Deltek

Deltek Test Runner User and Keyword Guide

Version: 2.2.5

Table of Contents

Introduction

The scope of this document is to guide the tester in using the *Test Runner* tool in performing different test automation tasks. Information on controls recorded and available keywords for scripting is outside the scope of this document.

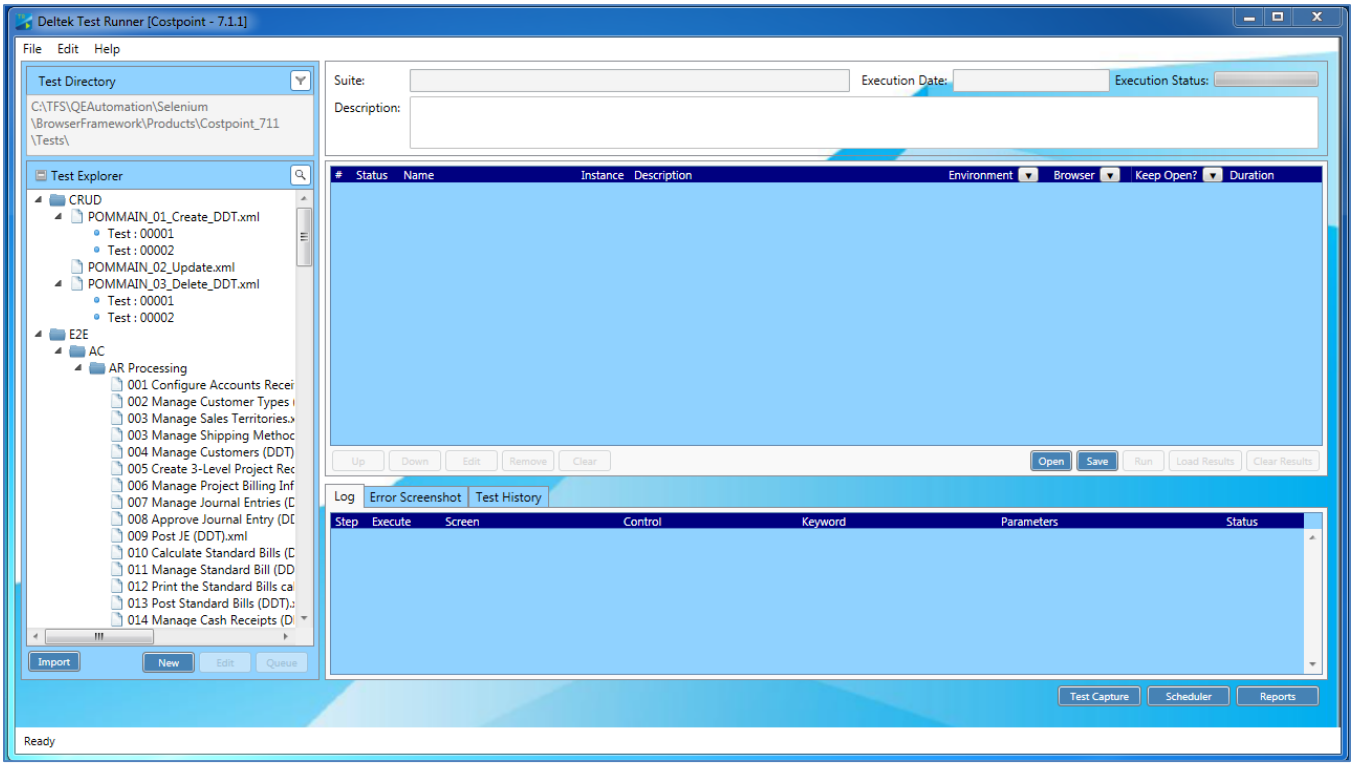


Figure 1: Test Runner

Test Runner is the central piece of *Deltek Automation Framework*. It serves as a single point of contact for all tasks in relation with automating tests for *Costpoint*. This document will provide a comprehensive guide and walkthroughs on all common tasks that a tester will be performing for any test automation project.

Below are the topics to be discussed in the document:

- Getting Started – Preparation of a test machine for *Test Runner*
- Creating and debugging a test script using *Test Editor*
- Queueing test scripts in a *test suite*
- Executing a *test suite*
- Viewing *test suite* results
- Scheduling *test suites* executions
- Viewing of execution reports
- Recording a test using *Test Capture*

Getting Started

System Requirements

Test Runner can be run on client machines that meet the following minimum requirements:

Operating System	Microsoft Windows XP (Service Pack 3) Microsoft Windows Vista Microsoft Windows 7 Microsoft Windows 8.1 Microsoft Windows 10
Memory	2.00 GB RAM
Disk space	100 MB
Required Software	Microsoft .NET Framework 4.5 Mozilla Firefox [Optional] Google Chrome [Optional]

Initial Setup

To start using the *Test Runner* tool, double-click *TestRunner.exe*. A dialog with choices on what application to test will be displayed.

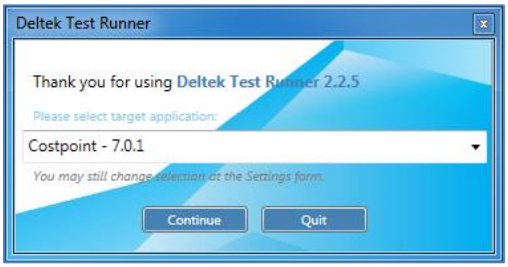


Figure 2: Selecting target application on initial launch

Test Runner will remember your chosen application, and will load *Test Runner* targeting the selected application on succeeding runs.

The option to change applications can also be found under the *Preferences* tab in the *Settings* menu. Changing the target application will load the necessary files and settings needed for that application.

After choosing an application to test, a dialog box containing information about new features of the current version will be displayed.

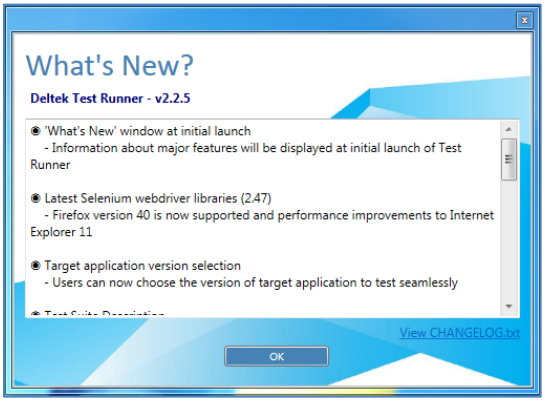


Figure 3: 'What's New' dialog box

Browser Settings

The *Deltek Automation Framework* utilizes *Selenium WebDriver* for automating web applications. While *Selenium WebDriver* supports most web browsers, some browsers have required configurations for *Selenium* to function properly. Before proceeding in any automation task, please ensure that the web browser used for testing has the following configurations.

Internet Explorer (IE)

For IE7 and up, the *Enable Protected Mode* settings should be the same for all security zones. The value can be on or off but all zones should have the same value. Below are the steps in setting up the *Enable Protected Mode*:

1. From the Internet Explorer main window, go to *Internet Options.*

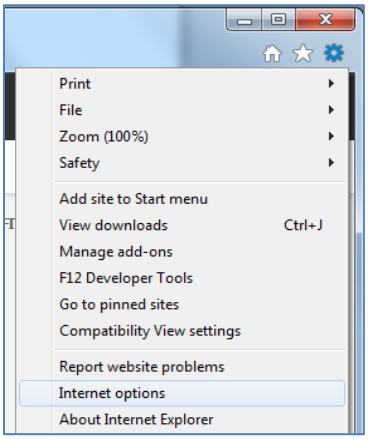


Figure 4: IE Internet Options

2. Go to the *Security* tab.

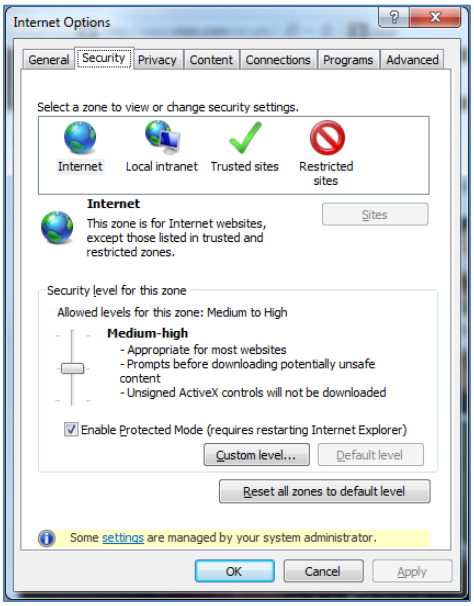


Figure 5: IE Internet Options Security tab

3. Select the *Internet* zone by clicking it.

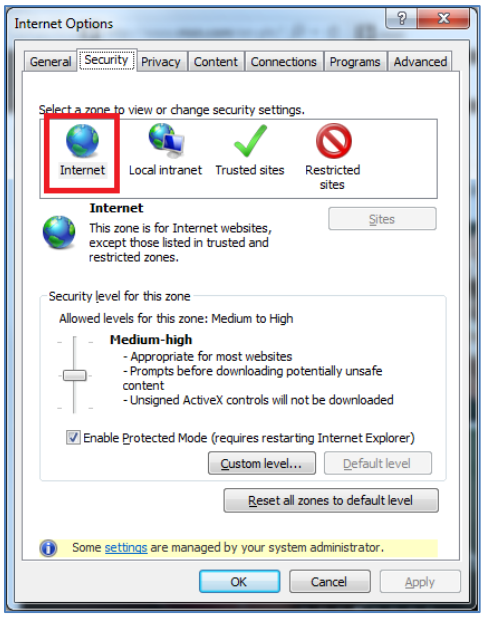


Figure 6: IE Internet Options Security Zones

4. Either turn on or turn off the *Enable Protected Mode* option.

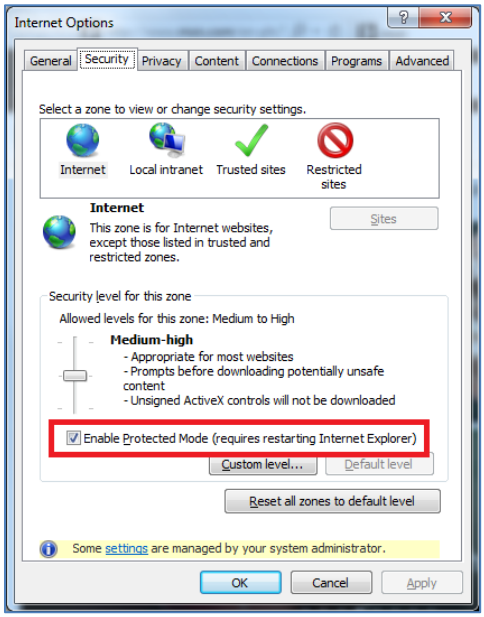


Figure 7: IE Enable Protected Mode

Repeat the same steps for all remaining zones (*Local intranet, Trusted sites, Restricted sites*), and that the values of *Enable Protected Mode* are the same value.

In addition to the *Enable Protected Mode*, it is important to set the browser zoom level to 100%. This would ensure the accuracy of Selenium's click method.

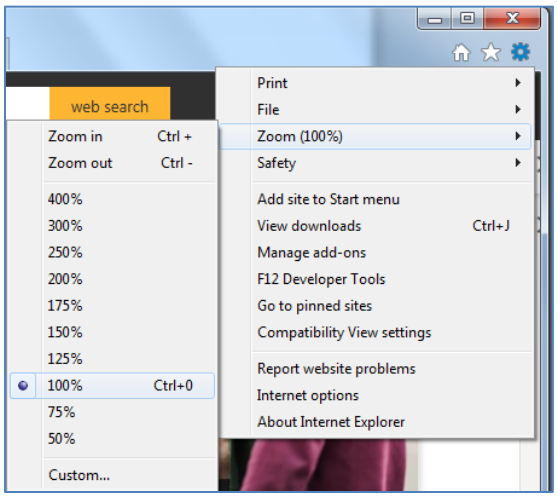


Figure 8: Browser zoom level

Environment

Before creating and running a test script, we need to setup *Environments* in *Test Runner*. An *Environment* contains information on where the *Test Runner* would execute the test scripts against, such as application URL, and default login (username, password, database). *Environment* settings can be accessed by going to *Edit->Settings* from the *Test Runner* menu.

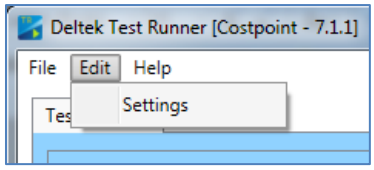


Figure 9: Test Runner Edit->Settings menu

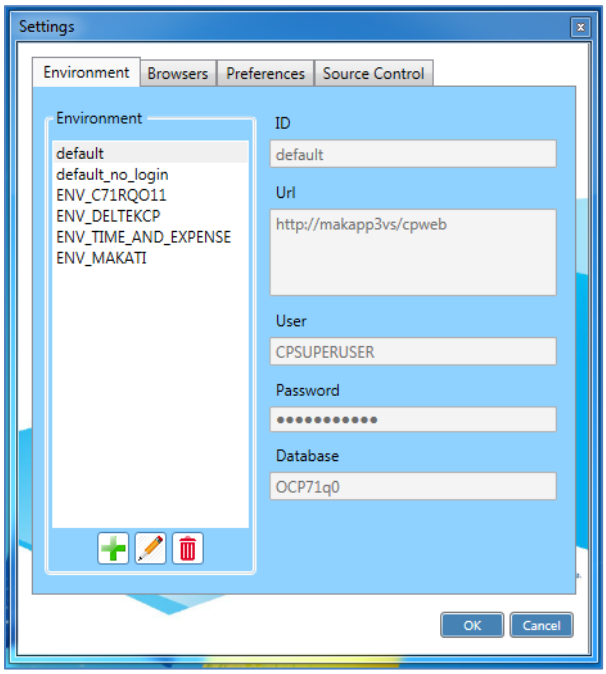


Figure 10: Environment settings

Adding new environment

1. Click the  button to create a new environment record.

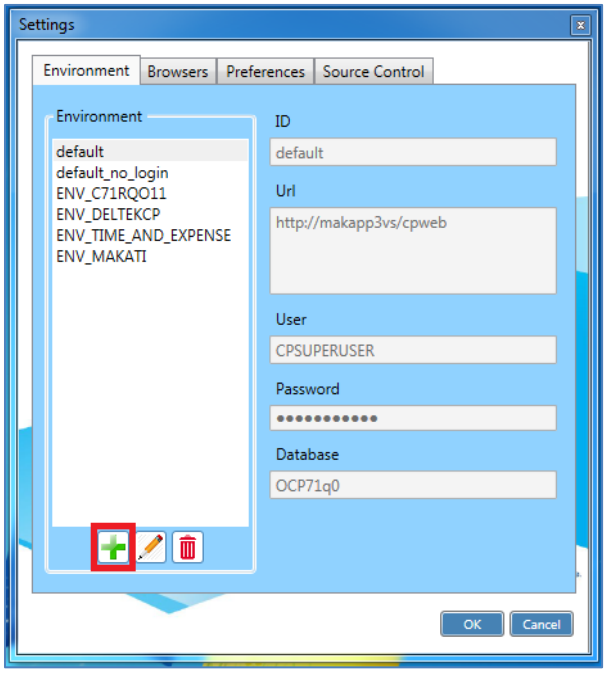


Figure 11: Add new environment

2. Enter the *ID* to identify the new environment.

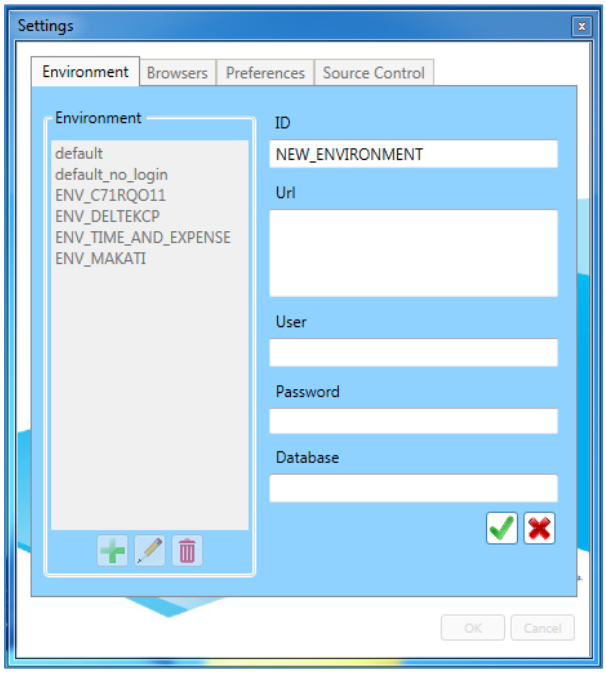


Figure 12: Environment ID

3. Enter the application URL in the *Url* field.

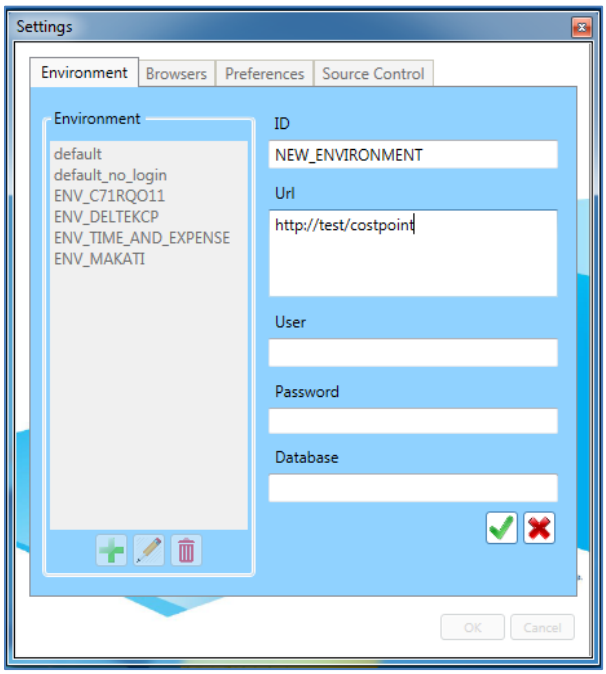


Figure 13: Environment Url

4. Enter the *User*, *Password*, and *Database* fields to be used *Costpoint* login [Optional*]

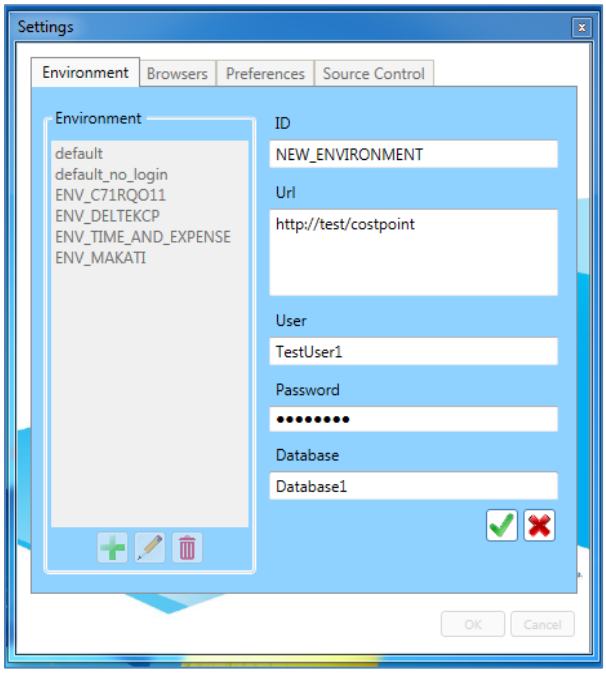


Figure 14: User, Password, and Database

** User, Password, and Database fields are optional. If these fields are set, the Test Runner will perform a login prior to executing the first step of the test. If these fields are left blank, steps that will log into Costpoint should be included in the test.*

