Drag the Ping asset you created into the Loop. Double click to edit it, in the arguments field type 192.168.1. {Loop1.Index }

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 4.6 tFlow\_Parallel

### 4.6.1 Description

The test will ping to addresses at the same time and report the results to the user

### 4.6.2 Modules Used

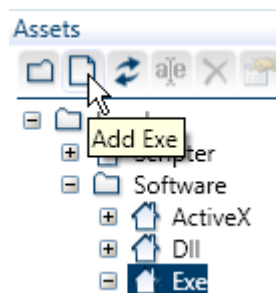
- a. Parallel
- b. Asset (Executable, Ping)
- c. Message
- d. End (Branch)

### 4.6.3 Details

- a. Ping the addresses 127.0.0.1 (localhost) and 192.168.42.1 simultaneously
- b. Display the text outputs to the user
- c. inform the user the test has ended

### 4.6.4 Test Details:

- a. Create a test named tFlow\_Parallel in your exercise folder
- b. Add an asset to the Studio environment (from the Assets pane, select Local\Software\exe and click on the Add Exe button)



- 1. Asset Name: Ping
- 2. Path: C:\WINDOWS\system32\ping.exe

This material is confidential and proprietary.  
Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.



- d. Add a parallel structure **Parallel** (from the Flow stencil)
- e. Drag a Ping asset step from the Asset pane to each of the branches of the parallel structure. Double click each of the Ping steps to edit them:
  - 1. On the left branch ping, set the argument field to 127.0.0.1
  - 2. On the right branch ping, set the argument field to 192.168.42.1



- f. Add a Message step **Message** (from the Utilities stencil) to each branch (right after the Ping steps). Double click each of the messages to edit them to display
  - 1. On the Left branch, the message will display `{Ping1.Result}`
  - 2. On the Right branch, the message will display `{Ping2.Result}`
- g. Add another message, right after the End Parallel, edit it to display “Test has ended”



- h. Add an END step **End** (from the Flow stencil)

This material is confidential and proprietary.  
Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.



## 4.7 tFlow\_VarAssign

### 4.7.1 Description

The test will demonstrate assignment of values and expressions to variables

### 4.7.2 Modules Used

- a. Assign Variable
- b. Message
- c. End
- d. (optional) MatShell

### 4.7.3 Details

- a. Set the value of two variables to 80 and 90, respectively
- b. Calculate the value of a third variable as a summation of the two variables
- c. Display the values of the three variables

### 4.7.4 Test Details:

- a. Create a test named tFlow\_VarAssign in your exercise folder
- b. Create three variables:
  - 1. Variable1 (Numeric, Scalar)
  - 2. Variable1 (Numeric, Scalar)
  - 3. Variable1 (Numeric, Scalar)
- c. Drag the variable Variable1 from the variable pane to the canvas. Double click to edit the step. Set the module from value ☒ From Value ☐ From Expression and set the value to 80
- d. Drag the variable Variable2 from the variable pane to the canvas. Double click to edit the step. Set the module from value ☒ From Value ☐ From Expression and set the value to 90
- e. Drag the variable Variable3 from the variable pane to the canvas. Double click to edit the step. Set the module from expression ☐ From Value ☒ From Expression and set the expression to {Variable1 + Variable2}

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.



- f. Add a message **Message** (from the Utilities stencil). Double click to edit the step to display  
*Calculate with the assign step: The value of Variable3 is {Variable3}*

*Calculate in the message: The value of Variable3 is {Variable1 + Variable2}*

- g. **Optional:** add a MatShell step (from the Utilities stencil). Double click to edit the step to:  
Variable3=Variable1 + Variable2



- i. Add an END step **End** (from the Flow stencil)

This material is confidential and proprietary.  
Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 4.8 tFlow\_While

### 4.8.1 Description

The test will scan a list of IP addresses until it reaches an unreachable address. It will display the address to the user

### 4.8.2 Modules Used

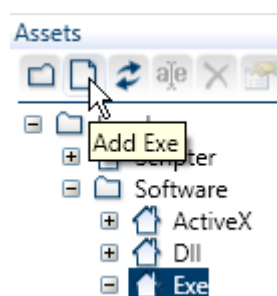
- a. While loop
- b. Asset (executable, Ping)
- c. Message
- d. End

### 4.8.3 Details

- a. Ping the addresses from 192.168.1.0 to 192.168.1.255
- b. If the ping results in an 'Unreachable' or 'Timed out', stop the scan, and display the last IP address to the user

### 4.8.4 Test Details:

- a. Create a test named tFlow\_While in your exercise folder
- b. Add an asset to the Studio environment (from the Assets pane, select Local\Software\exe and click on the Add Exe button)



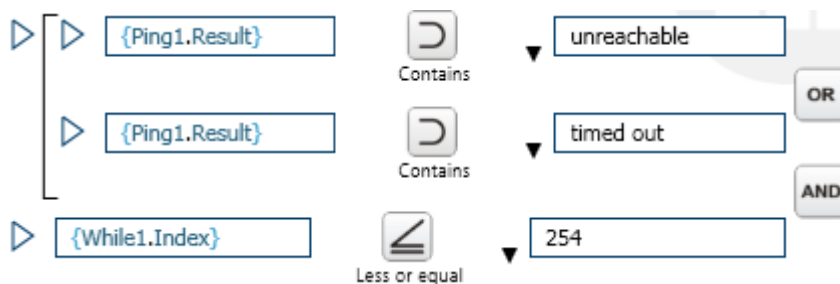
- 1. Asset Name: Ping
- 2. Path: C:\WINDOWS\system32\ping.exe

This material is confidential and proprietary.  
Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.



- c. Add a While loop **While** (from the Flow stencil). We will edit the module after the ping step is set (disregard the warning for now)
- d. Drag the Ping asset from the Assets pane inside the While loop. Double click to edit the Ping step, set the argument to: 192.168.1.{ While1.Index }
- e. Back to the While step, double click to edit the conditions (Notice: to group conditions, click SHIFT when selecting a condition)

(( {While1.Index} less or equal 254) AND (( {Ping1.Result} Contains unreachable) OR ( {Ping1.Result} Contains timed out))



- f. Add a message **Message** (from the Utilities stencil). Double click to edit the step to display:

*The last IP address is 192.168.1.{ While1.Index }*

*The result was:*  
*{Ping1.Result}*



- g. Add an END step **End** (from the Flow stencil)

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 4.9 tLoop\_1

### 4.9.1 Description

Repeat a message 10 times!

### 4.9.2 Modules Used

- a. Loop
- b. Message

### 4.9.3 Details

- a. Display a message “This message will repeat itself n more times” 10 times
- b. Each message will display a different n, according to how many messages are left...

### 4.9.4 Test Details:

- a. Create a test named tLoop\_1 in your exercise folder



- b. Add a loop structure **Loop** (from the Flow stencil). Double click to edit it – Iterations set to 10



- c. Add a message **Message** (from the Utilities stencil) inside the loop. Double click to edit the step to display:

*This message will repeat itself {10 - Loop1.Index} more times*

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 4.10 tLoop\_2

### 4.10.1 Description

The test demonstrates pointing an array with a loop index

### 4.10.2 Modules Used

- a. Assign Variable
- b. Loop
- c. Message

### 4.10.3 Details

- a. Display a message “This message will repeat itself xxx more times” 10 times
- b. The xxx will be a text (‘Ten’, ‘Nine’, ..., ‘One’)


### 4.10.4 Test Details:

- a. Create a test named tLoop\_2 in your exercise folder
- b. Create a variable (CTRL+ALT+V):
  - 1. Name: Time\_text
  - 2. Type: String
  - 3. Dimension: Vector
  - 4. Initial values: Ten, Nine, Eight, Seven, Six, Five, Four, Three, Two, One
- c. Drag the variable Time\_text from the variable pane to the canvas (by default it will set the variable to its default values)



- d. Add a loop structure  (from the Flow stencil). Double click to edit it – Iterations set to 10



- e. Add a message  (from the Utilities stencil) inside the loop. Double click to edit the step to display:

*This message will repeat itself {Time\_text(Loop1.Index)} more time(s)*

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 4.11 tLoop\_3

### 4.11.1 Description

The test demonstrate nested loops

### 4.11.2 Modules Used

- a. Start
- b. Loop
- c. Message
- d. Instruction
- e. End (stop test)

### 4.11.3 Details

- a. Display a message for any combination of the multiplication table (e.g. 4x4 is 16)
- b. In Parallel, display a window that allows the user to end the test

### 4.11.4 Test Details:

- a. Create a test named tLoop\_3 in your exercise folder



- a. Add a loop structure **Loop** (from the Flow stencil). Double click to edit it – Iterations set to 10
- b. Inside the loop, add another loop structure. Double click to edit it – Iterations set to 10



- c. Inside the inner loop, add a message **Message** (from the Utilities stencil) inside the loop. Double click to edit the step to display:

The result of  $\{\text{Loop1.Index} + 1\} \times \{\text{Loop2.Index} + 1\}$  is  $\{(\text{Loop1.Index} + 1) * (\text{Loop2.Index} + 1)\}$



- d. Add another Start step **Start** (from the FLOW stencil) – place it to the right of the original Start step (at the same level)

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.