5. Click the  to save the new Environment record.

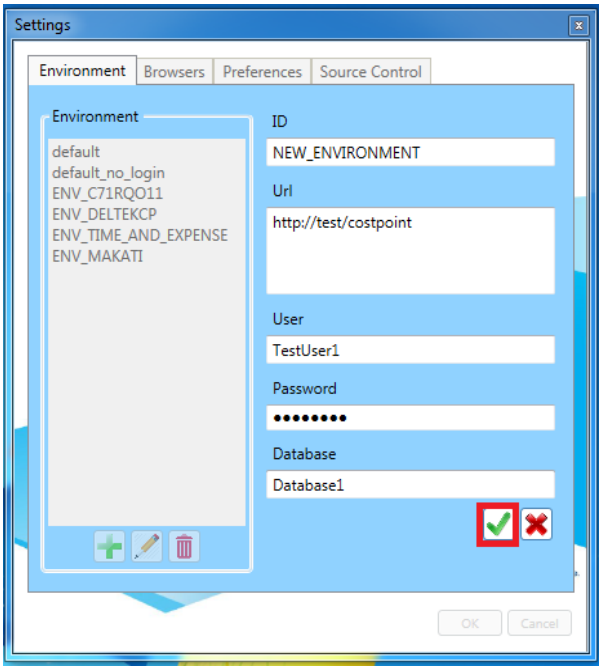


Figure 15: Save environment

6. Click *OK* to close the Settings form.

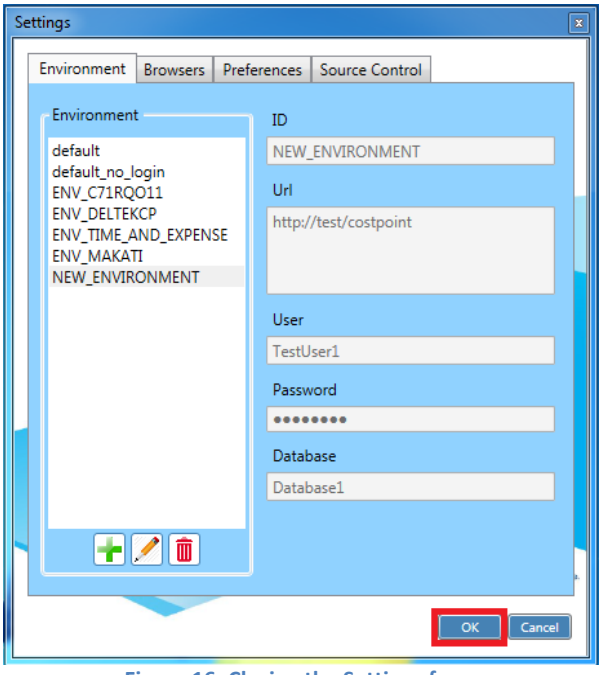


Figure 16: Closing the Settings form

Editing an existing environment

1. Select an *Environment* record and click the  button to edit the record.

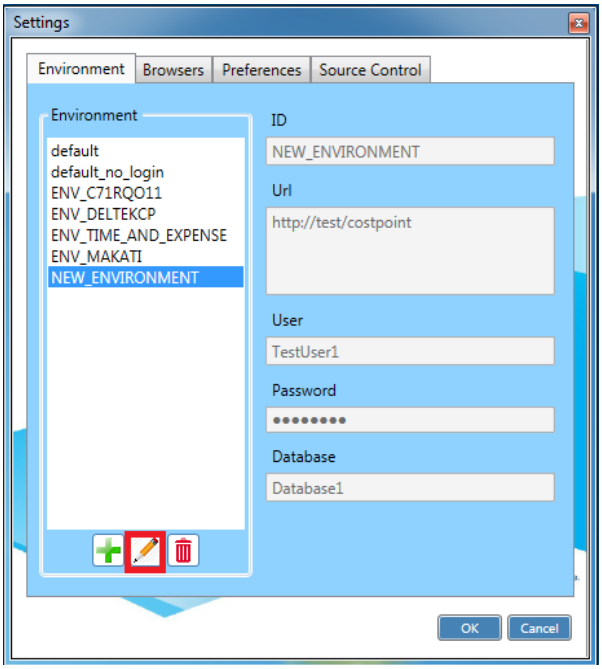


Figure 17: Edit existing environment

2. All of the fields will be enabled. Modify the desired fields.

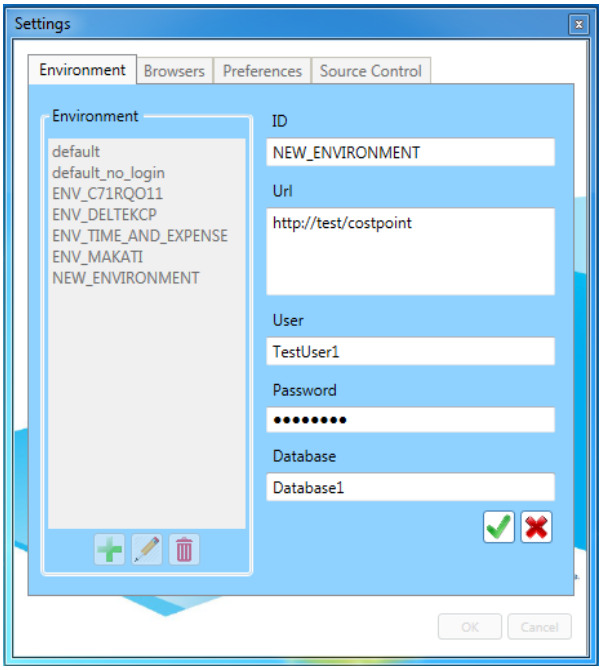


Figure 18: Modify environment fields

Deleting an existing environment

1. Select an Environment record and click the  button to delete the record

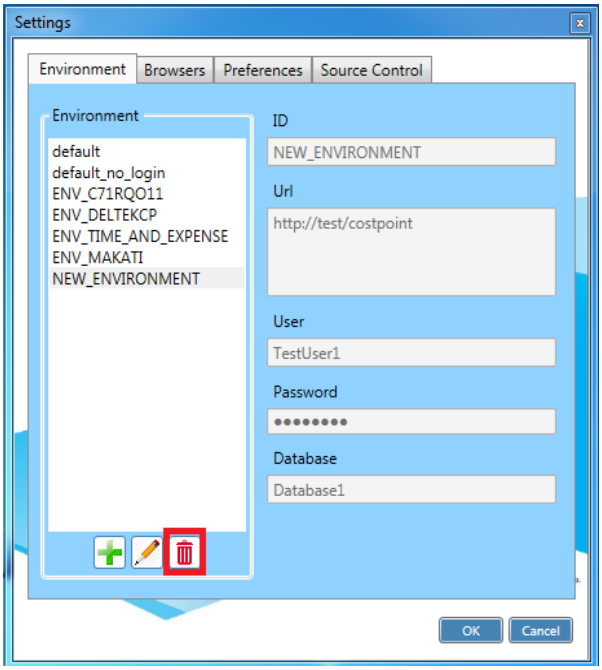


Figure 19: Delete environment

2. Click *Yes* to confirm the deletion, and then click *OK* to close the Settings dialog.

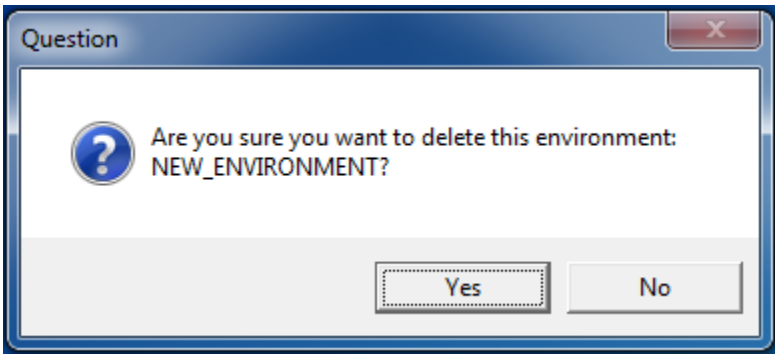


Figure 20: Delete confirmation

Browsers

The Browsers tab contains information that displays the version and notes on the browsers installed in your test machine. Compatibility and issues that may arise on different versions of each browser are detailed in the Notes section.

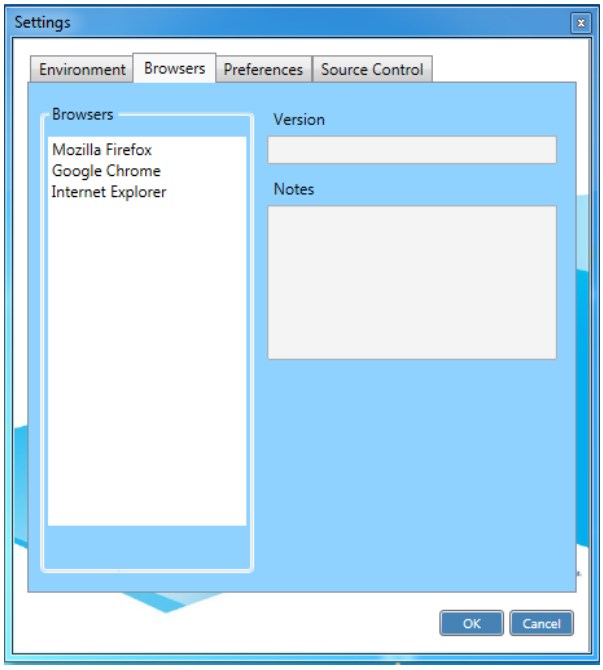


Figure 21: Browser

Preferences

The *Preferences* tab contains the *Target Application* section where the application to test can be changed, and the *Error Log Level* section where you can change the level of error logs displayed after a test is run. The following are the error log levels and their corresponding descriptions.

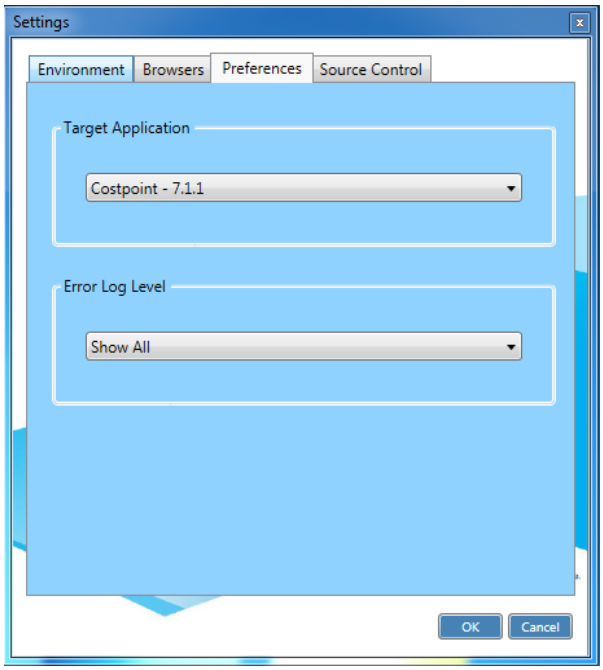


Figure 22: Error Log Levels in Preferences Tab

Show All – The default setting. Displays all the information contained in error logs.
Show Basic Error Info – Displays the error message, basic information, and error screenshot.
Show Error Only – Displays the error message and error screenshot only.

Creating a Test Script: The Test Editor

Opening the Test Editor

Click the *New* or *Edit* button under the *Test Explorer* tab to display the *Test Editor*.

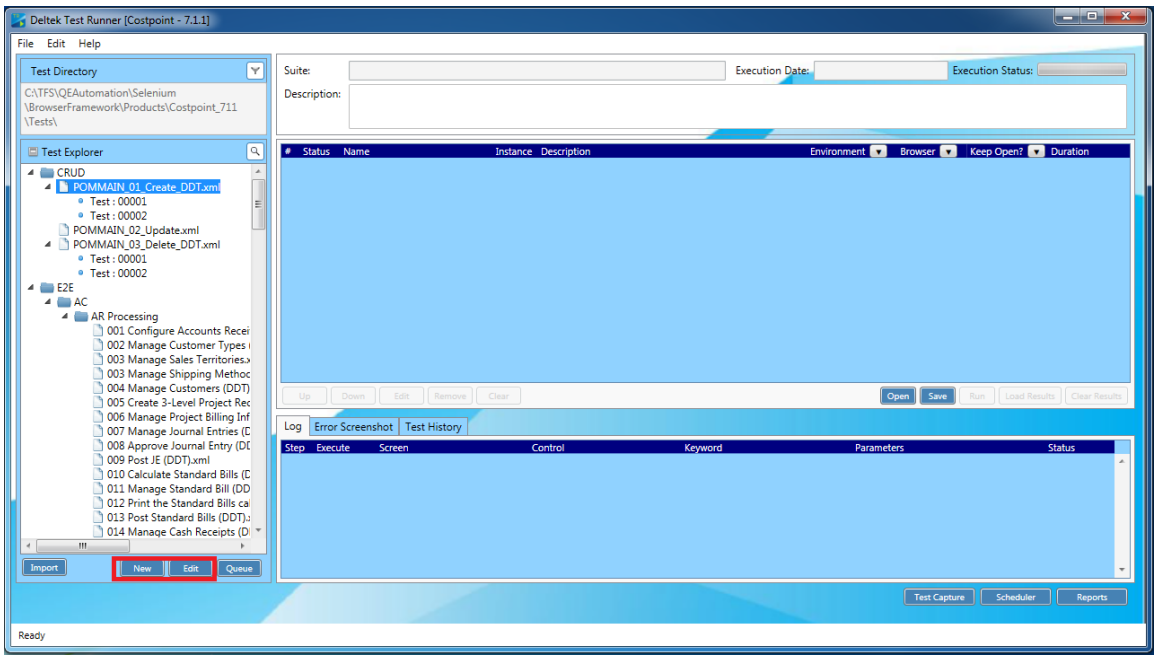


Figure 23: Opening the Test Editor

New Test

A new test will be available for edit on the *Test Editor* when the *New* button is clicked.

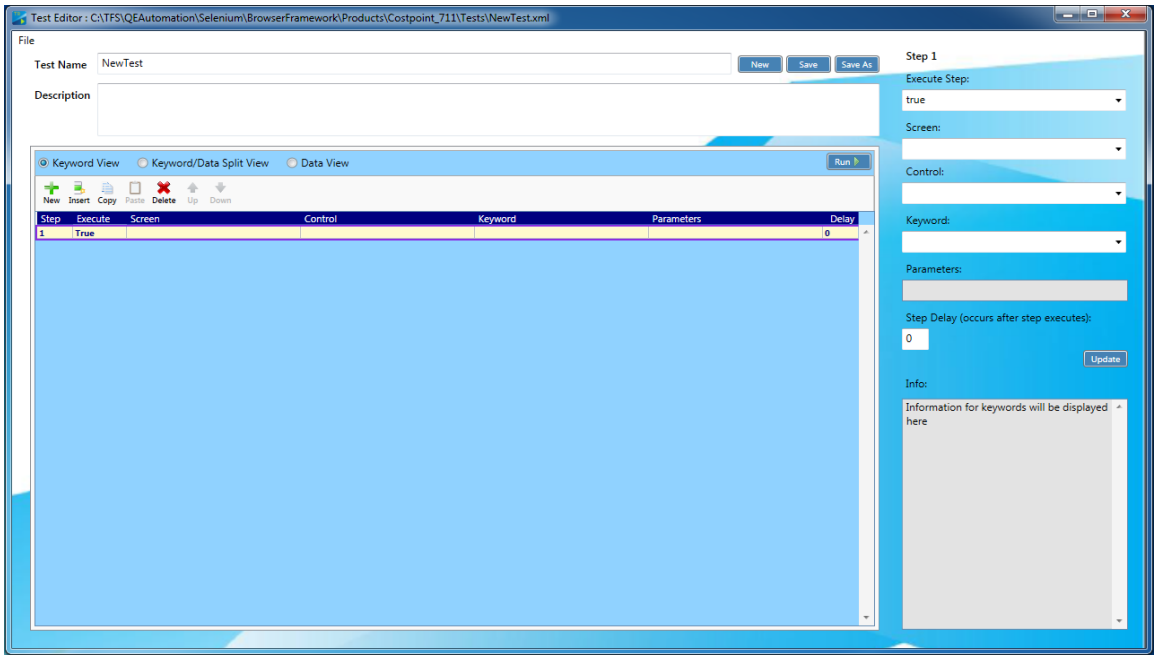


Figure 24: New Test

Import Test

The *Import* button allows the user to import a test from another location. When the *Import* button is clicked, a new file dialog will be displayed to prompt the user to select the test script to import.

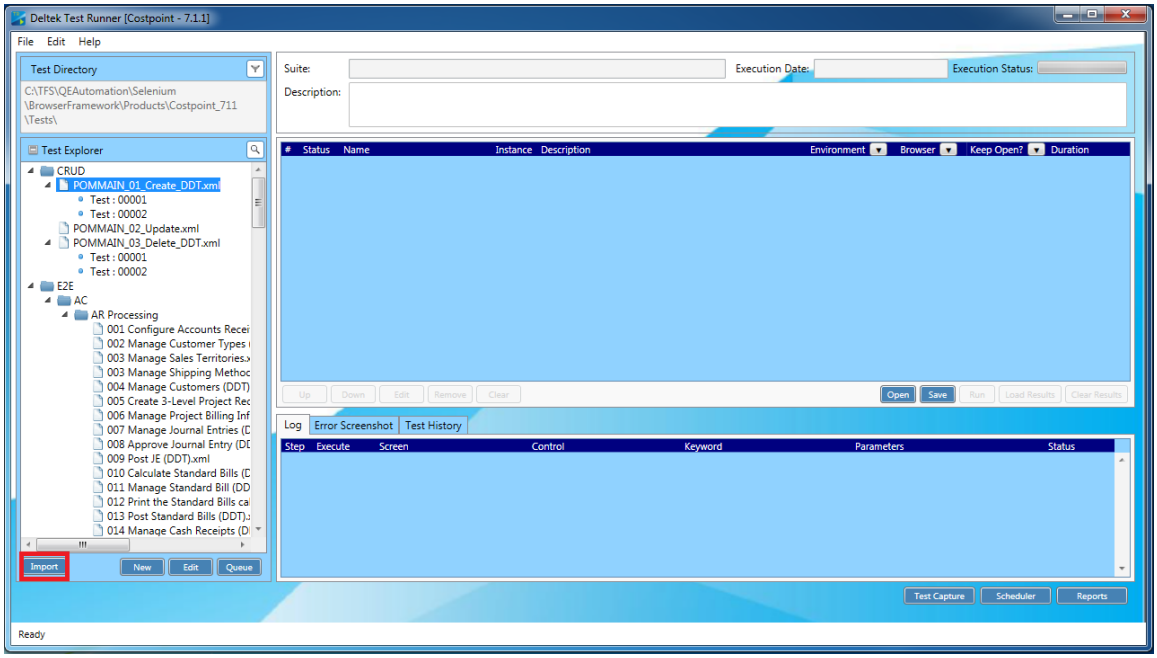


Figure 25: Import Test

Edit Test

When a test is selected in the *Test Explorer* tab in the *Test Runner* and the *Edit* button is clicked, the test will be displayed in the *Test Editor* for edit.

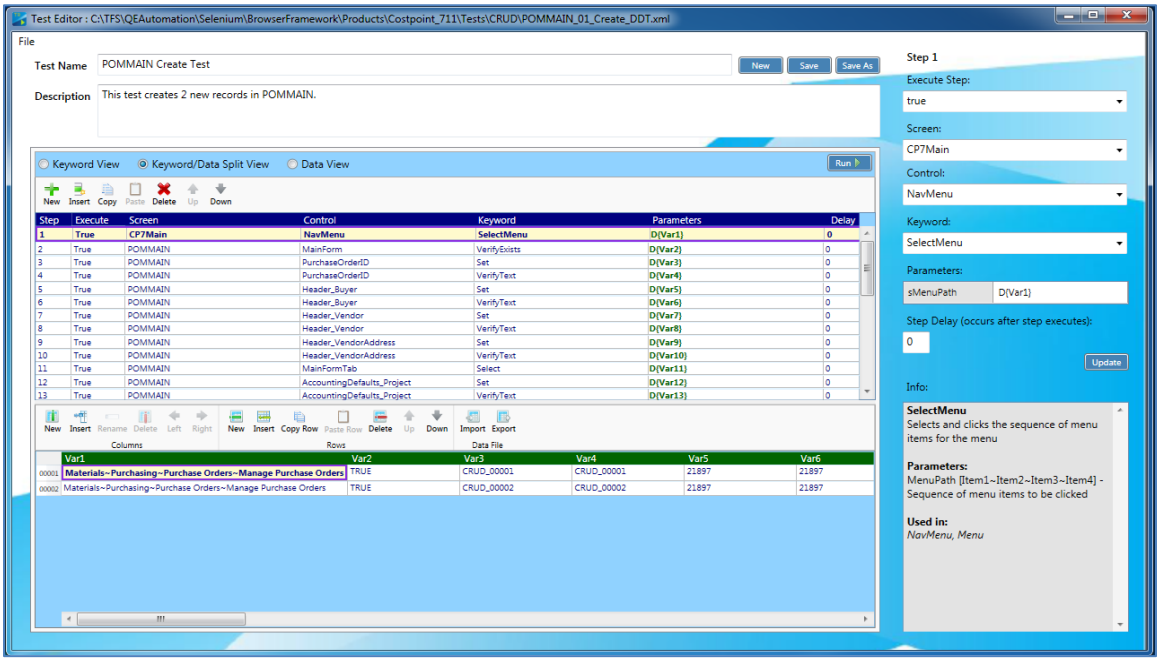


Figure 26: Edit Test

Test Name and Test Description

Test Name is used to identify the test while *Description* is used for any other information the user would like to preserve with the test. This can be a brief summary of the test or any other additional information regarding the test.

Saving Tests

Click *Save* or *Save As* buttons to save your tests. Tests are saved as XML files (.xml).

Keyword View

When the *Test Editor* is launched, the *Keyword View* is selected. This view lists all the steps of a test.

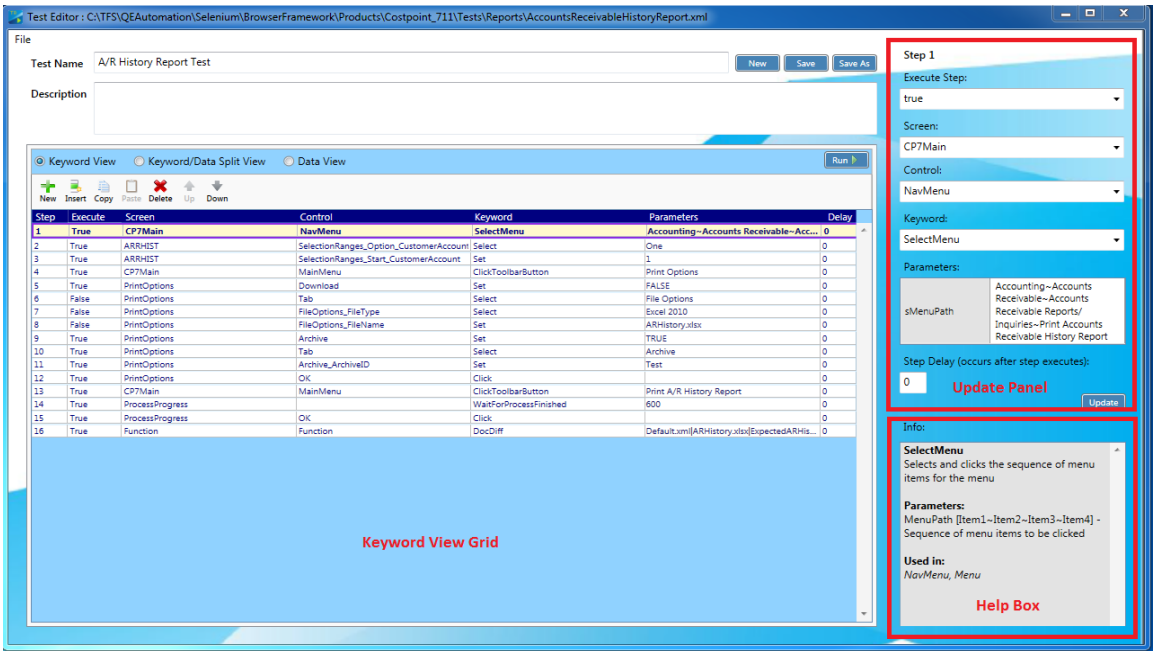


Figure 27: Keyword View

On the right-hand side of the *Test Editor* is the *Update Panel*. This is used to edit values of the selected test step in the *Keyword View* grid. The *Update Panel* fields are described below

Execute Step

This value is used to mark if the test step will be executed or skipped. Possible values are TRUE (will be executed), or FALSE (will be skipped).