- e. Add to the new flow (below the new Start) an Instruction module *Instruction* (From the Utilities stencil). Double click to edit it , set it's text to: Click to stop
- f. Switch to the Instructions Advanced tab and change the window location to 50,50

Window location (% percentage)

☐ Auto    ☒ Manual    X     Y



- i. Right below the instruction step, add an END step *End* (From the Flow stencil). Double click to edit it: Set it to “Stop this test” ☒ Stop this test

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.



## 5. Assets

### 5.1 tRsrc\_Calc

#### 5.1.1 Description

The test demonstrates the activations and de-activation of an executable asset

#### 5.1.2 Modules Used

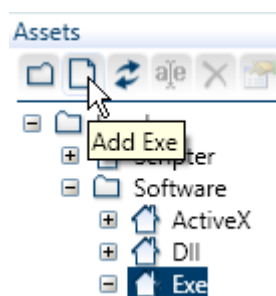
- a. Asset (Executable)
- b. Message

#### 5.1.3 Details

- a. Launch the windows calculator
- b. Display a message “click OK when you’re finished with the calculator”
- c. Once clicked – close the calculator application

#### 5.1.4 Test Details:

- a. Create a test named tRsrc\_Calc in your exercise folder
- b. Add an asset to the Studio environment (from the Assets pane, select Local\Software\exe and click on the Add Exe button)



- 3. Asset Name: Calc
- 4. Path: C:\WINDOWS\system32\calc.exe

- c. Drag the Calc asset from the Assets pane to the test canvas. Double click to edit it – remove the check from the “Wait For Exit” ☐ Wait For Exit

This material is confidential and proprietary.  
Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.



- d. Add a message **Message** (from the Utilities stencil). Double click to edit the step to display:  
Click when you are done with the calculators
- e. Copy the Calc step (CTRL+C) and paste it (CTRL+V) right after the message. Double click the step to edit it. Change the step to KILL ☐ Execute ☒ Kill

This material is confidential and proprietary.  
Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 5.2 tRsrc\_Dll

### 5.2.1 Description

The test demonstrates using a .NET DLL resource and the File System library

### 5.2.2 Modules Used

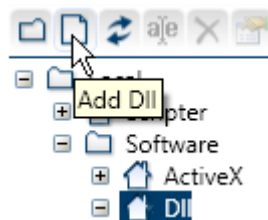
- a. Asset (.NET DLL)
- b. Message

### 5.2.3 Details

- a. Checks if a file (C:\deletme.txt) exist
- b. Read a file content
- c. *Note: create c:\deletme.txt before running the test*



### 5.2.4 Test Details:

- a. Create a test named tRsrc\_Dll in your exercise folder
- b. Create a variable (CTRL+Alt+V)
  - 1. Name: FilePath (String Scalar) Initial value: 'C:\deletme.txt'
  - 2. Name: ReturnValue (Numeric Scalar) Initial value: 0
- c. Add an asset to the Studio environment (from the Assets pane, select Local\Software\Dll and click on the Add Dll button)



- 1. Asset Name: File System
- 2. Path:  
"C:\Program Files\QualiSystems\Studio\Libraries\QualiSystems.API.FileSystem.dll"

This material is confidential and proprietary.  
Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

- d. Drag the File System asset from the Assets pane to the test canvas. Double click to edit it – it opens as a mini-canvas *tRsrc\_Dll.File System Resource*
- e. On the left side pane of the DLL mini-canvas, Appears the DLL content tree, it displays the assemblies, constructs namespaces and methods. Expand the tree's FileSystem branch to view some of the methods.
- f. Select the method from the DLL method tree . The method parameters will appear in the middle pane. In the 'path' field, type {FilePath} (or use CTRL+INS). In the output field (below) 'Return Value', type {ReturnValue} (or use CTRL+INS)
- g. Add another method, 'ReadTextFromFile' and use the {ReturnValue} variable in the path field.
- h. Close the DLL mini-canvas, and switch to the main test
- i. Add a Message step  (from the Utilities stencil), right after the DLL step. Double click to edit the step to display {ReturnValue}
- j. Add another message step to display the text from the file: {Method2.text}

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 5.3 tRsrc\_TCL

### 5.3.1 Description

The test demonstrates using a TCL script as a test asset.