Screen

This value is used to identify the screen the test will interact with. For example, the *CP7Login* screen represents the login page for the *Costpoint 7* application

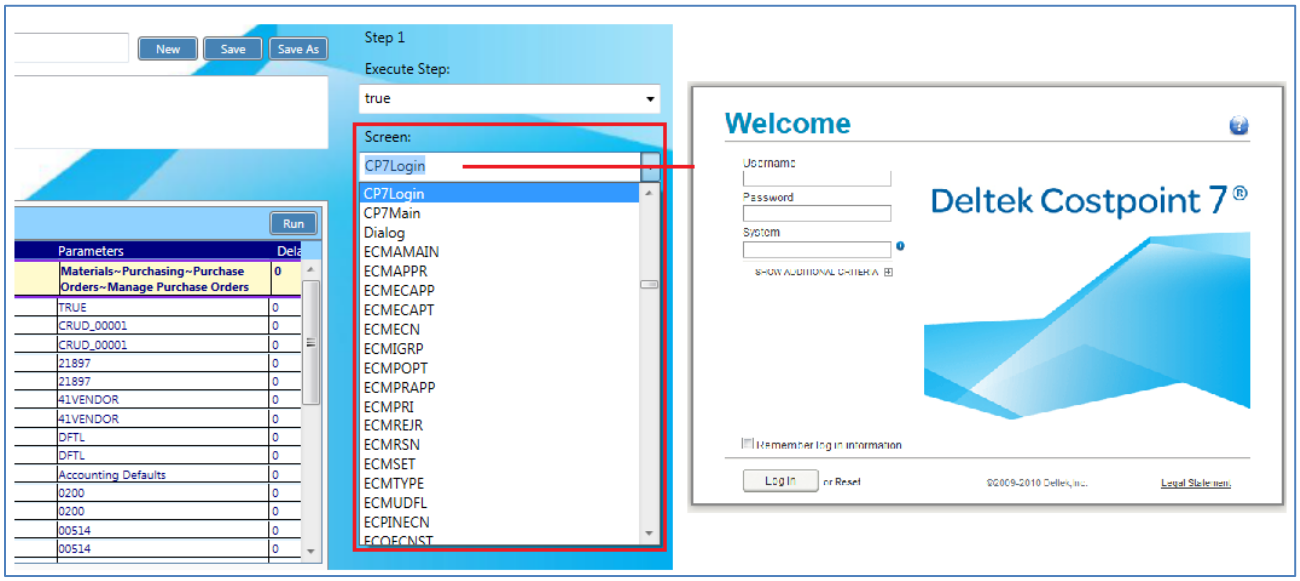


Figure 28: Screen

Control

This is a list of all possible controls within the selected screen that the test can interact with. Shown below are the control values available for the *CP7Login* screen:

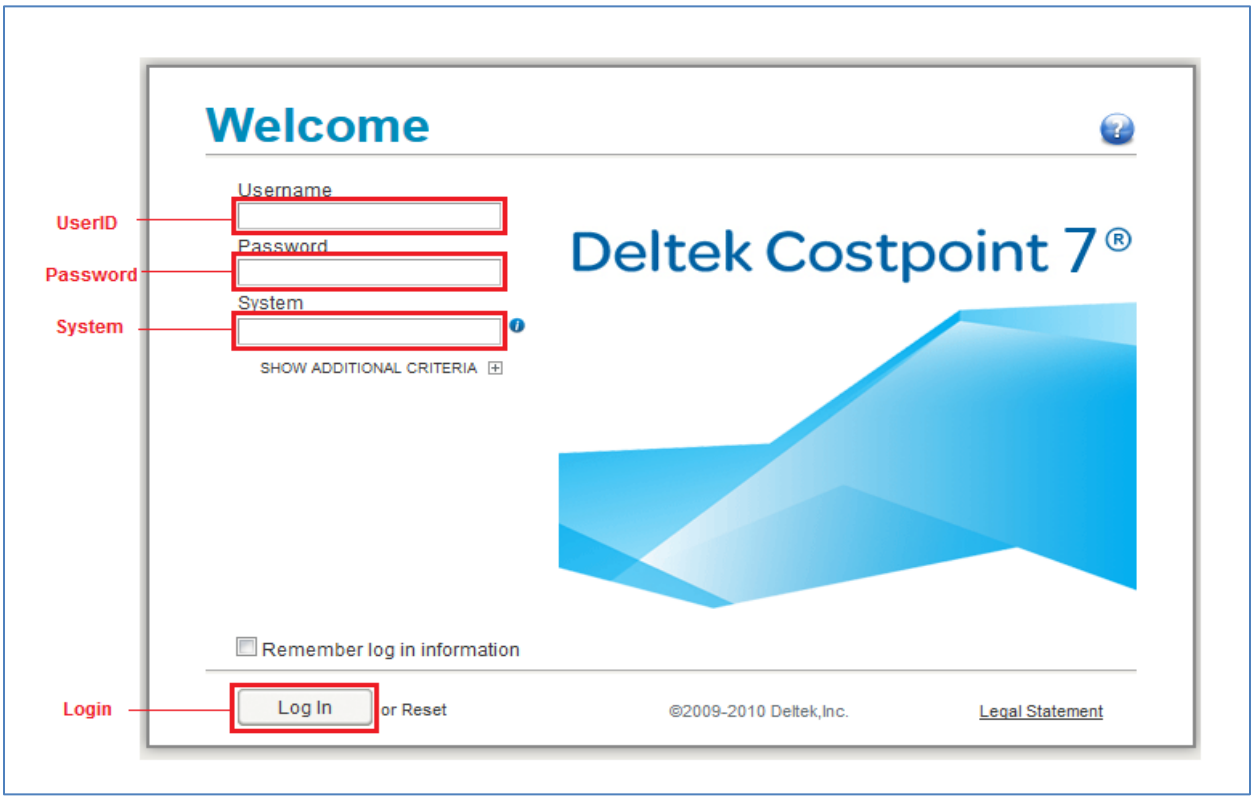


Figure 29: Control

Keyword

This is a list of all possible keywords for the selected control that the test can interact with. Shown below are the available controls for the *Login* button control.

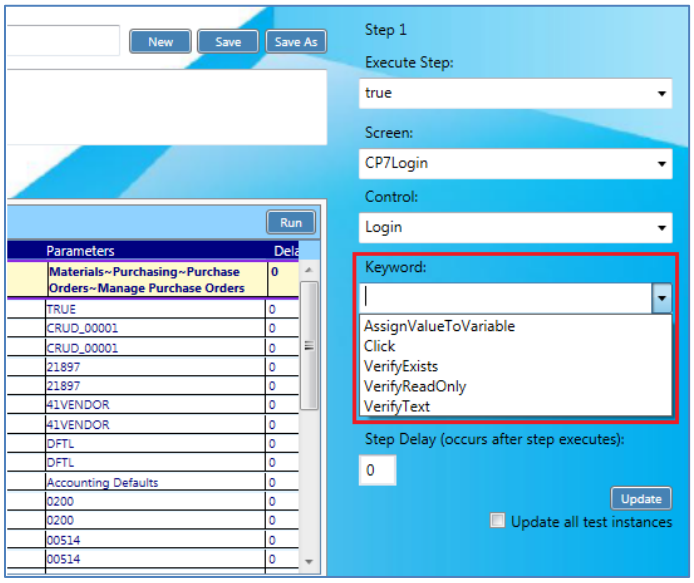


Figure 30: Keyword

Parameters

This is a list of all parameters required by the selected keyword. For example, the *VerifyText* keyword of the *Login* button control requires 1 parameter: the text to verify.

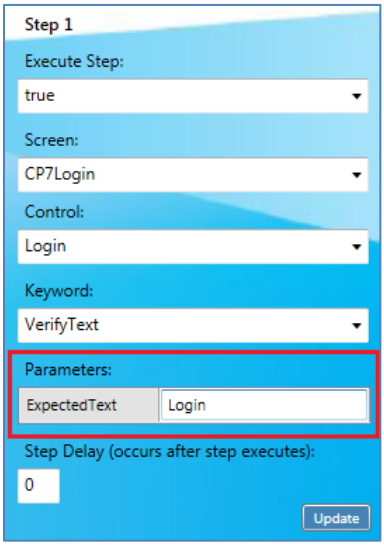


Figure 31: Parameters

Step Delay

This value is used to specify a time delay in seconds to be initiated after the step is executed. If *N* delay is specified for a step, the test editor will wait for *N* seconds after the step is executed before proceeding to the next step.

Help Box

The Help Box is located below the Update Panel. It is where information about keywords is displayed depending on the Keyword field in the Update Panel. Information include the keyword, its parameters, and the keyword's controls.

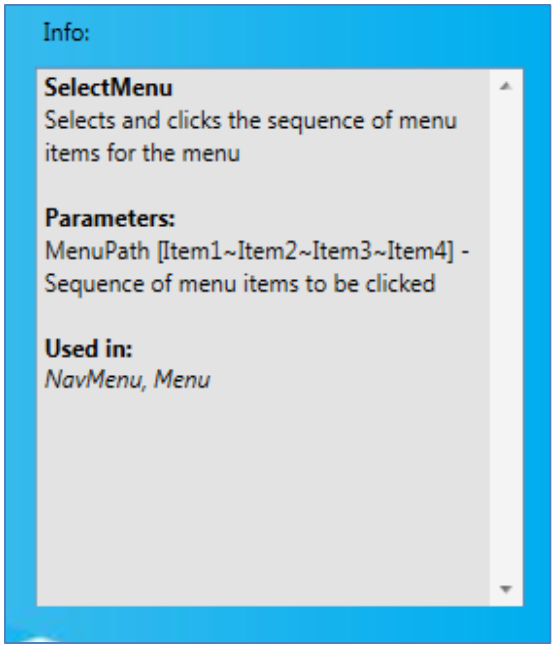


Figure 32: Help Box

New/Insert/Copy/Paste/Delete/Reorder Test Step

To add a new step row in the *Keyword View* grid, click on *New*. You can also insert a new blank step by clicking *Insert*. Click on *Copy* to copy an existing step, and *Paste* to paste a copied step. To delete an existing step, click on *Delete*. The *Up* and *Down* buttons are used to re-order the steps.

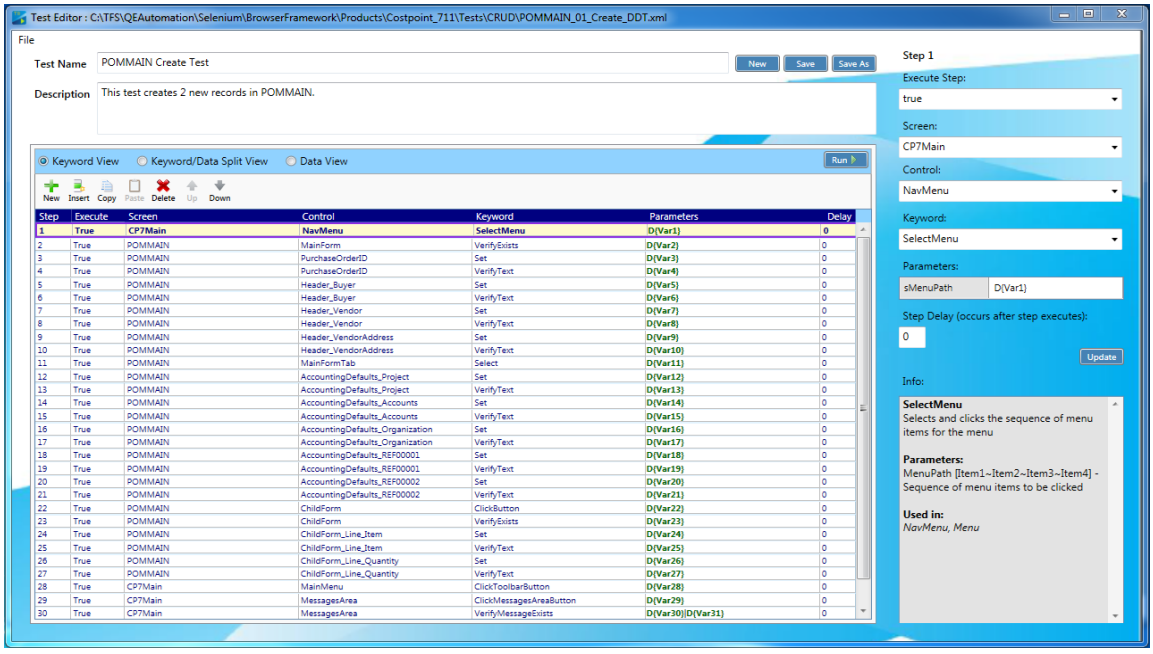


Figure 33: New/Insert/Copy/Paste/Delete/Reorder test steps

Data View

Select the *Data View* to display the test in terms of the data being processed by the test. You can also select the *Keyword/Data Split View* to show both *Keyword View* and *Data View* simultaneously.

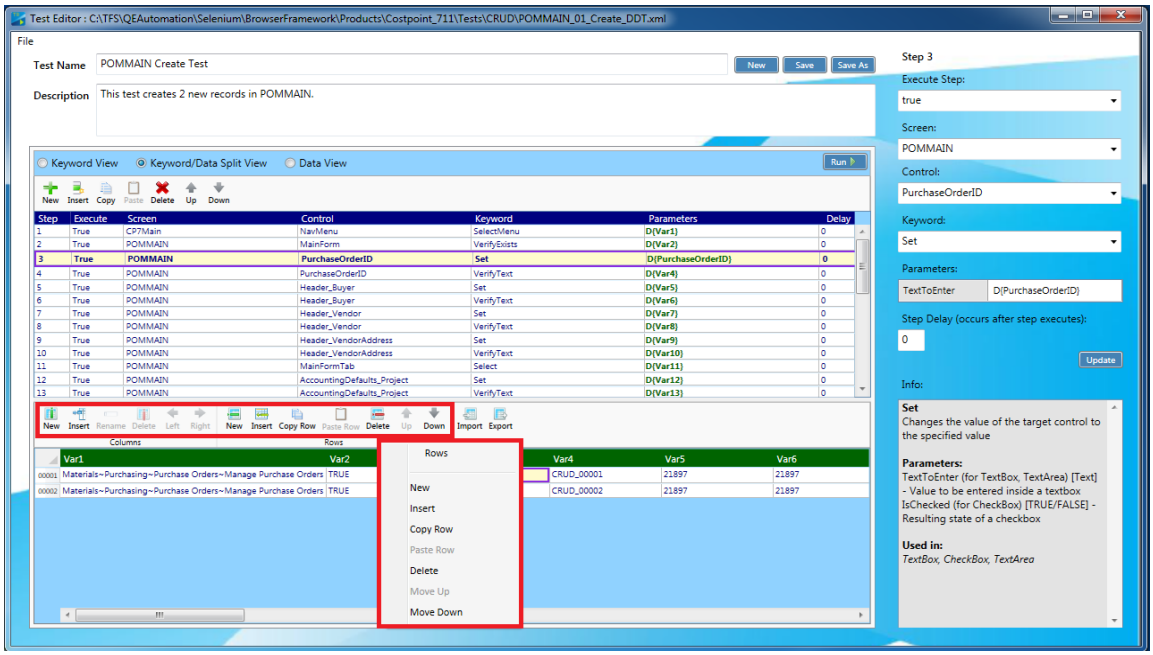


Figure 34: Keyword/Data Split View

Test Instance

A *Test Instance* is an iteration of a test tied up to a particular set of data. As seen in Figure 31, the two test instances are characterized by the two rows in the *Data View* grid.

You can add new instances/rows by clicking on the *New* button under the *Rows* toolbar group. Click on *Insert* to insert instances/rows before the highlighted instance/row. Click on *Copy Row/Paste Row* for copy and paste functionalities for instances/rows. Click on *Delete* to remove instances/rows. Re-order instances/rows by clicking on the *Up/Down* buttons.

Alternatively, right-click on any area of the *Data View* grid to access the *Context Menu*. This menu can be used to quickly access the functionalities for instances/rows, like the *New* and *Up/Down* buttons - all of which also appear under the *Rows* toolbar.

Using D{}

The two instances in Figure 31 use the same test steps, but can differ on parameters used for the steps. These values are referred to as *user-defined variables*, characterized by columns in the *Data View* grid.

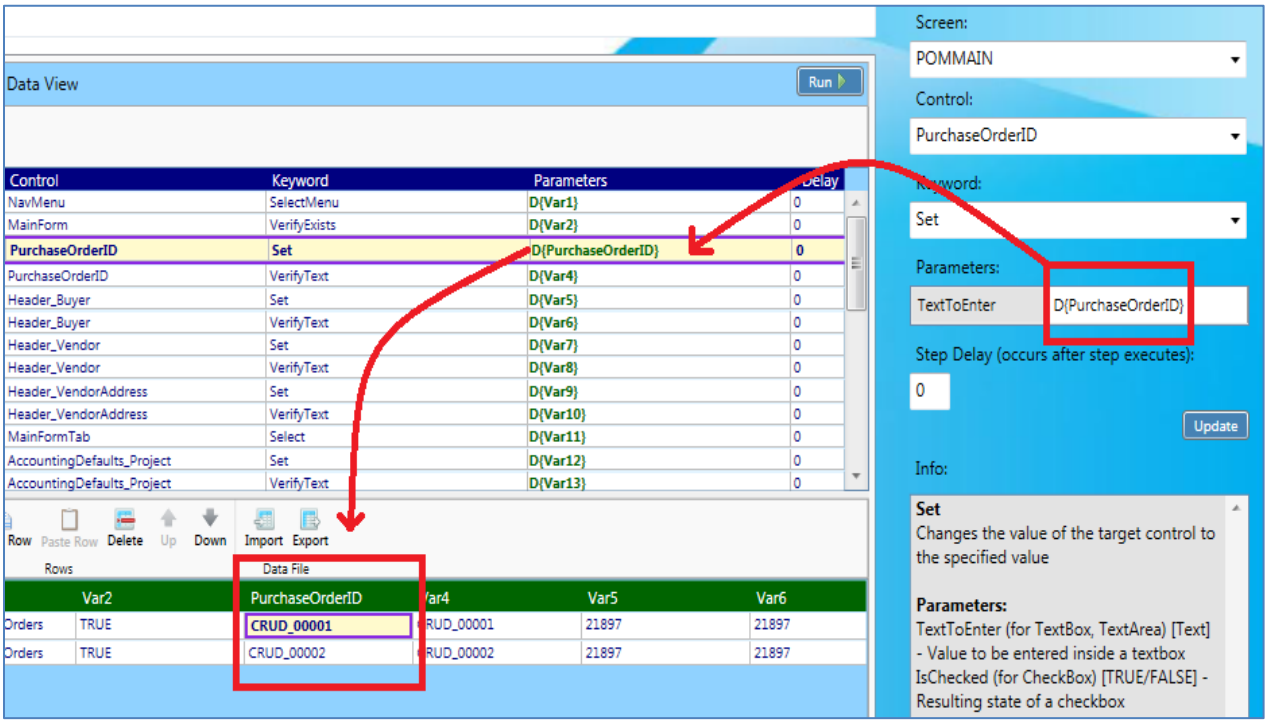


Figure 35: Using D{}

These variables are accessed as parameters in the *Keyword View* grid by enclosing the variable name by *D{}*. To access a user-defined variable named *PurchaseOrderID,* we simply use *D{PurchaseOrderID}* in the *Parameter* field under the *Keyword View* grid.

You can add new user-defined variables/columns by clicking on the *New* button under the *Columns* toolbar group. Click on *Insert* to insert columns before the selected column. Click on *Delete* to remove user-defined variables/columns or *Rename* to rename. Re-order columns by clicking on the *Left/Right* buttons.

Data Import/Export

The *Test Editor* supports exporting of data displayed in the *Data View* grid as CSV files or XLSX files. Similarly, you can also import data saved as CSV/XLSX files.

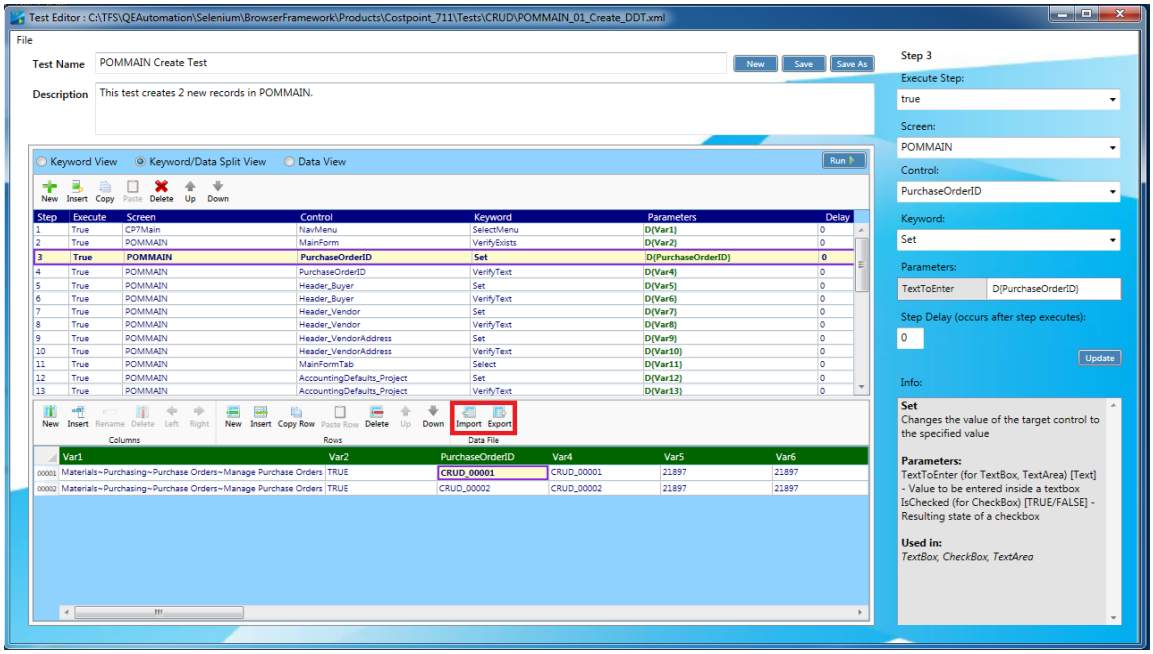


Figure 36: Import/Export Data into CSV/XLSX files

Running a Test: Ad hoc run

To execute the currently opened test, click on *Run*.

Ad Hoc Run Dialog

The *Ad Hoc Run* dialog is displayed to specify the test instance. When these are set, click *Continue* to proceed with execution.

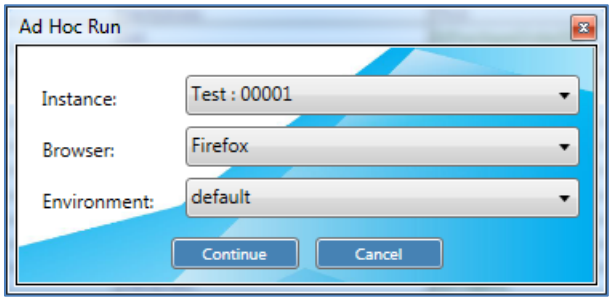


Figure 37: Ad Hoc dialog

Execution Results dialog

After the test execution, the *Execution Results* dialog will be displayed to show the results of the test. You can click on each row in this dialog to see the execution details of the selected step. During execution, a *SETUP* step is executed first prior to running the other steps, which includes logging in and navigating to the login URL, among other pre-execution events.