The TCL script file is located in the training kit, named *Traffic\_Generic.tcl*

**NOTE:** one must install a TCL engine on the system in order to use TCL resources. Some possible sources:

<http://www.activestate.com/activetcl/downloads>

<http://sourceforge.net/projects/tcl/>

<http://www.evolane.com/software/etcl/#downloadwin32>

### 5.3.2 Modules Used

- a. Asset (Scripter-TCL)
- b. Transform
- c. Message

### 5.3.3 Details

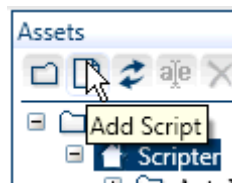
- a. Activate a TCL script to start traffic simulation
  - 1. Set the Traffic Generator address to 192.168.42.201
  - 2. Set the device type to Spirent Test Center
  - 3. Set the result to pass
- b. Display the step result (pass or fail)

### 5.3.4 Test Details:

- a. Creating the resource:
  - 1. Make sure the TCL engine is installed. Make note of the engine executable location (e.g. If using ActiveState 8.4 TCL engine, the location is C:\Tcl\bin\wish84.exe)
  - 2. Save the 'Traffic\_Generic.tcl' to the harddrive (e.g. C:\TestShell\Resources\)
  - 3. Add an asset to the Studio environment (from the Assets pane, select Local\Scripter and click on the Add Script button)

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.



4. The “Scripter Configuration” window appears. Fill out the following fields

- Asset Name: Send\_Traffic
- Properties\Script\Path: Click  to locate Traffic\_Generic.tcl
- Properties\Engine Name: TCL
- Properties\Engine Name: Click , switch to Default engine tab and make sure the engine path is correct. Click Browse to correct if necessary

5. Still on the Scripter Configuration, switch to the “Mapping” tab. Click on the Auto Scan button to map in the changeable fields in the script. Click OK to close the configuration

b. Create a test named tLoop\_2 in your exercise folder

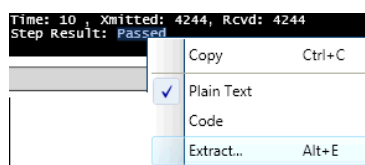
c. Drag the Send\_Traffic asset from the Assets pane to the test canvas. Double click to edit it. At the bottom part of the Scripter panel, fill in the following values:

Name	Pattern	Current Value	Value
host	IP	192.168.42.201	192.168.42.201
traffic	Text	Ixia	Spirent Test Center
Result	Text	Passed	Passed

d. Click , (located above the script area) to get the results. Click OK to close the scripter panel



- e. Add a Transform step *Transform* (from the Analysis and Report stencil). Double click to edit it.
- f. In the Input pane (middle right) scroll down to view the ‘Step Result’ at the bottom of the output.
- g. Highlight the word Passed, right click and select ‘Extract’ from the right click menu.



This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

- h. At the 'Extraction Method' select Before&After. This will add two splits (one before, cutting on the Result: string – and one after, cutting on the carriage return). The output should read Passed. Click OK



- i. Add a Message step **Message** (from the Utilities stencil), right after the Transformation step. Double click to edit the step to display `{Transform1.Measurement1 }`

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## Runner

### 5.4 tRunner\_Report

#### 5.4.1 Description

Creating a report with results and criteria

#### 5.4.2 Modules Used

- a. Loop
- b. Command Shell
- c. Criteria
- d. Case
- e. Message
- f. End (Branch)

#### 5.4.3 Details

- a. Ping three IP addresses (e.g. localhost, 192.168.42.200, 192.168.42.202 – the addresses are arbitrary, choose local IP's in your network)
- b. Check the Lost packets count – pass if 0, fail on any other number
- c. According to the criteria results, pop a message to the user about the ping result (e.g. Ping to 192.168.42.202 has failed!)

#### 5.4.4 Test Details:

- a. Create a test named tRunner\_Report in your exercise folder
- b. Create a variable (CTRL+ALT+V)
  - 1. Name: IP\_Array
  - 2. Type: String; Dimension: Vector; Size: 3
  - 3. Initial values: Localhost, 192.168.42.200, 192.168.42.202

This material is confidential and proprietary.  
Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.



Initial Value		Cur
Size:	3	Set
0	localhost	
1	192.168.42.200	
2	192.168.42.202	



- c. Add a loop structure **Loop** (from the Flow stencil). Click function key F2 to rename it to: *Loop over IPs*. Double click to edit it – Iterations set to 3



Command Shell

- d. Add a Command shell step **Shell** (From the Software stencil). Double click to edit it. It will open in a new mini-canvas document.

- e. Click on **Start Recording**. Click on the **Write** tab to open it, inside it type: ping  
 {IP\_Array(Loop\_over\_IPs.Index)}. Click on the **Transmit** button. This will add a Command step to the mini-canvas. Save and close the mini-canvas



- f. Back at the main test canvas, add a Criteria step **Criteria** (from the Analysis and Report stencil). Double click to edit it

- g. The Criteria Panel opens, Click on the Add button to add a new input. The Variable pane opens – select the Ping Result



- h. At the condition pane (The bottom half of the criteria panel), the condition appears. By default, string inputs are conditioned with the 'Contains' clause. Fill the condition field: Lost = 0. Click OK



This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.