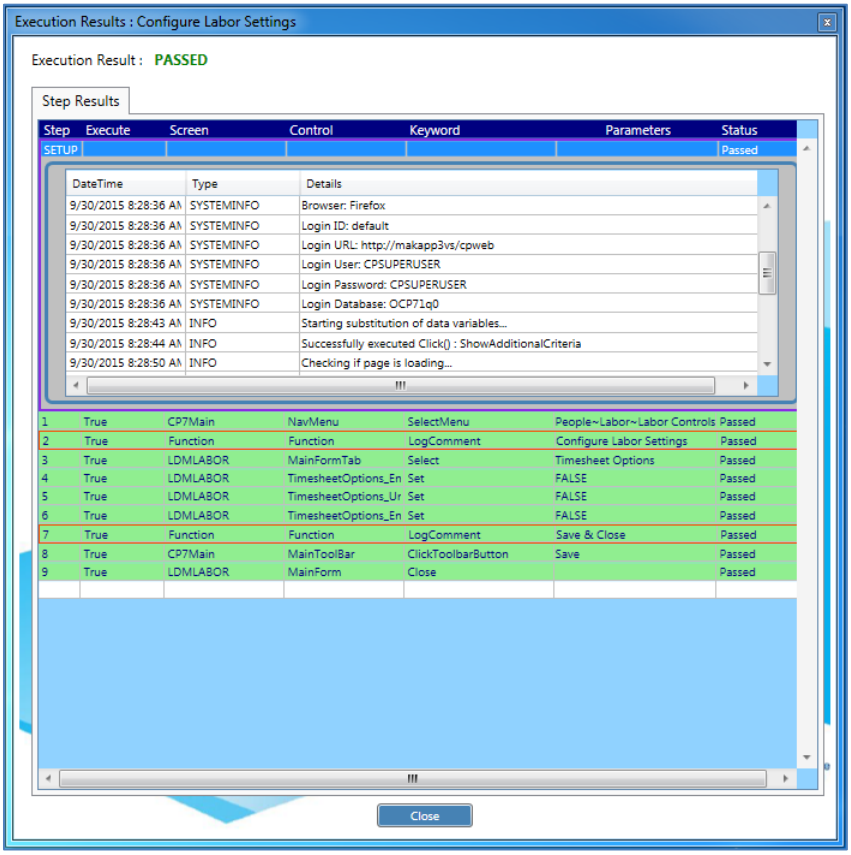


Figure 38: Execution Results dialog

Creating a Test Suite

Queueing Tests from the Test Explorer

In the *Test Explorer* tab, select a test and click *Queue* button to add the test inside the *test queue*:

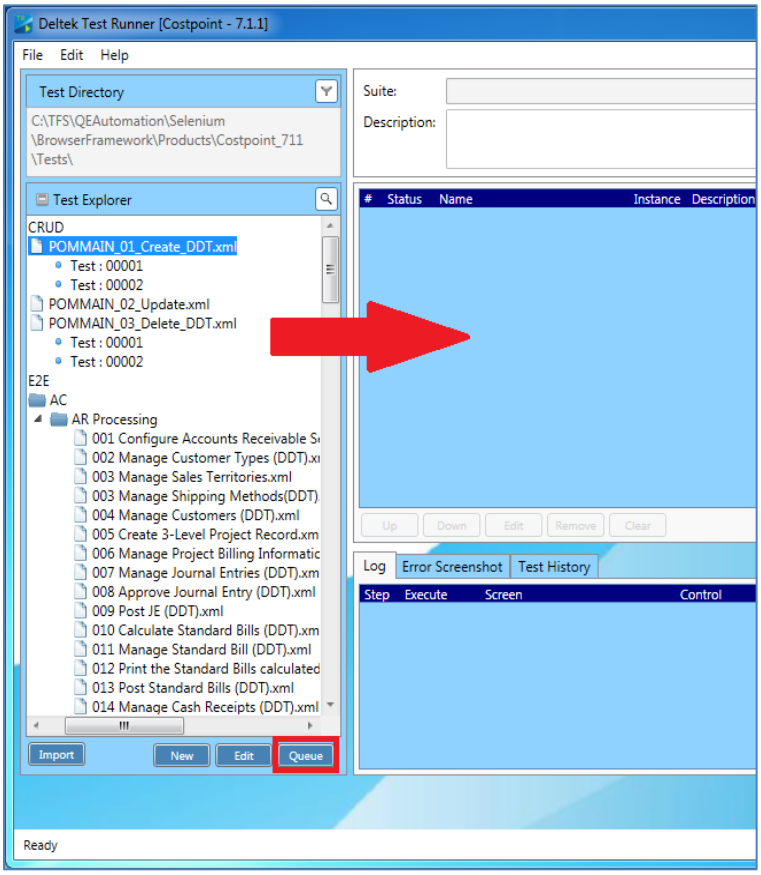


Figure 39: Queueing tests

Set the *Environment* and *Browser* from which the tests will be run. The *Keep Open* checkbox will allow a test in a test suite to keep its browser open when it finished executing. This way, the browser will be available for use by the succeeding tests in the suite:

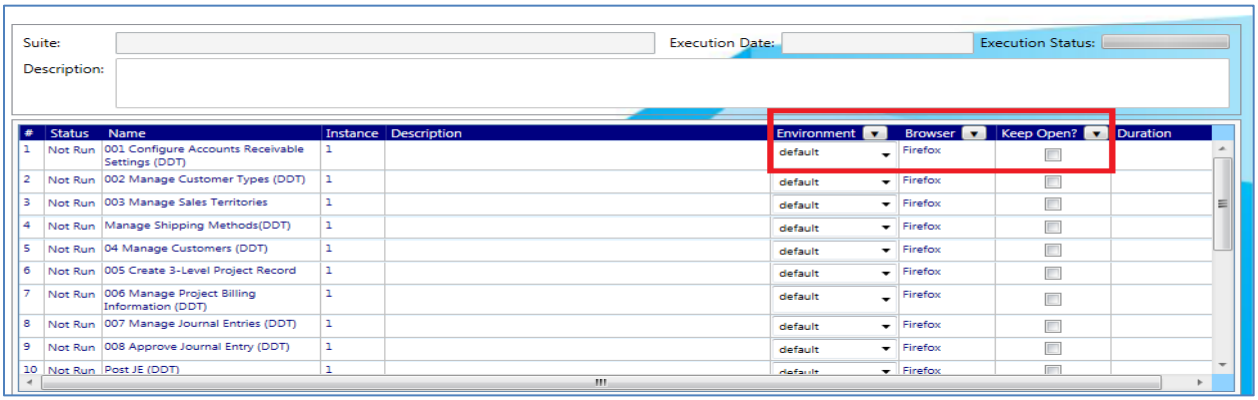


Figure 40: Configuring test suite settings

Editing tests in a test suite

To remove a test from the test suite, click *Remove*. To remove all tests from the test suite, click *Clear*. To re-order the tests inside the test suite, use the *Up* and *Down* arrows. You may also edit tests in the *Test Editor* by selecting a test in the *test queue* and clicking *Edit*.

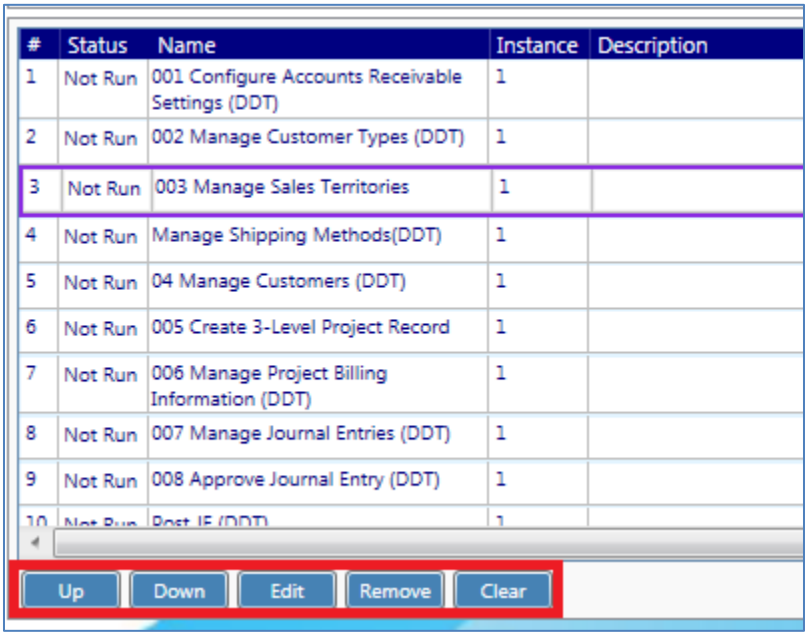


Figure 41: Editing tests in a test suite

Adding a description for a test suite

To add more information related to the suite, include them in the *Description* field.

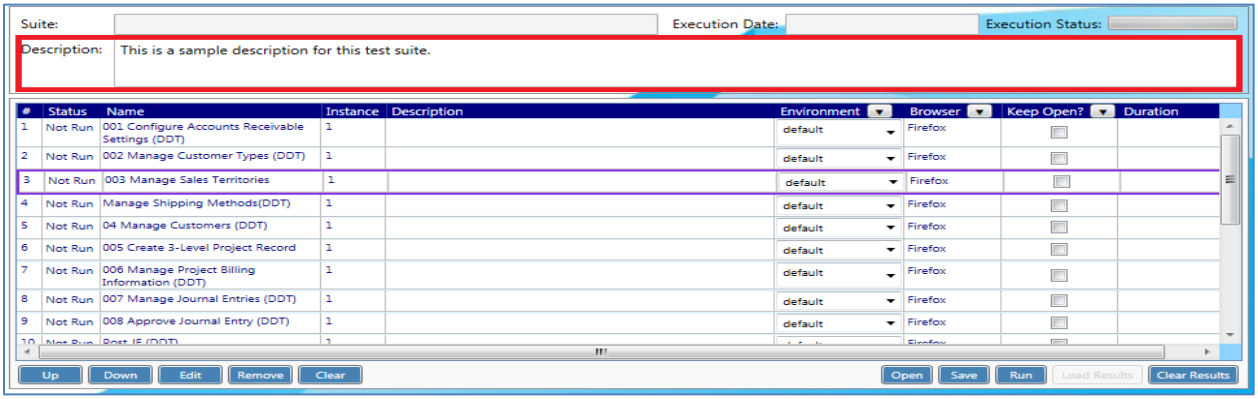


Figure 42: Adding a description for the suite

Saving a test suite

When all tests are added, click Save:

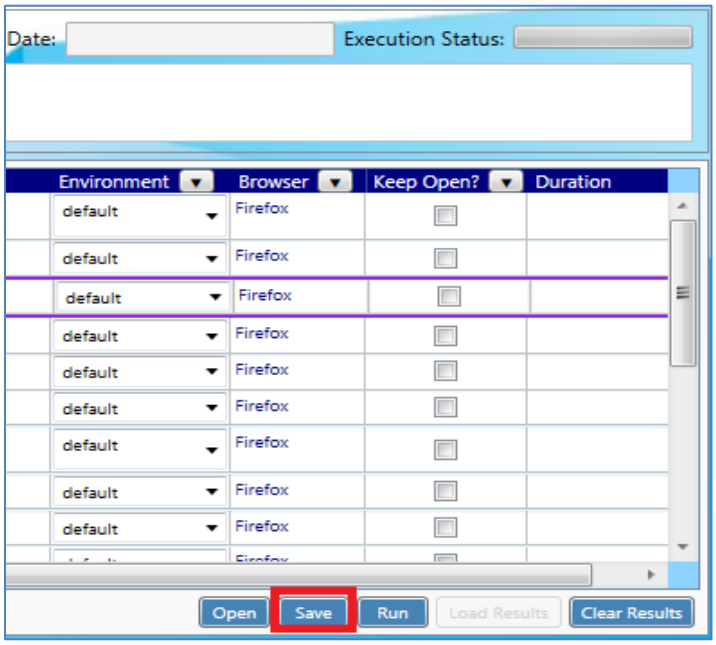


Figure 43: Saving a test suite

The *Save* dialog will appear. Enter the name of your new test suite. The path of the test suite file will be automatically displayed. Click *Save* to save your file.

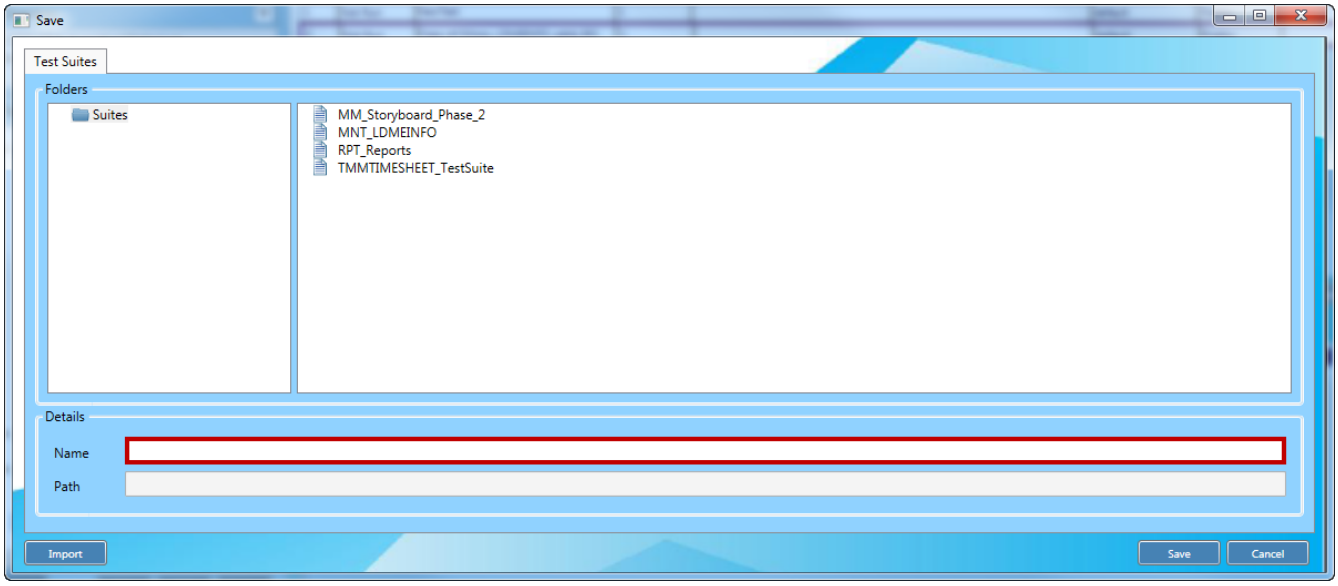


Figure 44: Saving a new test suite file

If you want to overwrite an existing test suite, just select an existing file you want to overwrite:

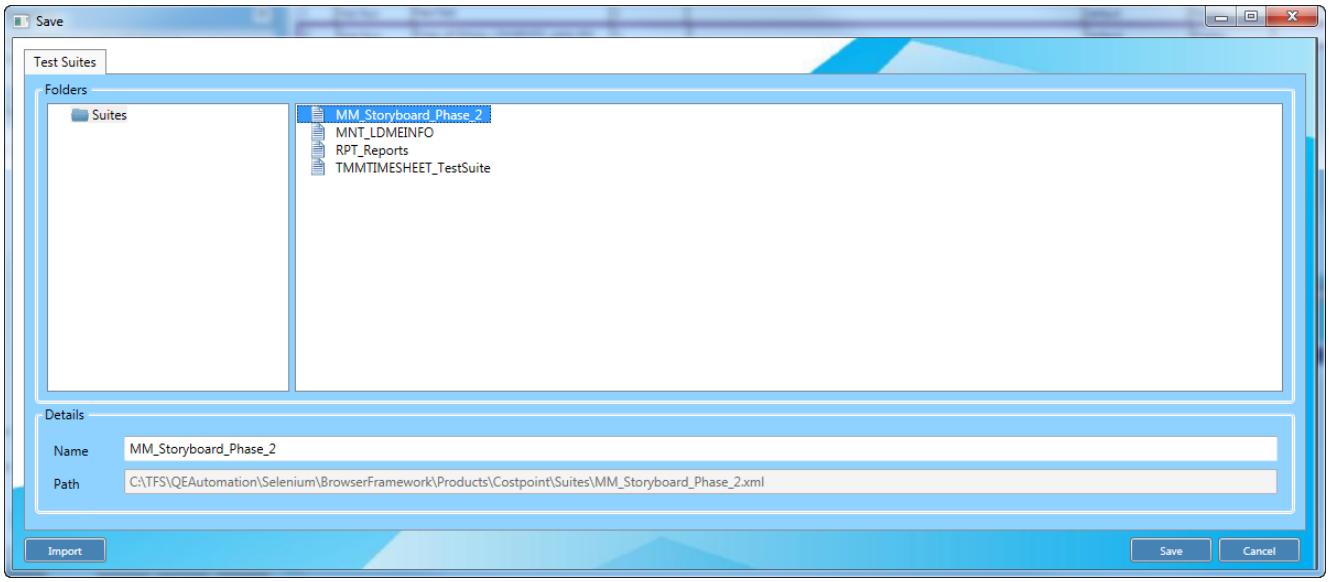


Figure 45: Overwriting an existing test suite file

Executing a Test Suite

Open existing test suites

To browse for existing test suites, click *Open* below the test queue:

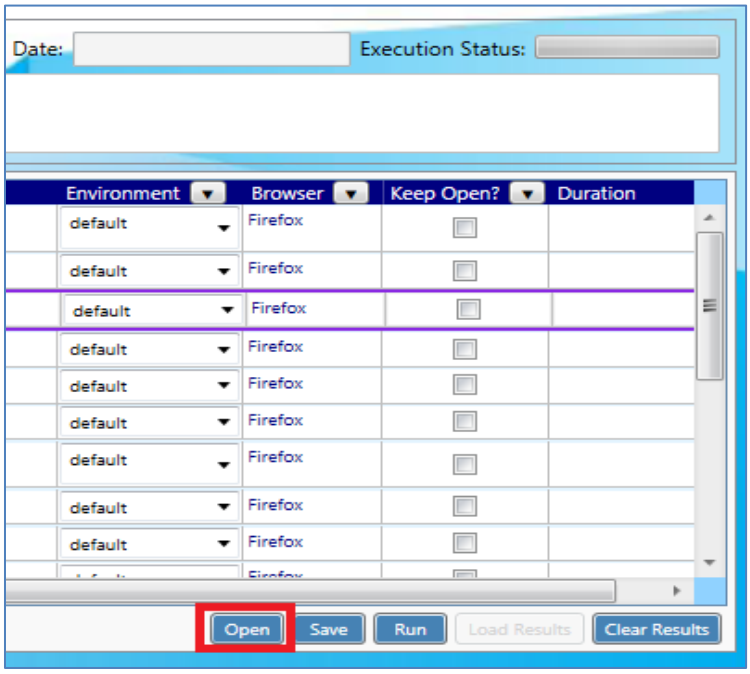


Figure 46: Opening an existing test suite

The *Open* dialog will appear. Select a test suite file and click *Open*. The tests will now be loaded in the *Test Runner*:

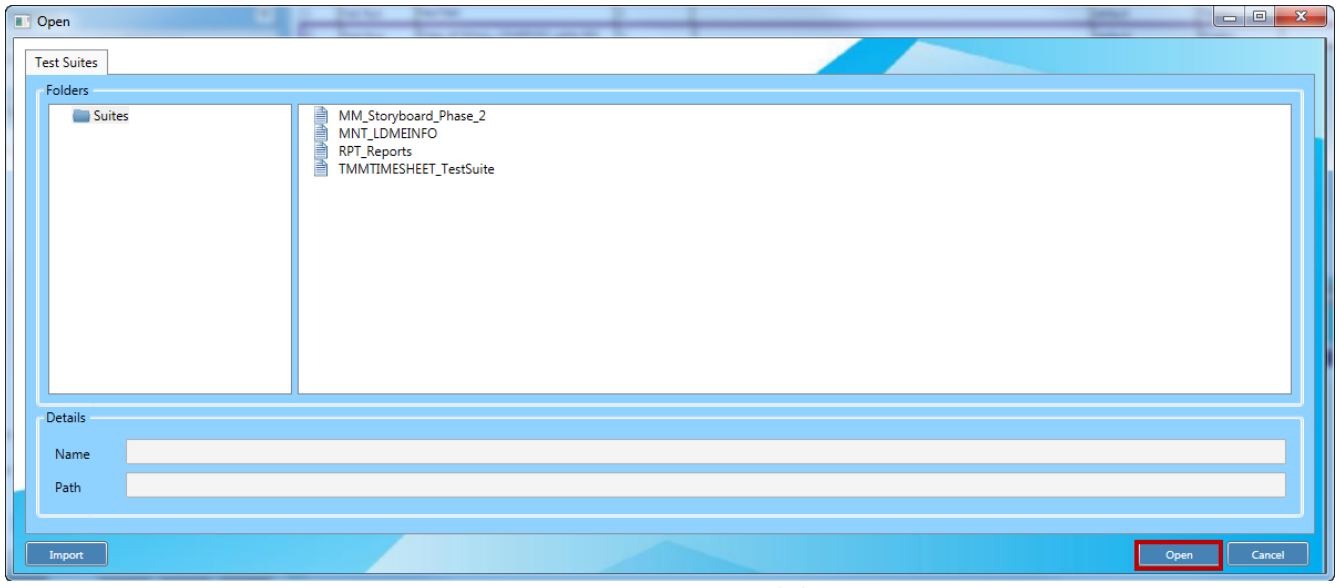


Figure 47: Open test suite dialog

If a test suite doesn't currently exist in the Open dialog, the *Import* button allows the user to import a test suite from another location.

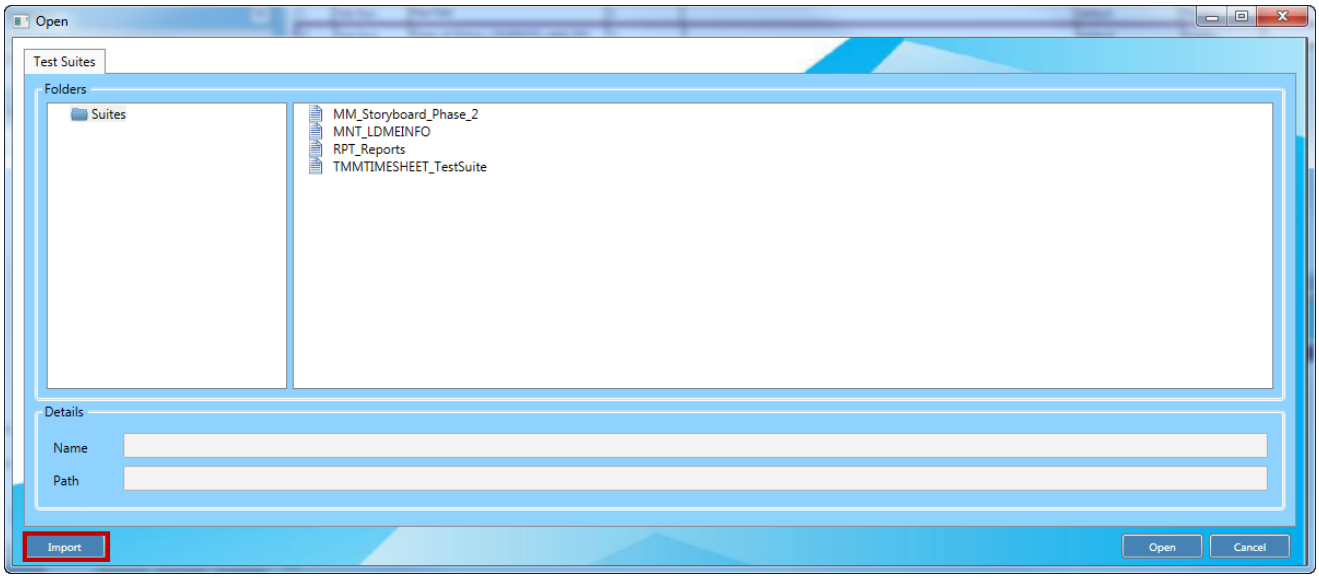


Figure 48: Import test suite

When the *Import* button is clicked, a new file dialog will display and prompt the user to select the test suite to import. After importing, select the test suite and click *Open* to load the tests.

Running Test Suites

Click *Run* to execute a test suite:

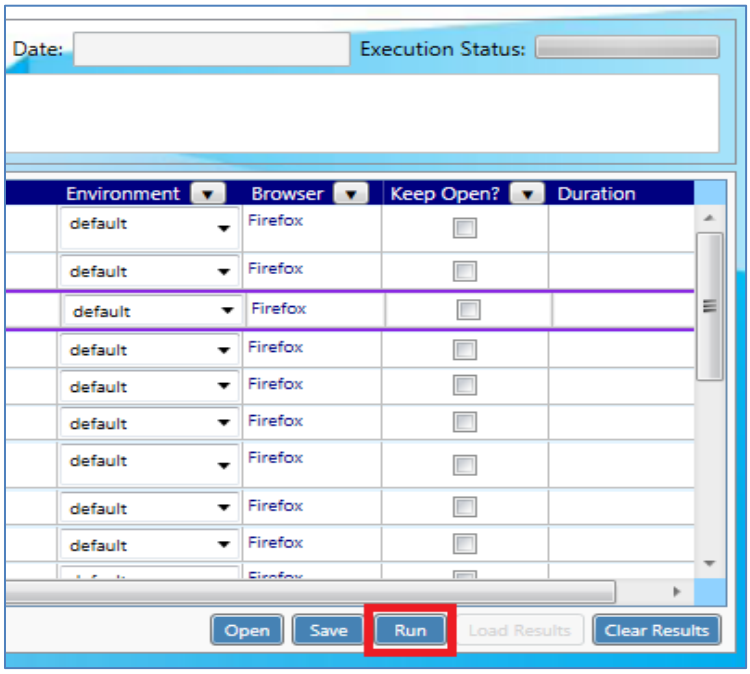


Figure 49: Running a test suite

During execution, the *Execute* dialog will appear. To cancel the current test, click *Cancel Test*. To cancel the whole test suite, click *Cancel All Tests*:

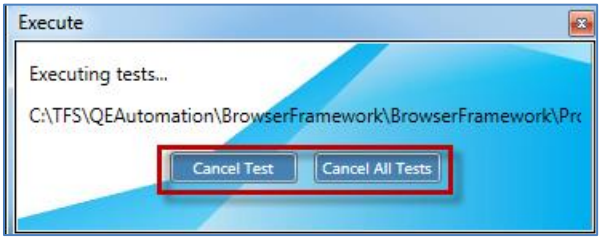


Figure 50: Execute dialog

Please note that clicking on *Cancel Test* and *Cancel All Tests* can take some time to process and complete. There will also be occasional instances wherein test cancellation fails to complete because the test interruption was initiated at a critical part of the test execution. As an alternative, you can initiate a definite cancellation of a test by restarting the *Test Runner*.

Test Execution Logs

After the test execution, the results will be displayed below the *test queue*. In the *Logs* tab, test execution logs can be viewed. This view displays the steps of the selected test. The test step is shaded green if it passed, and shaded red if it failed:

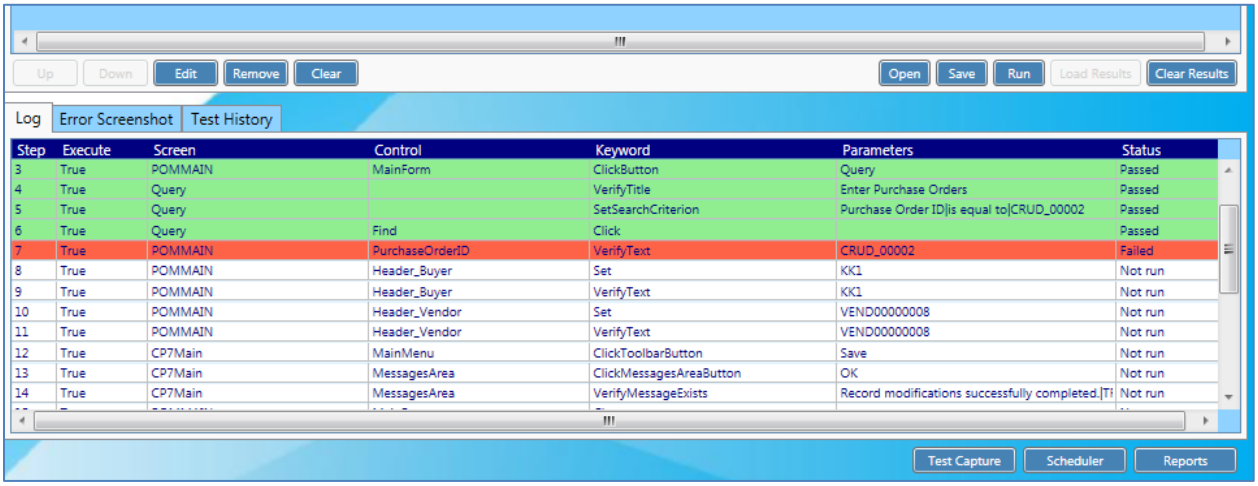


Figure 51: Test execution logs

Click the row that contains the step and details of the step will be displayed:

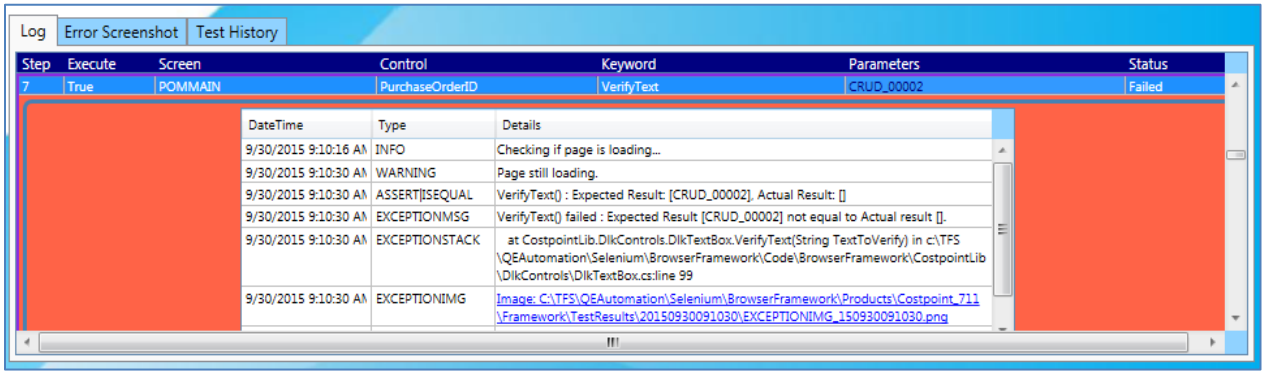


Figure 52: Detailed test step logs

Error Screenshot

In the *Error* Screenshots tab, the user can view a screenshot of the application when the error took place. To view the full image in *Microsoft Paint*, the user may click the *MsPaint* button:

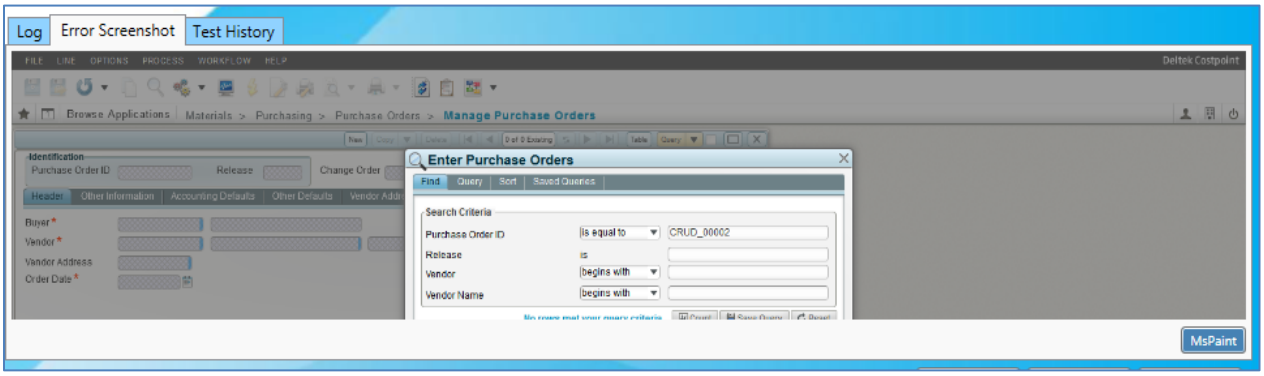


Figure 53: Error screenshot

Test History

In the *Test History* tab, the user can view the execution date and time of the test suite result. It also displays whether the test passed or failed:

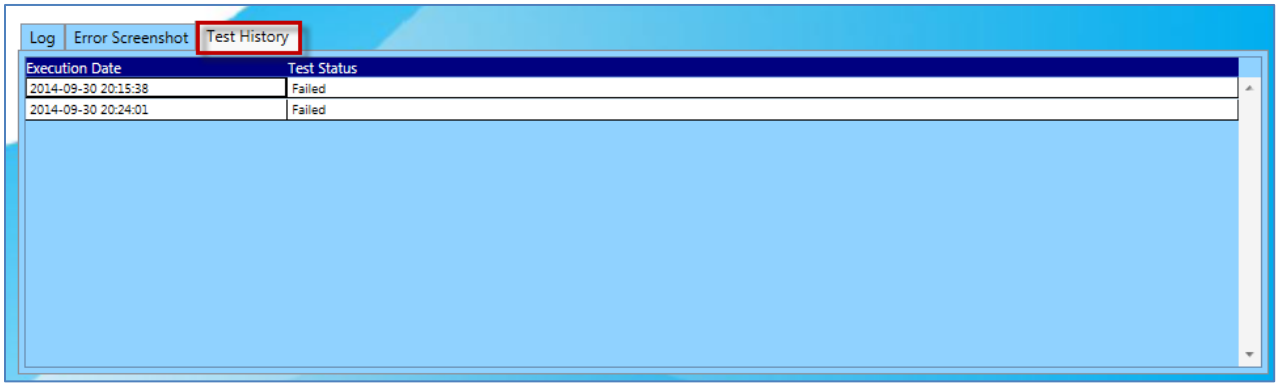


Figure 54: Test History

Examining Test Results

Loading Test Suite Results

Click *Load Results* to view past results. The *Load Suite Results* dialog will appear. Choose a test execution result from the dropdown menu and click *Select* to load the results:

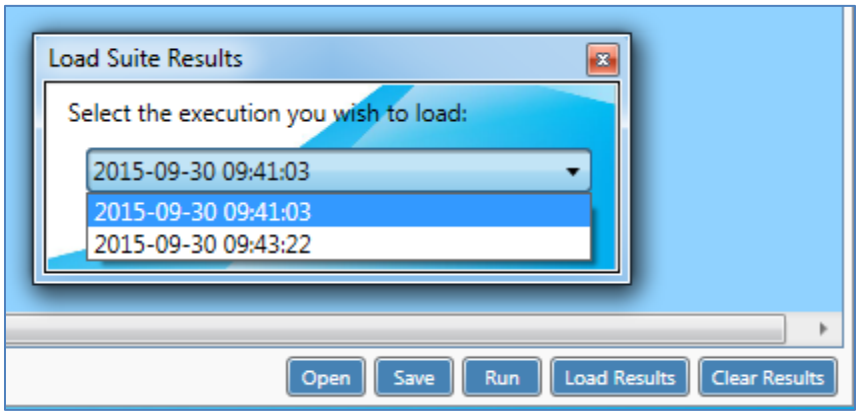


Figure 55: Loading past test suite execution results

Click *Clear Results* to clear and empty the *test queue* of execution result information. However, this does not delete the test execution results permanently and can be viewed again by loading from the *Load Results* button:

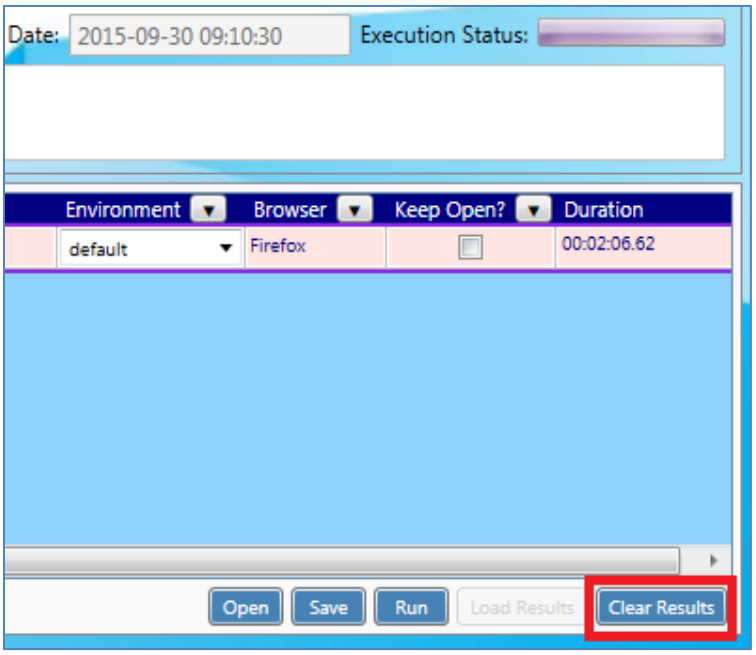


Figure 56: Clear Results

Scheduling Test Executions

Test suites can be scheduled to run at specified times set up via the *Test Scheduler*. To begin scheduling suite executions, click the *Scheduler* button on the *Test Runner*.

Creating a Test Schedule

Figure 57: Test Scheduler

On the Test Machines tab, select a test machine to display all suites available for scheduling. The grid on the right contains the schedules for each day of the week. Choose a day you want the suite to run and click the Add button. The *Set schedule* form will be displayed.

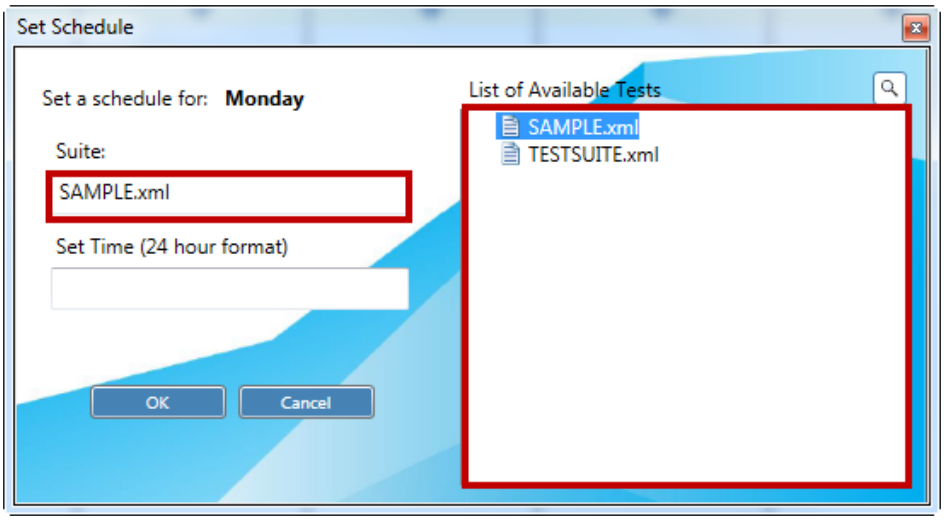


Figure 58: Set Schedule form

In the *List of Available Tests* pane, select the test suite. After selecting the test suite, the name of the chosen suite will appear in the *Suite* textbox.

Set the desired execution time in the *Set Time* textbox. Time inputs should be in military format. For example, you may input 14:00 to have your selected suite run at 2 PM.

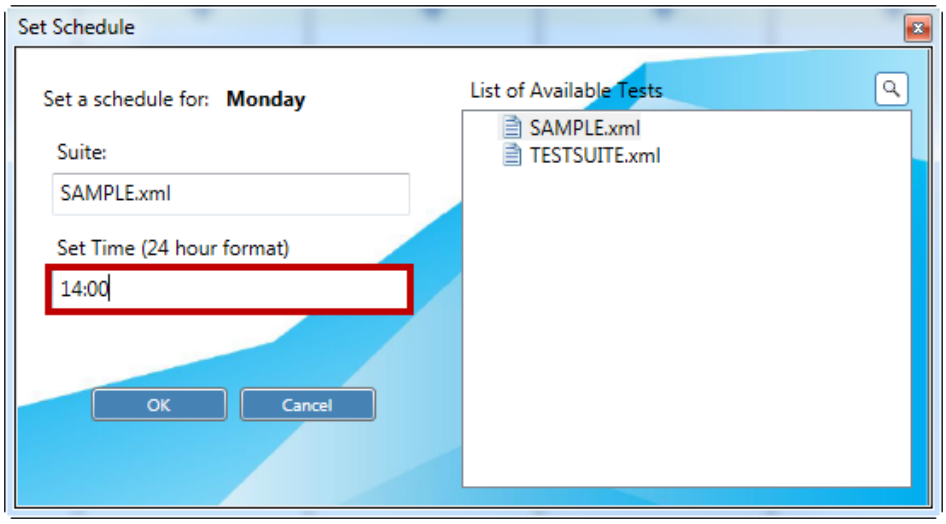


Figure 59: Setting execution time

After setting the time, click *OK*. The schedule will now appear in the Test Scheduler. Click *Save* to save your changes.

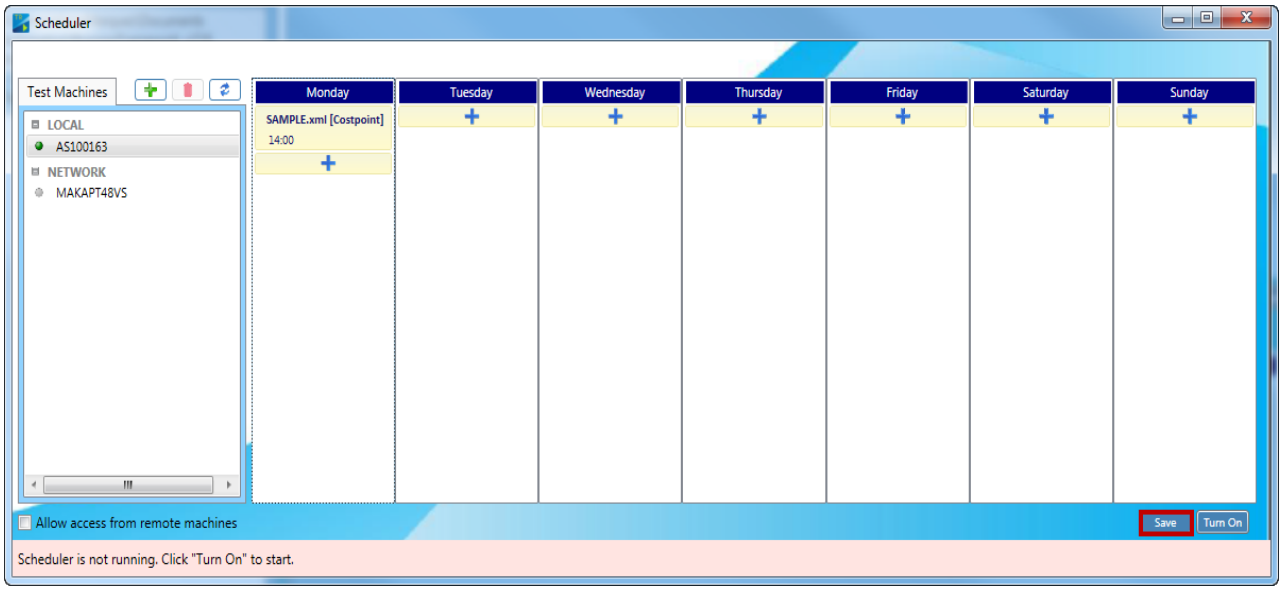


Figure 60: Test Scheduler with newly added schedule

Queueing Test Schedules

The user can queue a new test schedule after a scheduled test finished execution. To queue a test schedule after an existing test schedule, click the Add button below an existing test schedule. A new *Set Schedule* form will be displayed.

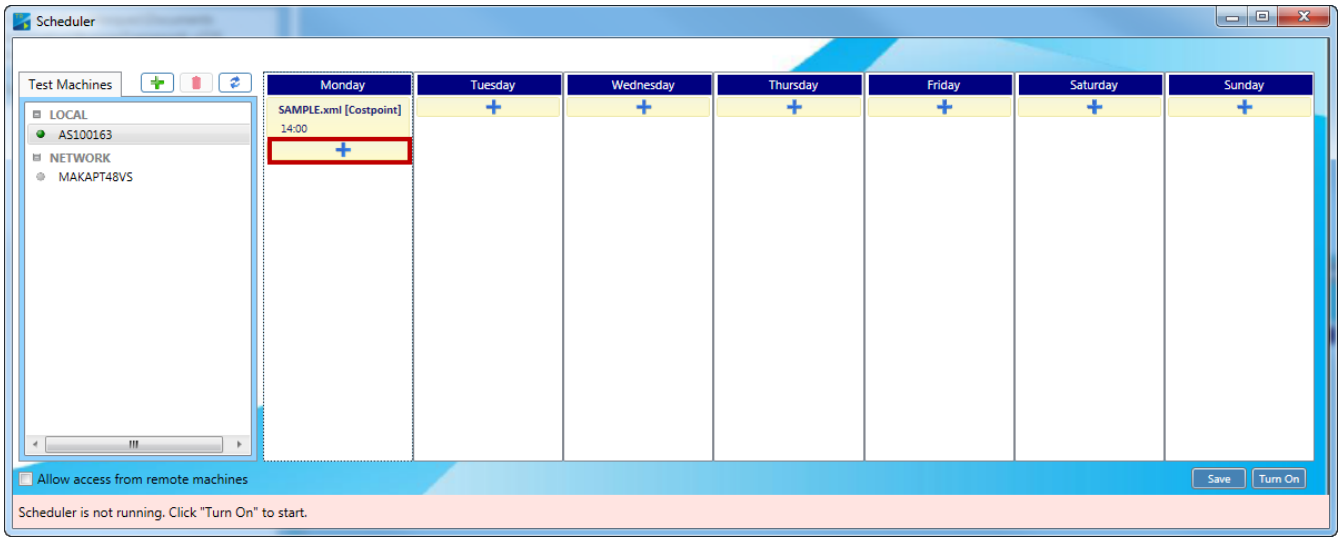


Figure 61: Queueing Test Schedules

In the *Set Schedule* form that appears, notice that the *Queue after* textbox already contains the suite scheduled last on queue. In the *List of Available Tests* pane, choose the test suite you like to queue next.

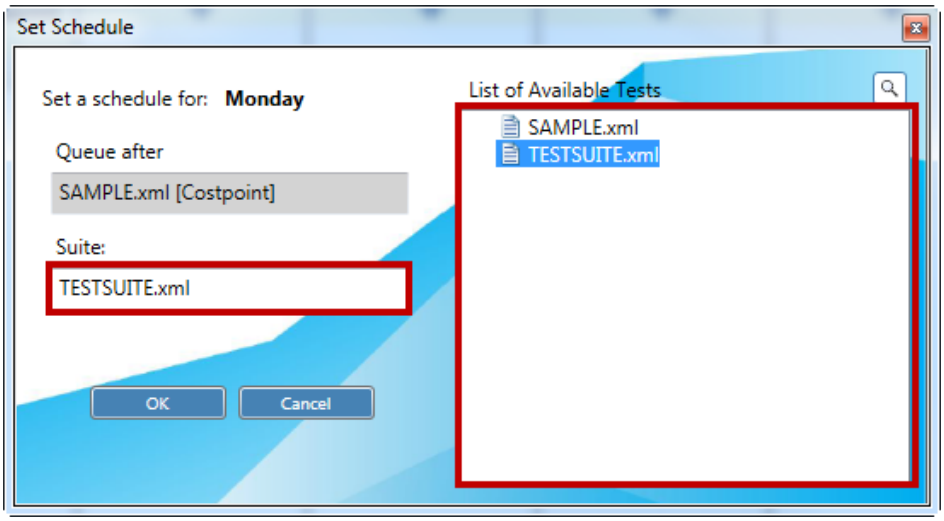


Figure 62: Queueing a Test Schedule with Set Schedule form

Click *OK* to set the suite next on queue. The user may add more suites on queue. After adding all suites, click *Save* to preserve the changes.

Turn On Scheduler

Click *Turn On* to monitor any test suite scheduled for execution, and run the tests at the specified time.

Remote Test Machines

The Test Scheduler also allows users to schedule test executions on remote machines.

Allow access from remote machines

To use this feature, the following pre-requisites should first be checked:

- Remote machine is in same network with the user's local machine.
- The user should have administrative rights to remote machine
- The remote machine has an installation of *Test Runner*.

Open *Test Scheduler* on the remote machine you like to access and put a check on *Allow access from Remote Machines*.

On your local machine click on *Add* and set the remote machine's host name or IP Address. The remote machine will now be visible in your local machine's *Test Machines* tab.