- i. Add a Case structure **Case** (From the Flow stencil) into the loop structure. Double click to edit its conditions. Change the 'Equal' clause to 'On go'. Click OK



- j. Add a message **Message** (from the Utilities stencil) into the left hand branch of the case (#1: On go). Double click to edit the step to display: Ping to {IP\_Array(Loop\_over\_IPs.Index)} passed!
- k. Add another message to the right hand branch of the case structure (Default:). click to edit the step to display: Ping to {IP\_Array(Loop\_over\_IPs.Index)} Failed!



- l. Add an END step **End** (from the Flow stencil), right after the End loop

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 6. Utilities

### 6.1 tUtil\_FileEngine

#### 6.1.1 Description

Creating an MS-Excel file

#### 6.1.2 Modules Used

- a. File Engine

#### 6.1.3 Details

- a. Create an excel file named train\_6a in C:\temp
- b. The file contents – (the curly brackets indicate functions):

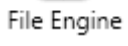
User	{user}
Time	{clock}

- c. Rename the file to deletme.xls
- d. Copy it to C:\
- e. Delete the file C:\temp\deletme.xls


#### 6.1.4 Test Details:

- a. Create a test named tUtil\_FileEngine in your exercise folder



- b. Add a File Engine step  (from the Utilities menu). Double click to edit it. The File Engine panel opens, by default, on the configuration tab. To start working, click on the File Manager button



- c. On the File Management Table, click on the Add New  button, to open the File Management Editor. By Default, the editor shows the Create command tab. Set the parameters:

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.


1. Type: XLS    **Type**
2. Parent Directory: Click Browse to select C:\Temp – and click on “Current Folder”
3. Name: train\_6a
4. Sheet Index: 1

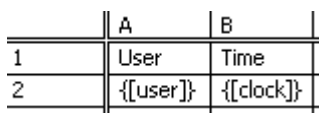
Click on the File Content Edit button 

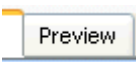
- d. The File Content Editor opens. By default it starts on the Editor tab. Type in cell A1 the word *User*



and in B1 the word *Time*

	A	B
1	User	Time


- e. To enter an expression, Mark the cell A2 and click on the “From Variable” button . The variable pane appears: type on the bottom field: {user} and click OK

- f. Repeat the above step with cell B2 and the expression {clock} . To view
- |   |          |           |
|---|----------|-----------|
|   | A        | B         |
| 1 | User     | Time      |
| 2 | {[user]} | {[clock]} |

the actual contents, click on the Preview tab . Once finished, click OK to close the File Content Editor

- g. Back on the File Management Editor, Click Add . This will add the create command to the File management Table (You might need to pan the Editor to see it). Click on the Rename command button 


- h. The file Management Editor displays the Rename command tab. Click on the Browse button select the File C:\temp\train\_6a.xls. In the New Name field, type deletme.xls. Click Add

- i. Click on the Copy command button , the editor will display the Copy pane. Click on the Browse... button near the From Path field to select the C:\temp directory.

NOTE: to confirm selection, Click on ‘Current Folder’  button and not ‘Save’ or ‘Open’

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

- j. Click on the lower Browse... (The one next to the 'To Parent Directory' field) and navigate to C:\. Again, remember to confirm with 'Current Folder'. In the Name field type: deletme.xls. Click on ADD to add the step to the Table
- k. Back at the File Management Editor, Click on the Delete command button . The editor displays the delete pane. Click on the Browse... button to select C:\temp\train\_6a.xls. Click Add and Close (as this is the last action)
- l. On the File Engine panel, Click OK (or Activate, to see the module in action – remember to delete C:\temp\deletme.xls manually after each run!)

This material is confidential and proprietary.  
Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 6.2 tUtil\_MatShell

### 6.2.1 Description

Mathematics with MatShell!