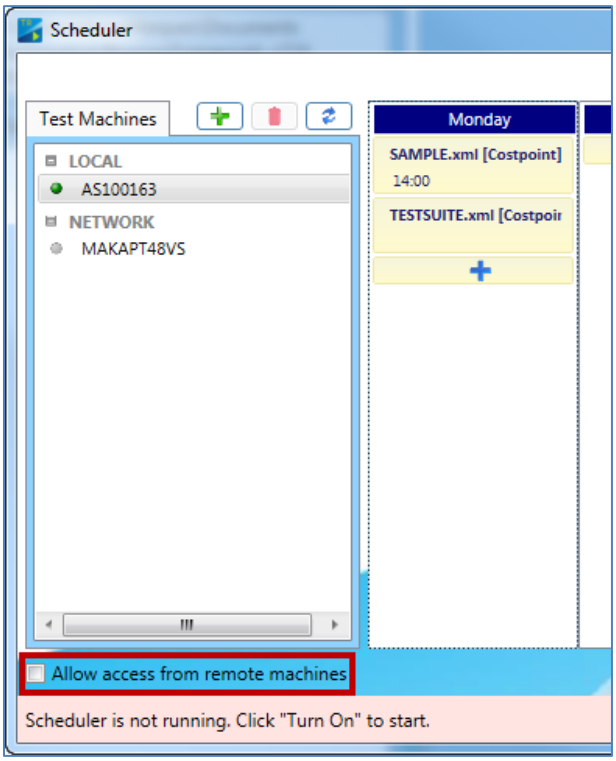


Figure 63: Remote test machines

Scheduling test suites on remote test machines

After you have added the remote machine, the process of scheduling your tests is similar to scheduling and running tests on your local machine.

Test Capture

The *Test Capture* feature allows the user to record simple actions or verify some elements from *Costpoint 7* web application and convert them into a test script. Click the *Test Capture* button found at bottom right corner of the Test Runner window to access *Test Capture*:

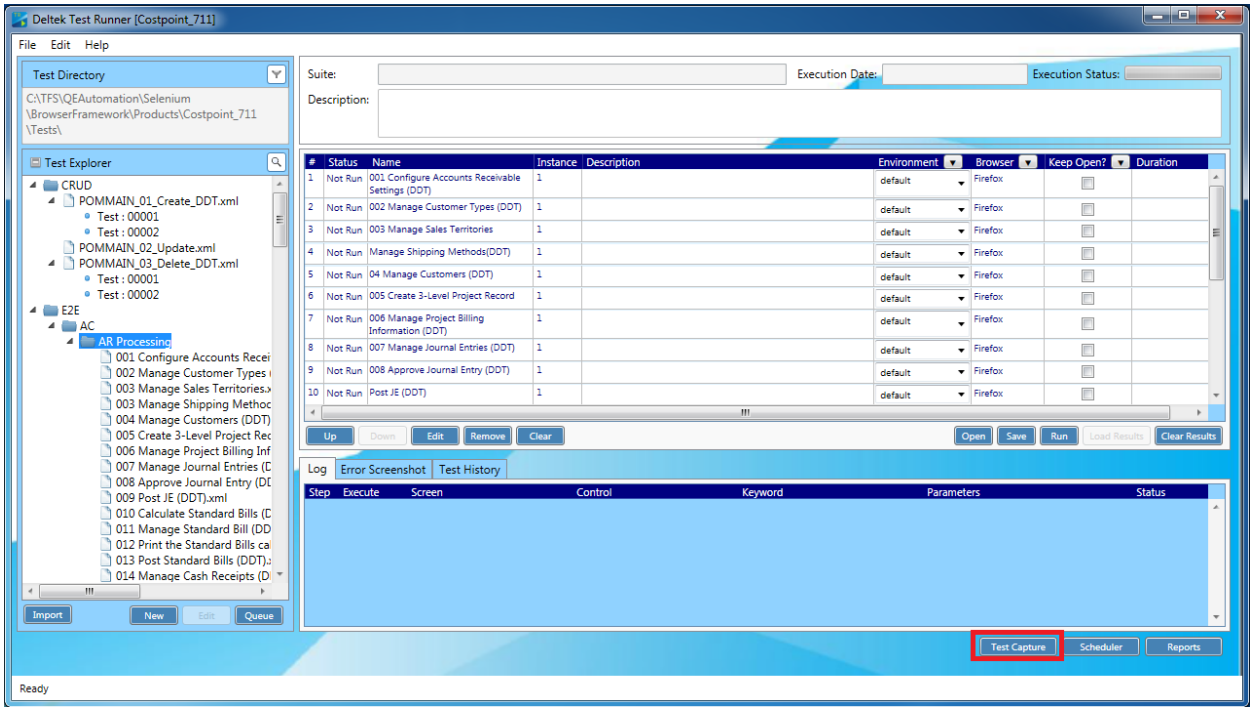


Figure 64: Test Capture

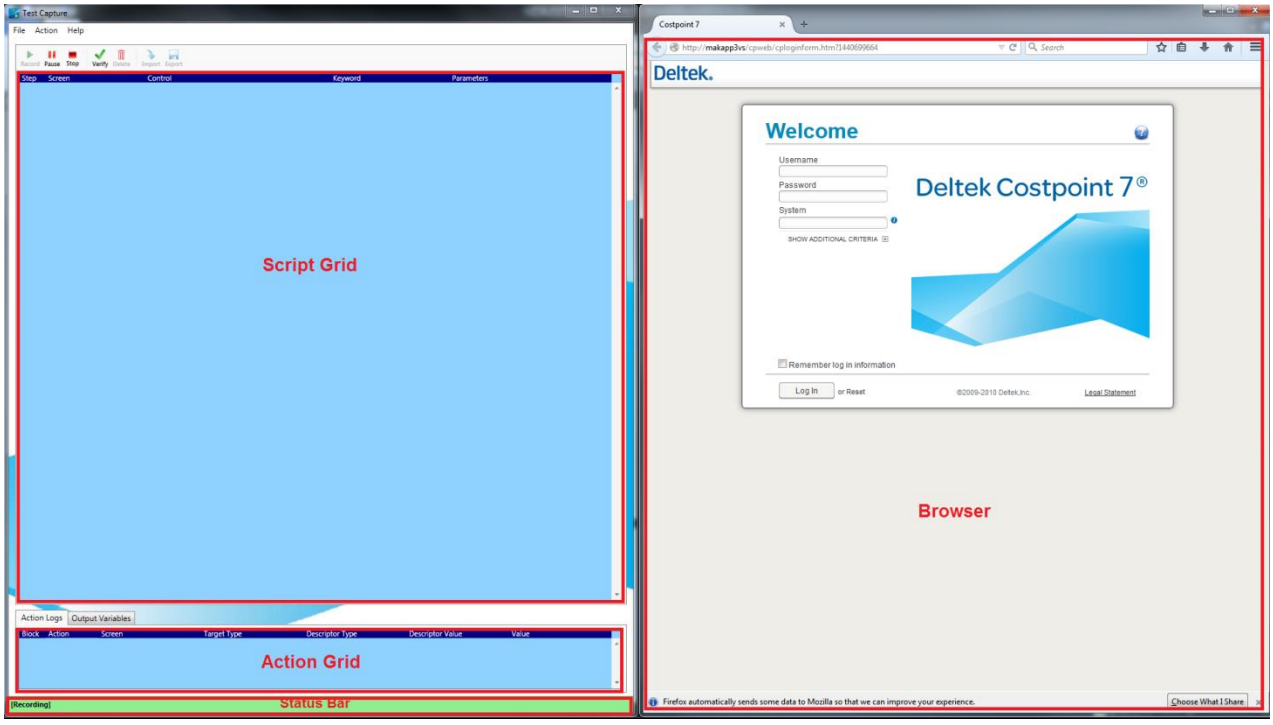


Figure 65: Test Capture (left) and the browser (right)

Below are the simple usage instructions of this feature. It is also advised to take a look at *Appendix C: Test Capture Limitations* for additional information about the feature.

Script Grid

This grid displays the test script created during the recording.

Action Grid

This grid contains the actions being recorded for tracing. Captured clicks and keystrokes are shown here.

Commands

Record 
This will open a new instance of Mozilla Firefox and start the recording of the test. After clicking this button, clicks and keystrokes performed on the web application will be captured and converted into test steps. The Script grid and the Action grid will be populated as long as the recording is ongoing.

Start Browser
This will start a new instance of Mozilla Firefox. This is used for instances where the user wants to record a specific set of steps without having to go through the login steps. It can be accessed through the menu bar: *Actions* > *Start Browser*.

Verify 
This will verify the current value of the selected element.

Delete 
This will delete current selected step. The user can only perform step deletion if Test Capture is in Paused state.

Pause 
The recording will be paused. Actions on the browser will not be recorded until the *Record* button is pressed.

Stop 
The recording will be stopped and the browser will be closed. Clicking the *Record* button after stopping the recording will start a new recording and clear the current test script.

Clear 
This will clear the entire recorded test script.

Import 
This will import an existing test script into test capture. The user will be given option to playback imported steps when resuming test recording.

Export 
This will save the recorded test script into XML format. The exported test script may be further edited in the *Test Editor*.

Status Bar

The status bar provides the user information on the ongoing status of the recording. The colors of the status bar have significant meanings that the user must be aware of:

[Recording]

Green – The test recording is ready to accept any user action.

[Ready to Verify]

Yellow – The test recording is in *Verify* state. Clicking on an element will verify its current value.

[Wait]

Light Yellow – Signal for the user to wait before performing another action.

[Paused]

Orange – The test recording is paused.

[Info]

Pink – Displays information messages (or errors) to the user

Auto Login

The user can opt to perform automatic login to Costpoint, by selecting any of the *Environments* defined in the *Settings* form.

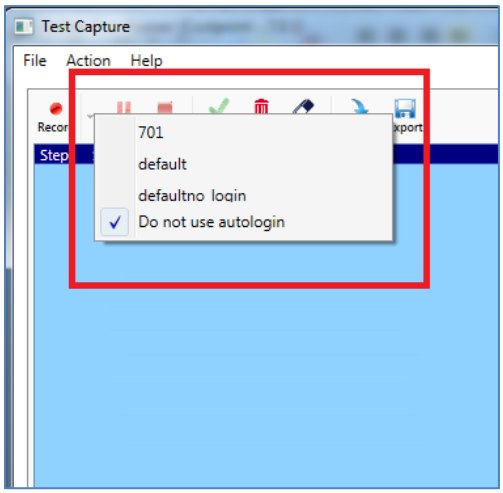


Figure 66: Auto Login

Recording will only commence, as soon as the automatic login finishes. The default setting is 'Do not use autologin'.

Alternate Keywords: ALT+Click

Alternate keywords are control keywords accessible to users in addition to 1 action keyword (ex: *TextBox -> Set*) and 1 verify keyword (ex: *TextBox -> VerifyText*) that are available by default.

For now, only 2 control types make use of alternate keywords: *TextBox* and *Table* controls.

Control Type	Action Keyword	Verify Keyword	Alternate Keyword
TextBox	Set	VerifyText	*ClickTextBoxButton*
Table	SetTableCellValue	VerifyTableCellValue	*GetTableRowWithColumnValue*

To access alternate keywords, instead of the just clicking on a page element, use **ALT+Click**.

Output Variables

Some control keywords make use of *Output Variables* as parameters. These variables are used to store values retrieved by these keyword and can be accessed by succeeding steps via the **O{OutputVariable}** notation.

One of the keywords that make use of an output variable is *GetTableRowWithColumnValue* keyword of the *Table* control:

KEYWORD	Description	Parameters
GetTableRowWithColumnValue	Assigns row number that matches Value under the column ColumnHeader to VariableName	ColumnHeader Value **VariableName**

When the alternate keyword for the *Table* control (*GetTableRowWithColumnValue*) is accessed by the user using ALT+Click, a dialog will be displayed to get from user the output variable name to be used for the keyword:

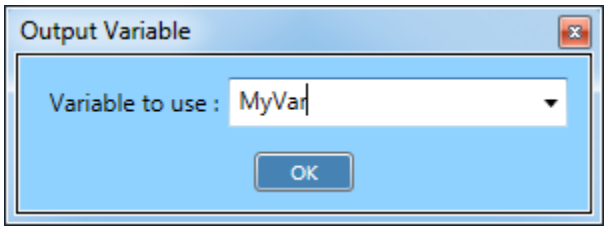


Figure 67: Output Variable dialog

Once a name has been provided by the user, the output variable (*MyVar*) will be used as the *VariableName* parameter in the *GetTableRowWithColumnValue* keyword (Step 3).

All output variables created by user via the *GetTableRowWithColumnValue* keyword will be listed in the *Output Variables* tab.

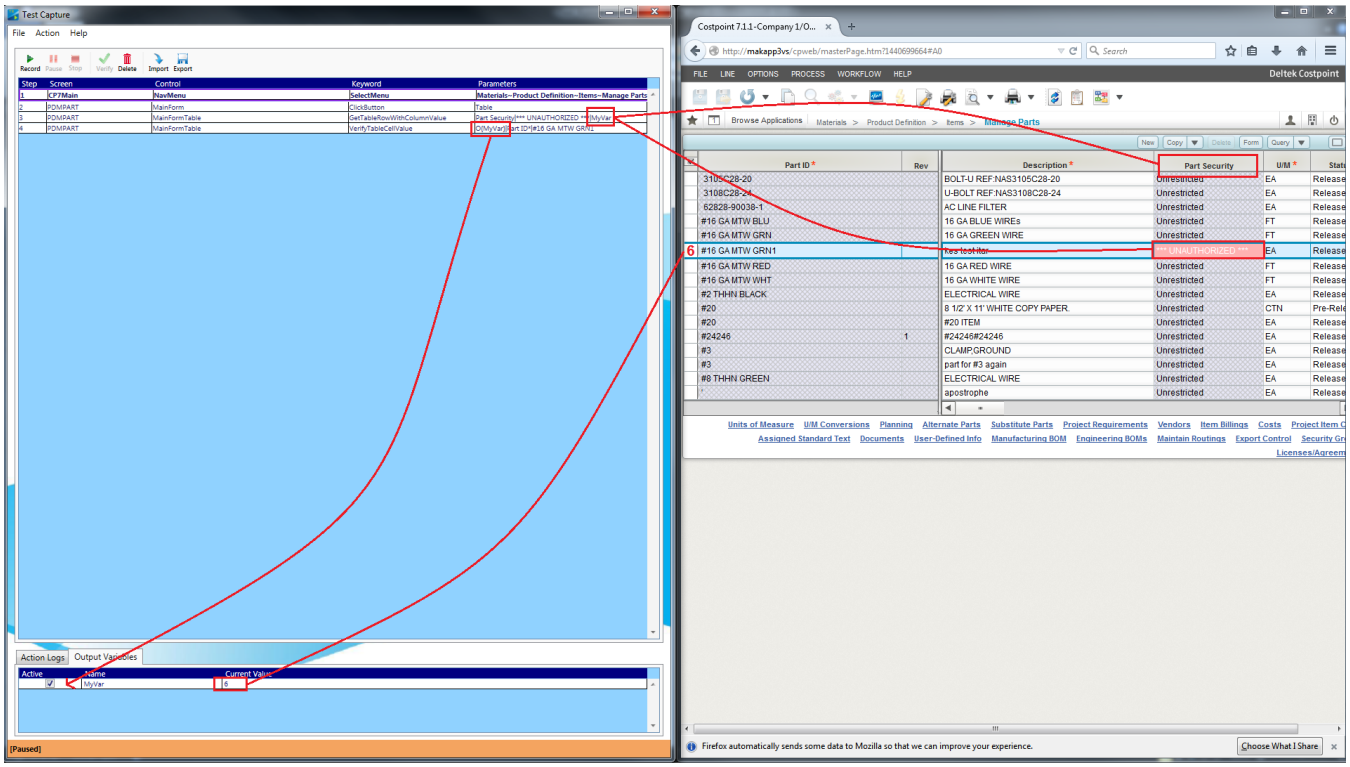


Figure 68: Using OutputVariables with GetTableRowWithColumnValue keyword

Notice that MyVar was also used for any succeeding step working with the same Table control, like in Step 4 where the VerifyTableCellValue keyword was used, because MyVar's Active checkbox is checked in the Output Variables tab. If user wishes to use the actual value (6), instead of the MyVar, he simply unchecks the Active checkbox.

For now, only the Table control type makes use of output variables.

Reports

Test Runner has a built in reporting to provide the status of execution for all test suites. The report can be accessed by clicking the *Reports* button.

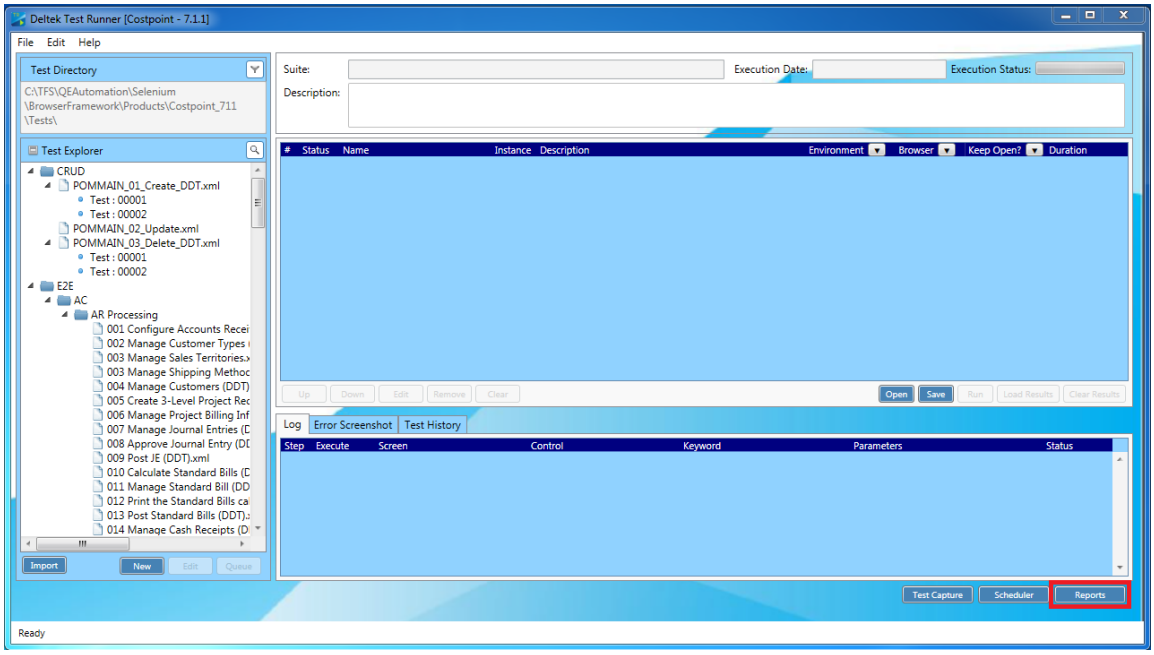


Figure 69: Reports

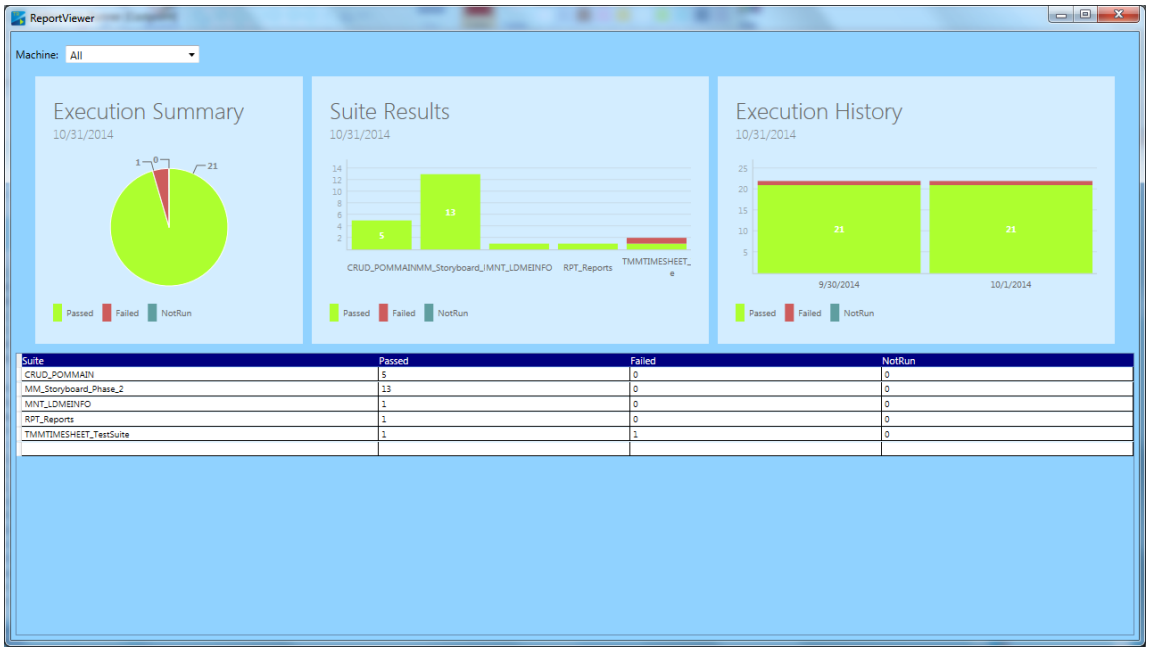


Figure 70: Report Viewer

Execution Summary – Contains the latest execution status of all test suites

Suite Results – Individual status of each test suite

Execution History – Column graph showing the trend of execution status of all test suites

Appendix A: Troubleshooting Guide

Issue: Selenium is not correctly navigating to application URL as specified on configuration file

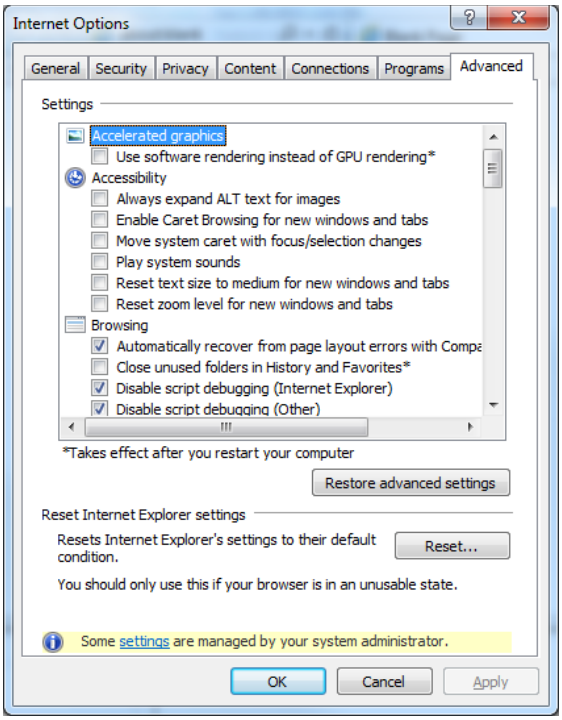
Solution 1: Test with Firefox

This should be performed to confirm that issue is browser-specific and not in selenium's core functionality. Unlike chrome and IE, Firefox does not use an independent driver service executable, and does not need any firewall exception.

If it worked, issue can be related to starting/allowing remote connections using these separate drivers (See #3).

Solution 2: Reset IE Default Settings

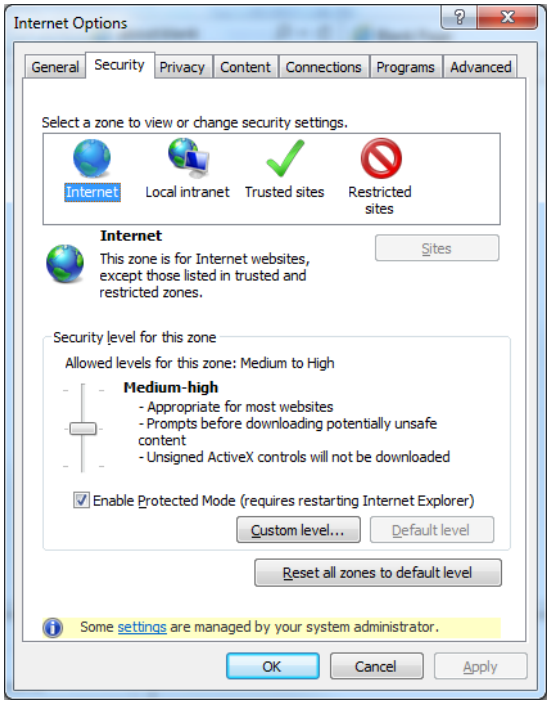
There are some installed IE plugins that needs to be disabled because they are not compatible with Selenium. To reset IE settings to default configuration, open *Internet Options* and go to *Advanced* Tab, then click on *Reset…*



Note that doing this will disable all IE user installed plugins (you need to re-enable them as needed), and reset other custom configurations to default values.

Solution 3: Set Internet Explorer Protected Mode settings for all zones

Set *Protected Mode* settings for all zones (*Internet*, *Local Intranet*, *Trusted Sites*, and *Restricted Sites*) to the same setting (either all ON or all OFF).



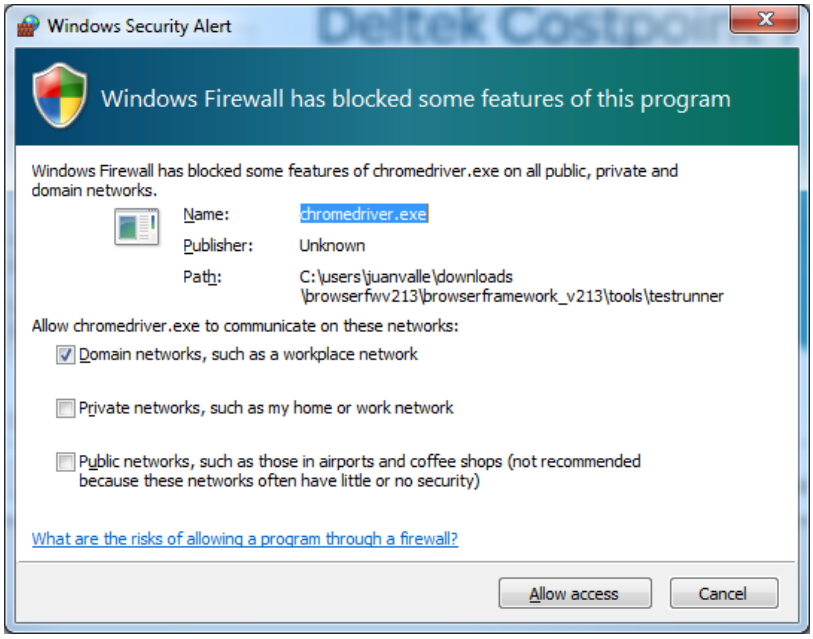
Solution 4: Set browser zoom level to 100%

Ensure that *zoom level* for browser is also at 100% (this should be set already since default is 100%).

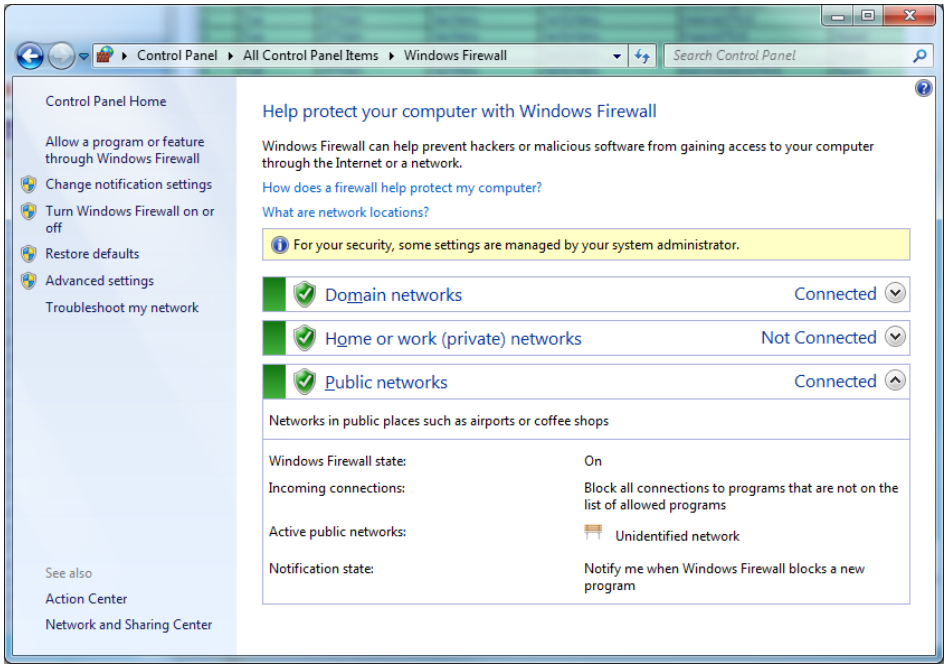
Solution 5: Allow IEDriverServer.exe/chromedriver.exe to go through the Windows Firewall

Ensure IEDriverServer.exe and chromedriver.exe are allowed to go through Windows Firewall.

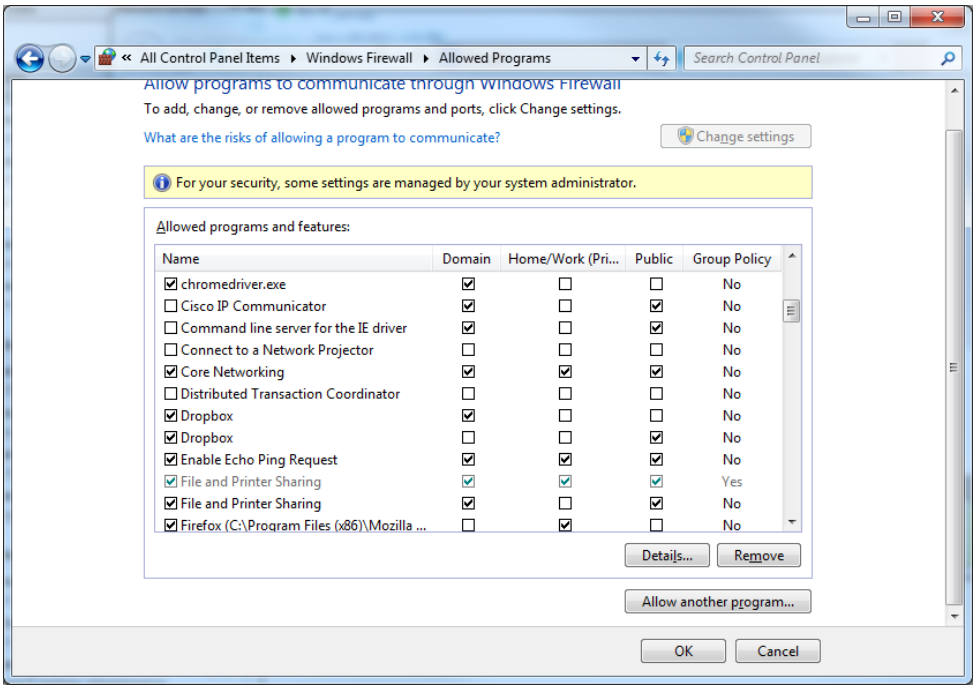
On the very first run of a test script using IE or Chrome, below dialog box should be displayed:



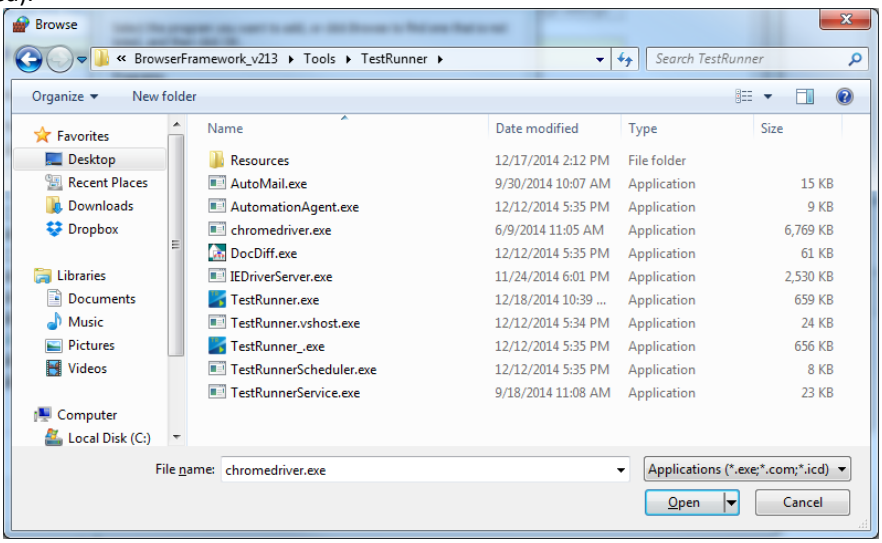
Click on *Allow Access* to create a firewall exception. To check whether a firewall exception was created, go to *Control Panel->Windows Firewall*, and click *Allow a program or feature through Windows Firewall*.



Check whether *chromedriver.exe* (or *iedriverserver.exe*) is listed on the allowed programs:



If chromedriver.exe is not in the list, click on *Allow another program…* and browse to the location of *chromedriver.exe* (or *IEDriverServer.exe*). This is under the *../Tools/TestRunner* folder (The *Tools* folder is where the *TestRunner.exe* program is located).



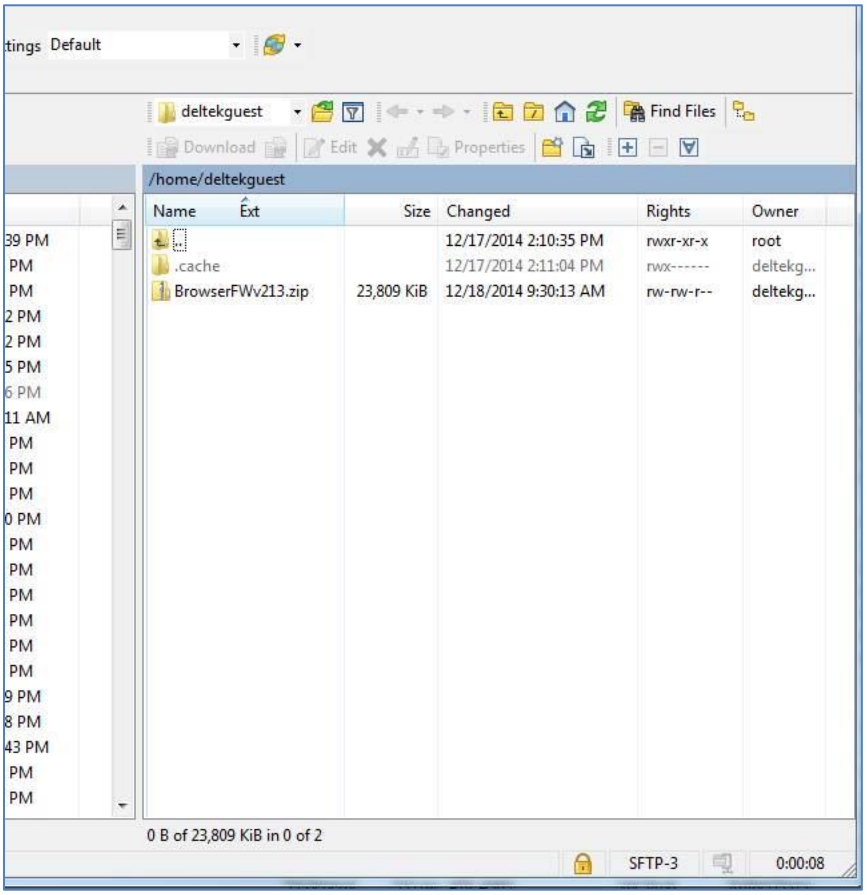
The assumptions for step (3) are:

- *Windows Firewall* is turned ON. To check, go to *Control Panel->Windows Firewall*.
- User has privileges to change *Windows Firewall* configurations. If user has restricted rights to target machine (example: no administrative privileges, under a strict domain/group policy), he cannot add/remove applications allowed to bypass the *Windows Firewall* and may need assistance from the network's administrator.

Appendix B: General Download Instructions

1. Download the *Deltek Test Runner* zip file from a Deltek FTP site

   Prerequisites:
   - 1.1 You will need a SSH/SFTP client (WinSCP can download here)
   - 1.2 Run WinSCP
   - 1.3 Create 'New Site' with the following attributes
     - File Protocol: SFTP
     - Host Name: ftpdeveloper.cloudapp.net
     - Port Number: 22
     - Username: deltekguest
     - Password: (do not save)
   - 1.4 Login
   - 1.5 Accept key
   - 1.6 Enter Password "D3lt#k1188" (do not include the double quotes)
   - 1.7 You should see filename titled TestRunner<…>.zip



2. Go to Windows Explorer, right click on the zip file (from the directory you just downloaded it to) and select “Extract All”

3. From Windows Explorer, go to: “<downloaddirectory>/TestRunner<…>/BrowserFramework/Tools”

   This is where you can start *TestRunner.exe*.

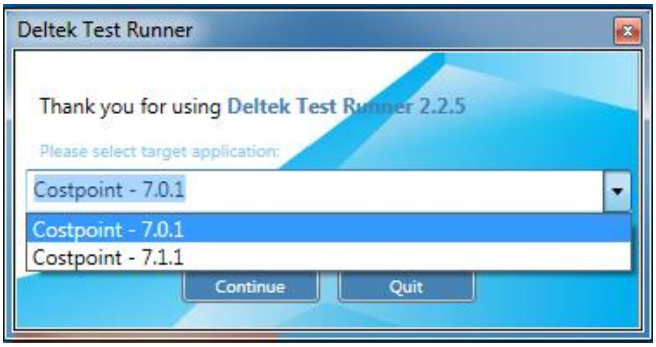
   The test scripts will be in “<downloaddirectory>/TestRunner<…>/BrowserFramework/Products/Costpoint/Test”

Appendix C: Quick Start Reference

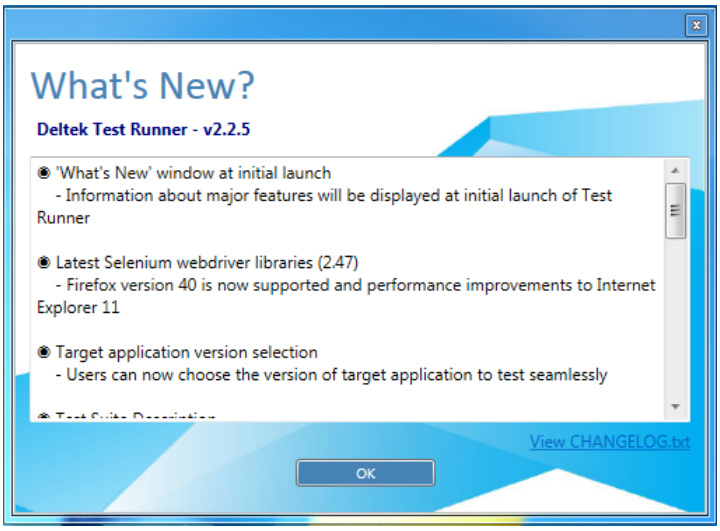
Setting your configuration

1. Open *Test Runner* by going to: "<downloaddirectory>/TestRunner<…>/BrowserFramework/Tools", and double-clicking *TestRunner.exe*

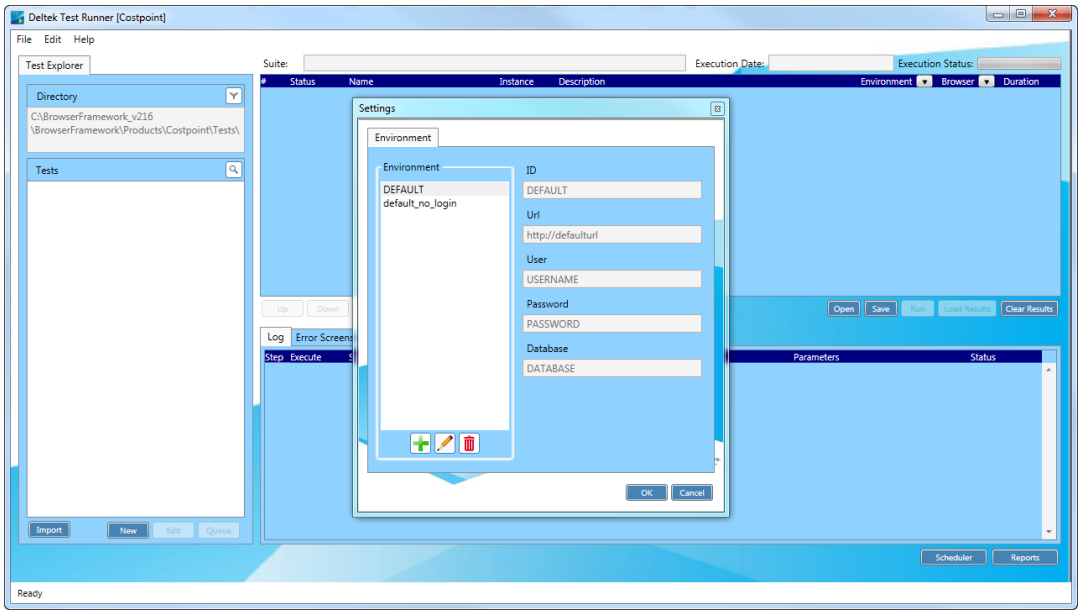
2. Click on the dropdown inside the dialog that appears and choose the application you want to test, then click CONTINUE



3. A 'What's New' dialog box showing a list of new features will appear. Click OK to continue



4. Click on the EDIT menu bar, and select SETTINGS:



5. Click the pencil icon at the bottom, and edit this, by entering the information for:

   ID = Arbitrary name that you want to call this environment
   URL = The URL that you connect to *Costpoint* in your test environment
   USER = User that you login to *Costpoint* with
   PASSWORD = Password for that user
   DATABASE = Your *Costpoint* DB that you log into

   Click OK when you are done.

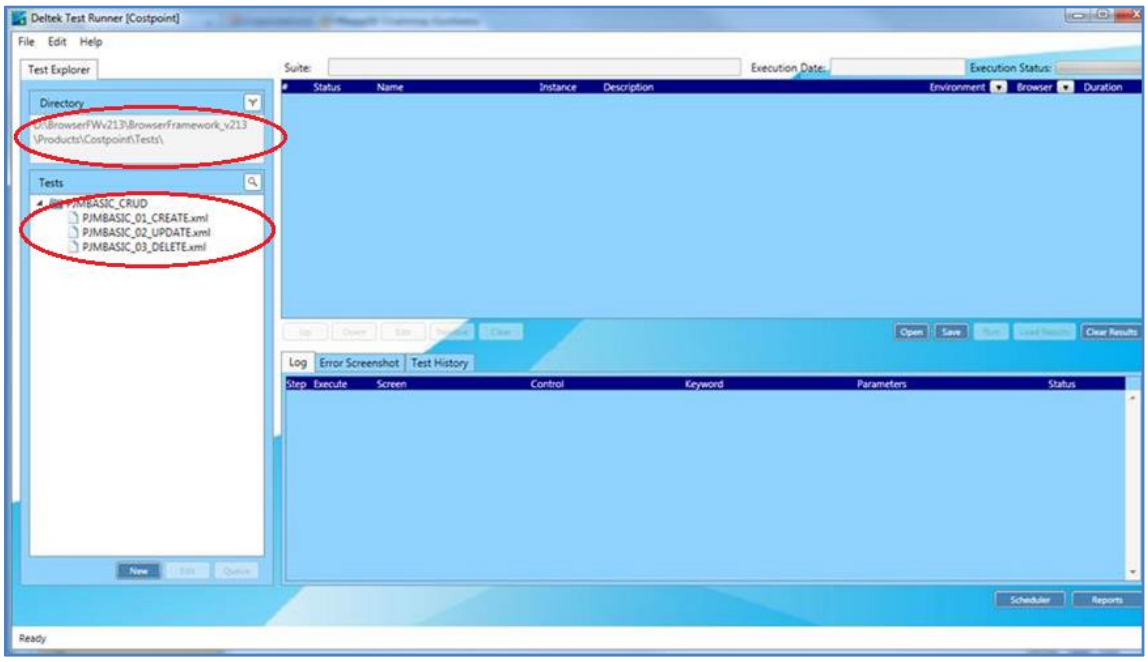
Running Tests

1. Place tests inside this directory:

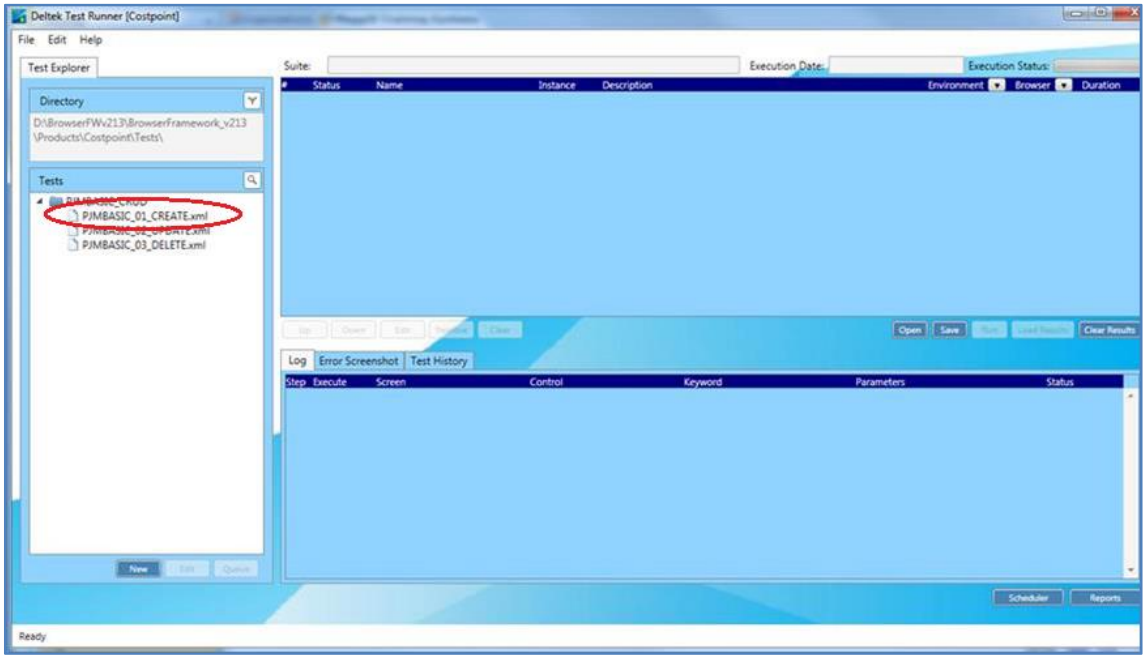
   "<downloaddirectory>/TestRunner<…>/BrowserFramework/Products/Costpoint/Test"

   You can create a subfolder/s inside this directory and move the scripts into that created subfolder/s for better management

2. Open *Test Runner* and the scripts should now be visible in the *Test Explorer*, at the left hand side of the *Test Runner* window.