### 6.2.2 Modules Used

- a. MatShell
- b. Case
- c. Message
- d. End (Branch)

### 6.2.3 Details

- a. Calculate a random number between 2 and 100
- b. Find all integer dividers of that number (e.g. if the number is 60, it's dividers are 1, 2, 3, 4, 5, 6, 10, 15, 20, 30)
- c. If the number of dividers is 1 – pop a message *The Number XX is a prime!*
- d. On any other case, display the number and its list of dividers

### 6.2.4 Test Details:

- a. Create a test named tUtil\_MatShell in your exercise folder
- b. Create the following Variables:
  - 1. Number (Numeric Scalar)  
Description: The random number (between 2 and 100)
  - 2. n (numeric Scalar)  
Description: Used to scan the numbers from 1 to the calculated Number (minus 1)
  - 3. Counter (Numeric Scalar)  
Description: This variable indexes the Divider vector, it will increment by 1 each time a divider is found

This material is confidential and proprietary.  
Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 4. Divider (Numeric Vector)

Description: The Divider vector holds the found dividers of {Number}. If there is only one member (The number 1) - this means that Number is a prime



c. Add a MatShell step **MatShell** (from the Utilities stencil). Double click to edit it:

1. Type `Number=rand`. Click the Activate button and hover with the mouse cursor over the word `Number` – The tooltip will display the variable details and its value. Click on activate again and hover to see the change in value (a random between 0 and 1)
2. Change the script to be `Number=rand*100` – click activate and check the `Number` value
3. Change the script again to `Number=round(rand*98)+2` – this creates an integer between 2 and 100
4. Add to the script the lines (each at a separate line)
  - i. `for (n=1:(Number-1))` `/* this will iterate between this line and end */`
  - ii. `if (mod(Number,n)==0)` `/* mod returns the remainder of Number/n */`
  - iii. `Divider(Counter++)=n` `/* Counter++ means Counter=Counter+1 */`
  - iv. `end` `/* this ends the if clause */`
  - v. `end` `/* this ends the for loop */`

d. Add a case structure to the canvas (from the Flow stencil) edit it and set the condition according to the length of the Divider vector



- e. Add a message **Message** (from the Utilities stencil) into the left branch of the case (#1 `length(Divider) == 1`). Double click to edit the step to display: The number {Number} is a Prime!
- f. Add another message to the right branch of the case (Default:). Double click to edit the step to display: The number: {Number}  
The dividers: {vect2str(Divider, ' ')}

g. After the End Case step, add an END step **End** (from the Flow stencil)

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 6.3 tUtil\_WriteToFile

### 6.3.1 Description

File interaction

### 6.3.2 Modules Used

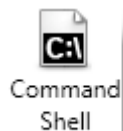
- a. Command Shell
- b. File Engine
- c. Transform
- d. End (Branch)

### 6.3.3 Details

- a. Execute an IPCONFIG command on your system, save the output to a file C:\Temp\test\_1
- b. Load the file into a variable, parse the IP address from it
- c. End

### 6.3.4 Test Details:

- a. Create a test named tUtil\_WriteToFile in your exercise folder



- k. Add a command shell tool (from the Software stencil). Double click the command shell tool, a mini-canvas opens as a new document
- b. On the Command Shell mini-canvas (tUtil\_WriteToFile.Command Shell1), click on Start Recording button. On the Terminal window type ipconfig (and click enter). The Command1 step was added to the mini-canvas, and a Module Variable (Command1.Result) was added to the variable pane



This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.



- c. Save and close the Command Shell mini-canvas. Switch to the main test



- d. Add a File Engine step **File Engine** (from the Utilities menu). Double click to edit it. Click on the File Manager button

- e. On the File Management Table, click on the Add New button, to open the File Management Editor. Make sure the Create button is pressed. Set the parameters:

1. Type: txt **Type**
2. Parent Directory: click browse to select C:\Temp (Remember: to confirm click on Current Directory – NOT save)
3. Name: test\_1

Click on the File Content Edit button



- f. On the File Content Editor, Right click inside the editor pane and select Insert Variable. Insert the Command1.Result variable. To confirm switch to the Preview tab . Click OK to go back to the File Management editor. Click Add and close. Back at the File Engine table
- g. In order to read the file, we must create it first. Click Activate. The output display should read: *1 File(s) created.*

- h. Click on the Add New button, to open the File Management Editor. Select the Load command . Set the parameters:

1. Path: Click on the Browse... button to select C:\Temp\test\_1.txt. Click Add and Close

- i. On the file engine, click OK and return to the main test canvas



- j. From the Analysis stencil add a Transform step **Transform**. Double click to edit it. Click on the Add



Source button on the Source pane (top left) . The variable pane appears: Select

File\_Engine1.Load\_Content\_0

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

k. To extract the IP address, add the following transformation steps:

1. On the Input pane (middle right), Highlight the words *IP Address*, right click and select Extract... From the Extraction Method menu, select Split After and click Extract. A new transformation is added to the list on the left.
2. On the Input pane (middle right), Highlight the words *Subnet Mask*, right click and select Extract... From the Extraction Method menu, select Split Before and click Extract.
3. On the Input pane (middle right), Highlight the IP Address itself (e.g. 192.168.42.52), right click and select Extract... From the Extraction Method menu select Regular Expression and change the Pattern to IP

Extraction Method  
Regular Expression ▼

Pattern  
IP ▼

1. Click OK



- m. Back on the main test canvas, add an END step **End** (from the Flow stencil)

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 7. Variables

### 7.1 tVar\_Expressions

#### 7.1.1 Description

Using expressions and variables in various fields. Compare this to exercise tUtil\_MatShell

#### 7.1.2 Modules Used

- a. Assign Variable
- b. Loop
- c. Case
- d. Message
- e. End (Branch)

#### 7.1.3 Details

- a. Calculate an integer random number, between 0 and 100
- b. Scan the numbers from 1 to 9, and note any number that divides the calculated number without a remainder
- c. Display to the user the number and it's dividers

#### 7.1.4 Test Details:

- a. Create a test named tVar\_Expressions in your exercise folder
- b. Create the following variables (CTRL+ALT+V)
  - 1. Name: Data (Numeric Scalar) Description: The calculated random number
  - 2. Name: Array (Numeric Vector; Initial values [1,2,3,4,5,6,7,8,9]) Description: The list of potential dividers
  - 3. Name: Message (String Scalar) Description: The concatenated list of dividers, ready for display as text

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

- c. Drag the {Data} variable from the variable pane to the canvas (this will create an Assign Variable step, with Data as a target). Double click to edit it. Set the module to 'From Expression'

☐ From Value ☒ From Expression and the expression to: `{round(rand * 100)}`



- d. Add a loop structure **Loop** (from the Flow stencil). Double click to edit it – Iterations set to `{length(Array)}` - this will iterate as many times as there are items in the list – changing the list size will change the number of iterations
- e. Add a case structure to the canvas (from the Flow stencil) edit it and set the condition according to the mod function (it returns the remainder of a division between two numbers) – if the mod of {Data} and {Array(Loop1.Index)} is zero, there is no remainder and {Array(Loop1.Index)} is a valid divider



- f. Drag the {Message} variable from the variable pane to the canvas (this will create an Assign Variable step, with Message as a target). Double click to edit it. Set the module to 'From Expression' ☐ From Value ☒ From Expression and the expression to: `{strcat(Message, ',num2str(Array(Loop1.Index))')}`. This will concatenate the newfound divider (turned into a string with the number to string function to the previous list (if there was any).



- g. Add a message **Message** (from the Utilities stencil) right after the End Loop. Double click to edit the step to display: The number {Data} is divided by {Message}

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 7.2 fVar\_Function

### 7.2.1 Description

This exercise is part of the tVar\_FunctionCall exercise. One must finish this exercise before starting tVar\_FunctionCall.

The exercise will find the dividers of a given number. The number is set from a variable which is published as input (i.e. can be set from another test). Then the test (acting as a function) will publish out the list of dividers.

### 7.2.2 Modules Used

- a. Loop
- b. Case
- c. Assign Variable
- d. End

### 7.2.3 Details

- a. Create an input node to the test (a numeric scalar input)
- b. Find all the dividers (e.g. if the number is 30, the divider list is [1,2,3,5,6,10,15,30])
- c. Publish the divider list as an output (a numeric vector)

### 7.2.4 Test Details:

- a. Create a test named fVar\_Function in your exercise folder
- b. Create the following variables:
  - 1. Name: InputNum (Numeric, Scalar, published input, Initial value: 30) Description: The number from the caller test.
  - 2. Name: Counter (Numeric scalar) Description: The index for the Divider List
  - 3. Name: DividerList (Numeric, Vector, published output, Initial value size: 0) Description: The input number's divisors list (Numeric Vector).

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.



- c. Add a loop structure **Loop** (from the Flow stencil). Double click to edit it – Iterations set to `{InputNum + 1}`
- d. Add a case structure (from the Flow stencil) inside the loop. edit it and set the condition according to the mod function (it returns the remainder of a division between two numbers) – if the mod of `{InputNum}` and `{Loop1.Index}` is zero, there is no reminder and `{Loop1.Index}` is a valid divider



- e. Drag into the case structure, under branch #1, the DividerList variable from the variable pane. This will create an Assign Variable step (with the DividerList as the target). Double click to edit it. Set the module to 'From Expression' ☐ From Value ☒ From Expression and the expression to: `{Loop1.Index}`
- f. Right under the Assign Variable step (DividerList1), drag the Counter variable from the variable pane to create an Assign Variable step (with Counter as the target). Double click to edit, and Set the module to 'From Expression' ☐ From Value ☒ From Expression and the expression to: `{Counter + 1}`



- g. Right after the End Loop step, add an END step **End** (from the Flow stencil)

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

## 7.3 tVar\_FunctionCall

### 7.3.1 Description

A previous exercise is required to fulfil this one. Please complete exercise fVar\_Function before working on this exercise.

This test will demonstrate the use of function tests

### 7.3.2 Modules Used

- a. Instruction
- b. Function Test
- c. Message
- d. End (Branch)

### 7.3.3 Details

- a. Get from the user a positive integer number
- b. Use the test fVar\_Function as a module to get the dividers of the number entered
- c. Display the entered number and the divisors in a message to the user

### 7.3.4 Test Details:

- a. Create a test named tVar\_FunctionCall in your exercise folder



- b. Add an Instruction step **Instruction** to the canvas (from the Utilities stencil). Edit the Instruction (double click), set the text to be “Please enter a whole positive number” and add a user input:

Type:	textbox
Label:	Number
Default Value:	30
Output Name:	Number
Output Type:	Numeric
Output Dimension:	Scalar

This material is confidential and proprietary.  
Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

Click OK.

- c. Back at the canvas, drag the test: fVar\_Function from the test pane, right below the instruction step.

This will create a test step. Double click to edit it.

In the Data Source field, right click and select Insert Variable. Select Instruction1.Number.

## Inputs

#	Name	Type	Dimension	Data Source	Description
1	Input Number	Numeric	Scalar	{Instruction1.Number}	An integer (Numeric Scalar)

Click OK to close and go back to the canvas



- d. Add a message **Message** (from the Utilities stencil) right after the Instruction. Double click to edit the step to display:

The divisors list of {Instruction1.Number} returned by the function is:

```
{vect2str(fVar_Function1.Divisor_Vector, '')}
```



- e. Add an END step **End** (from the Flow stencil)

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.



## 7.4 tVar\_Sessions

### 7.4.1 Description

A simple test to demonstrate working with multiple sessions variables

### 7.4.2 Modules Used

- a. Command Shell

### 7.4.3 Details

- a. Create two sessions of Command Shell ('First' and 'Second')
- b. Set one of the sessions ('First') directory to C:\Windows
- c. Set the other session ('Second') directory to C:\Temp
- d. Open again the 'First' session – what will be the directory shown?

### 7.4.4 Test Details:

- a. Create a test named tVar\_Sessions in your exercise folder
- b. Create two variables (CTRL+ALT+V)
  - 1. Name: *First* (Session Scalar)
  - 2. Name: *Second* (Session Scalar)



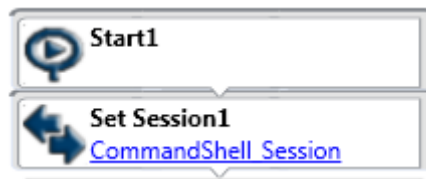
- c. Click on the View button in the Ribbon [View](#). Mark the Active Sessions checkmark. This will turn the Active Sessions pane visible.



- d. Add a command shell tool (from the Software stencil)
- e. Double click the command shell tool, a mini-canvas opens as a new document (Notice that the Start and Set Session steps are automatically added). A session variable named CommandShell\_Session is created automatically. We'll replace it with the variable we created

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.



- f. On the terminal pane, highlight the {CommandShell\_Session} variable in the Session Name field. Right click and select Insert Variable. Insert the variable *First*. Session Name: {First}. Click on the Start Session Button Start Session (wait for the session to start – the Start Session button will turn to Reset)
- g. Click on Start Recording button. In the terminal, type the command *cd windows* (and click enter). This will add a Command step to the mini canvas. The path in the terminal should now be: PS C:\WINDOWS>
- h. Close the mini-canvas and switch back to the main test. Watch the Active Session pane. It will display all open sessions (at least two of them: CommandShell\_Session and First)
- i. Add another Command Shell step. Double click to edit its mini-canvas.
- j. Change the Session Name to {Second}. Click on Start Session, and wait for the panel to connect. Once connected click on Start Recording
- k. In the terminal pane type *cd temp*. Notice the step added to the mini-canvas and the change in the command prompt in the terminal
- l. Close the mini-canvas. Switch back to the main test – watch the Active Session pane, it should display (at least) three sessions you've created
- m. Add another Command Shell step. Double click it to view its mini-canvas. Watch the command prompt in the terminal – it should be PS C:\>
- n. Change the Session Name to {First} (or use ALT+INS to insert it from the variable pane). Click on the Start Session button. Once the session reconnects, the command prompt should be PS C:\WINDOWS> . Which means, the panel connected to the *First* active session.
- o. In the Active Session Pane, you can right click on the CommandShell\_Session and select to close the session

This material is confidential and proprietary.

Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

This material is confidential and proprietary.  
Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.

This material is confidential and proprietary.  
Any unauthorized reproduction, use or disclosure of this material or any part thereof, is strictly forbidden.