3. Click on PJMBASIC_01_Create.xml in the *Test Explorer* and click the *Queue* button at the bottom. It will move this to the upper right grid called the *Test Queue*.

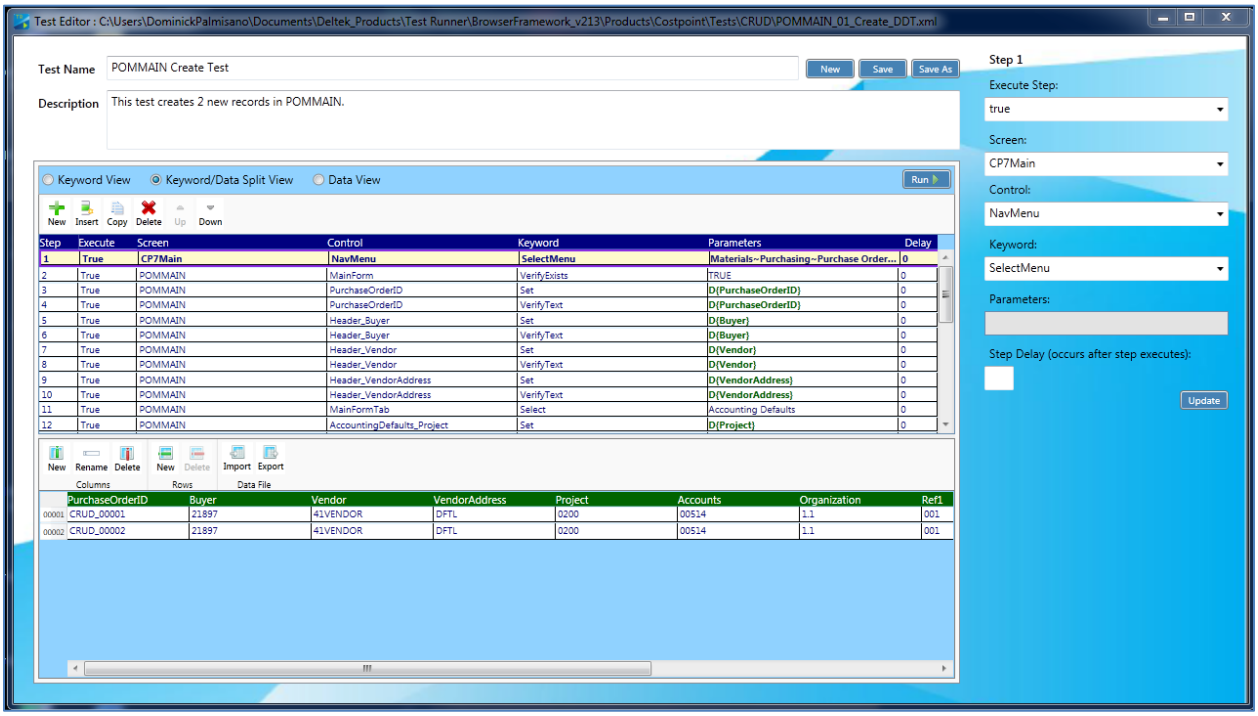


4. Select RUN at the bottom of the *Test Queue* and you should see the tool bring up *Costpoint* and run a basic script to add a project.

Notes

1. You can queue up all 3 of these (Step 5) and run them all at once if you choose.
2. Feel free to play with the editor and change the scripts in any way to fit your business workflows.
3. We will work to get more scripts for you. We have 1000s of them, but a lot are currently data-specific.

Taking a closer look at CRUD tests



The above CRUD test is an example of a *Data Driven Test* or (DDT) test. This test contains multiple data sets that will be executed within the test, as marked by the row numbers, 000001 and 000002 in the data grid view at the bottom of the screen.

Note that you have three view options, the *Keyword View*, *Keyword/Data Split View*, and just the *Data View*.

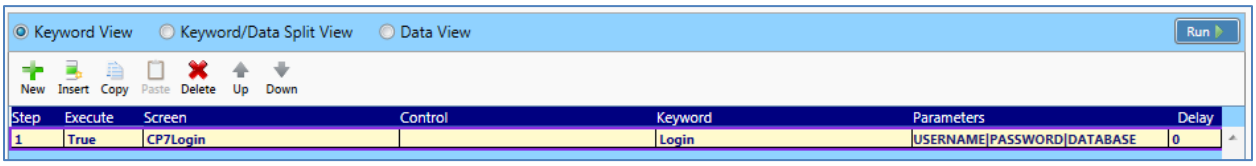
Currently the view is set to *Keyword/Data Split View*

Appendix D: Keyword Guide

Keywords are classified into *screen keywords*, *control keywords*, and *function keywords.*

Screen Keywords

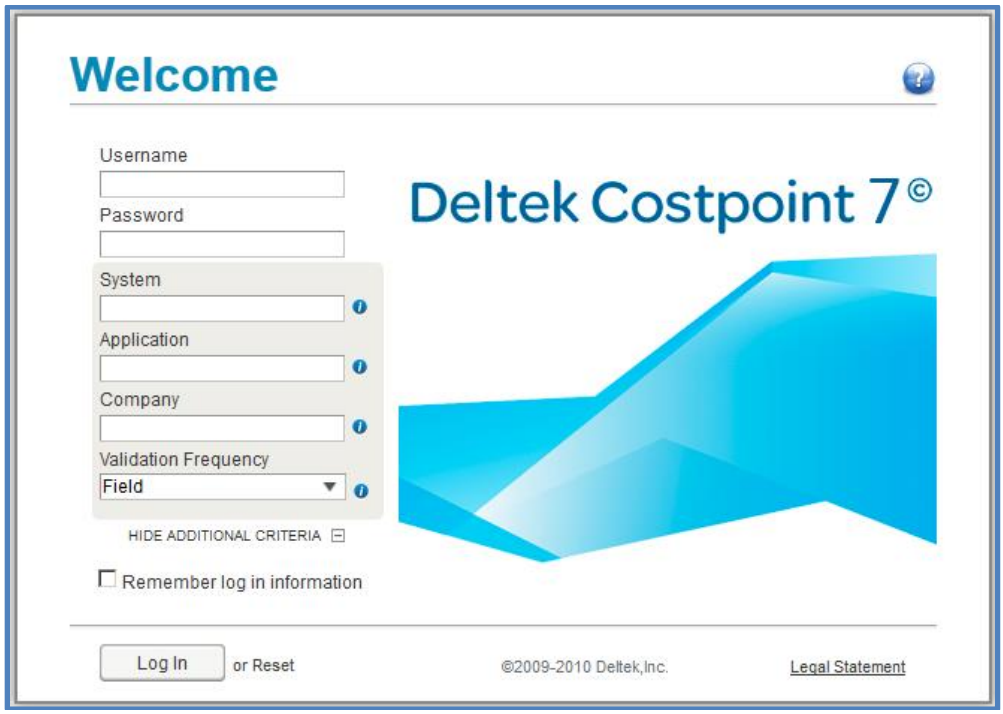
Screen keywords are keywords that are directly tied up to *screens*. To access these keywords in keyword view grid, leave the *Control* column blank.



Screen Keywords

CP7Login

SCREEN	Description	Keywords
CP7Login	Refers to the *Costpoint* login screen	Login



Costpoint 7 Login screen

KEYWORD	Description	Parameters
Login	Log into *Costpoint* using minimum required login information	UserID Password System

Dialog

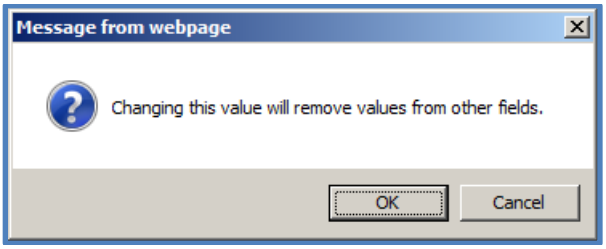
SCREEN	Description	Keywords
Dialog	Refers to dialog box raised by the operating system	FileDownload ClickOkDialogWithMessage ClickCancelDialogWithMessage

KEYWORD	Description	Parameters
FileDownload	Process file download by interacting with a combination of windows raised by the browser and OS	FileName WaitTime

KEYWORD	Description	Parameters
ClickOkDialogWithMessage	Click OK button on dialog containing expected message	ExpectedMessage

KEYWORD	Description	Parameters
ClickCancelDialogWithMessage	Click Cancel button on dialog containing expected message	ExpectedMessage

KEYWORD	Description	Parameters
ClickOkDialogIfExists	Click OK button on the dialog containing the expected message if the dialog exists. If not, it will just add a log that no alert was encountered and the execution will pass. This keyword was added to handle scenarios where dialogs are encountered in one database and does not exist in another database.	ExpectedMessage

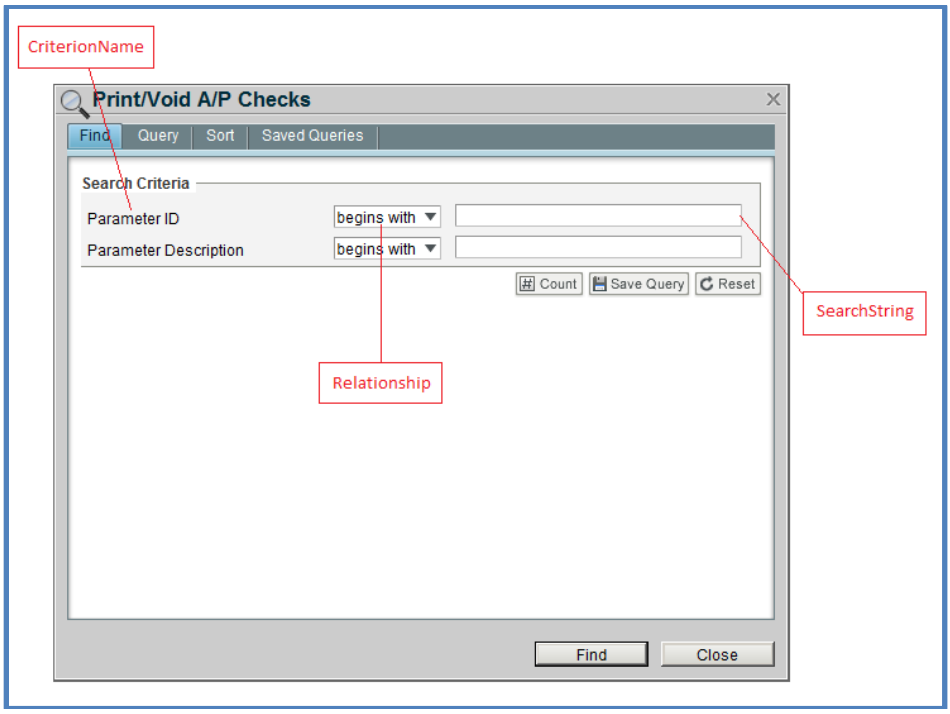


Dialog with OK and Cancel button

Query

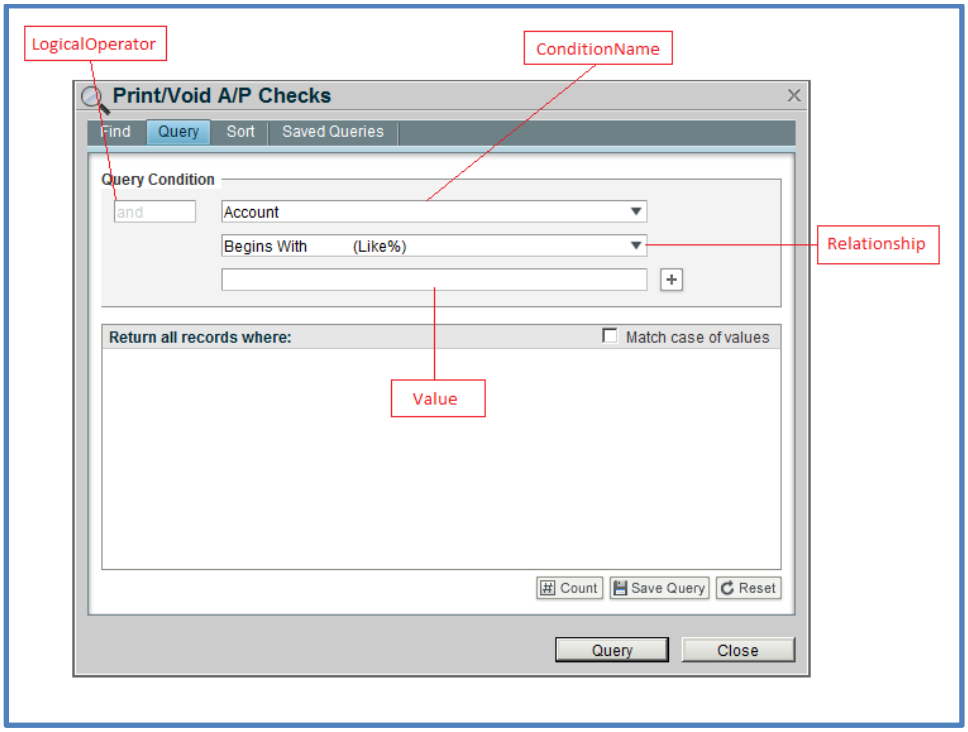
SCREEN	Description	Keywords
Query	Refers to the Query window	SetSearchCriterion AddQueryCondition

KEYWORD	Description	Parameters
SetSearchCriterion	Adds one line of search criterion to be queried using the Find tab	CriterionName Relationship SearchString



SetSearchCriterion keyword

KEYWORD	Description	Parameters
AddQueryCondition	Adds one line of condition to be queried using the Query tab	LogicalOperator ConditionName Relationship [optional] Value



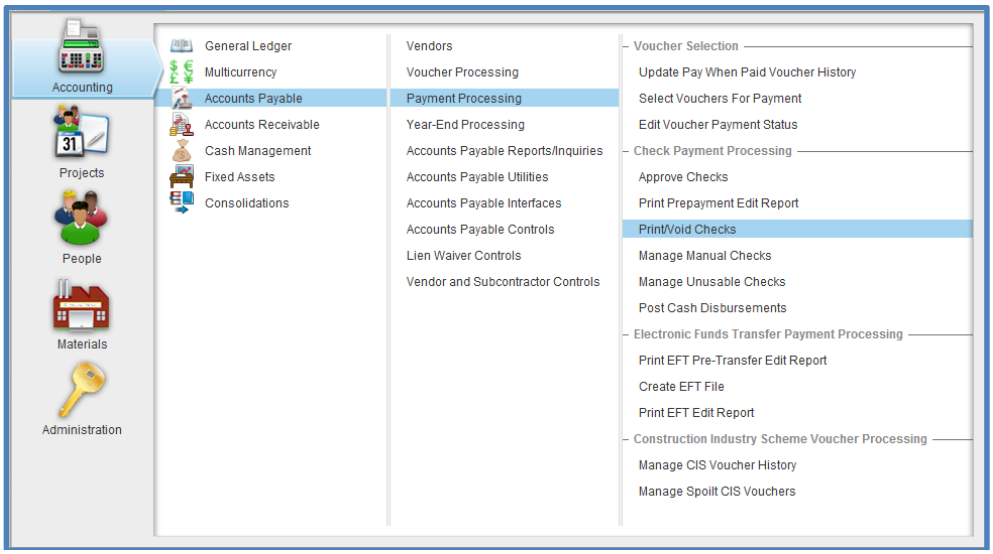
AddQueryCondition keyword

Control Keywords

Control keywords are keywords tied up with *Controls*.

NavigationMenu

CONTROL	Description	Keywords
NavigationMenu	Refers to the navigation menu control used to select target Costpoint app	SelectMenu VeryMenu



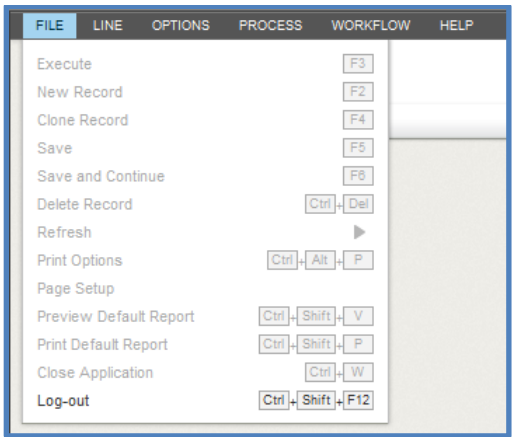
NavigationMenu

KEYWORD	Description	Parameters
SelectMenu	Select sequence of menu items existing in the navigation menu	MenuPath

KEYWORD	Description	Parameters
VerifyMenu	Verify existence/absence of input menu item sequence	MenuPath ExpectedValue

Menu

CONTROL	Description	Keywords
Menu	Refer to menu control at topmost area of Costpoint application window	SelectMenu VerifyMenu



Menu

KEYWORD	Description	Parameters
SelectMenu	Select sequence of menu items existing in the menu	MenuPath

KEYWORD	Description	Parameters
VerifyMenu	Verify existence/absence of input menu item sequence	MenuPath ExpectedValue

Toolbar

CONTROL	**Description**	**Keywords**
Toolbar	Refers to Costpoint toolbar	ClickToolbarButton VerifyToolbarButtonExist VerifyToolbarButtonReadOnly

Toolbar

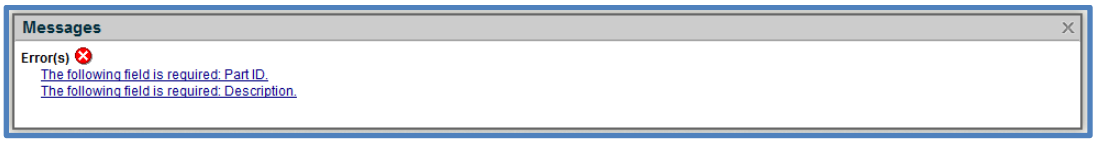
KEYWORD	Description	Parameters
ClickToolbarButton	Clicks toolbar button with caption ButtonCaption	ButtonCaption

KEYWORD	Description	Parameters
VerifyToolbarButtonExist	Verify existence/absence of toolbar button with caption ButtonCaption in toolbar	ButtonCaption ExpectedValue

KEYWORD	Description	Parameters
VerifyToolbarButtonReadOnly	Verify readonly state of toolbar button with caption ButtonCaption	ButtonCaption ExpectedValue

Messages

CONTROL	Description	Keywords
Messages	Refers to Costpoint message area displaying errors, informational messages, questions, etc.	WaitExists VerifyExists VerifyMessageExists Close ClickMessagesAreaButton



Messages

KEYWORD	Description	Parameters
WaitExists	Waits for message area to display upto *SectToWait* seconds	SecsToWait

KEYWORD	Description	Parameters
VerifyExists	Verify existence/absence of message area. Parameter can either be *TRUE* or *FALSE*	MessageExists

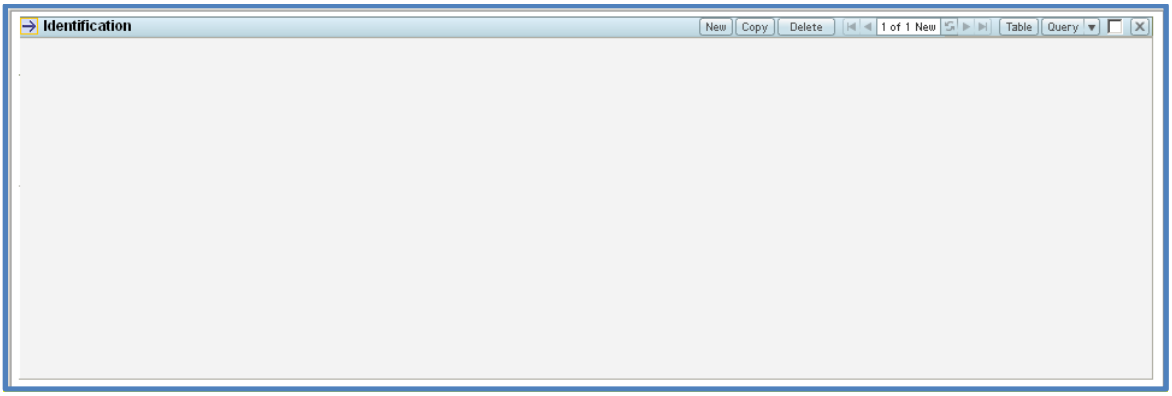
KEYWORD	Description	Parameters
VerifyMessageExists	Verify existence/absence of message area with message *ActualMessage*. *ExpectedMsgExists* can either be *TRUE* or *FALSE*	ActualMessage ExpectedMsgExists

KEYWORD	Description	Parameters
Close	Closes the message area control	

KEYWORD	Description	Parameters
ClickMessagesAreaButton	Clicks button on message area control with caption *ButtonCaption. ButtonCaption* is normally either *OK* or *Cancel*	ButtonCaption

Form

CONTROL	Description	Keywords
Form	Refers to window like container found in CP apps, characterized by colored blue heading	VerifyExists VerifyTitle ClickButton SetCheckBoxValue Close



Form

KEYWORD	Description	Parameters
VerifyExists	Verify existence/absence of form	ExpectedValue

KEYWORD	Description	Parameters
VerifyTitle	Verify text displayed on heading of form	ExpectedTitle

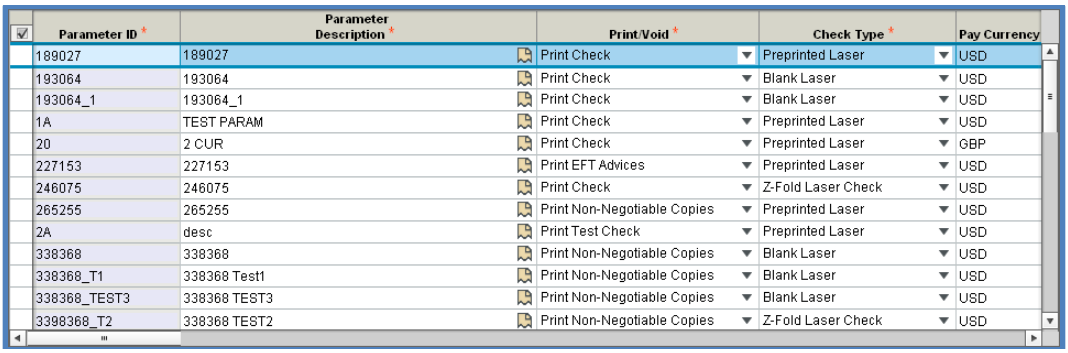
KEYWORD	Description	Parameters
ClickButton	Click a button located on heading of form	ButtonName

KEYWORD	Description	Parameters
SetCheckBoxValue	Set checked state of checkbox located on heading of form	CheckBoxName Value

KEYWORD	Description	Parameters
Close	Closes the form	

Table

CONTROL	Description	Keywords
Table	Refer to table control positioned inside a DlkForm control. This is normally accessible when the 'Table view' of the form is selected	GetTableRowWithColumnValue ClickTableCellByRowColumn SetTableCellValue VerifyTableCellValue VerifyTableRowExistWithColumnValue VerifyTableRowCount VerifyExist VerifyTableCellReadOnly VerifyTableColumnHeaders VerifyTableCellList VerifyAvailableInTableCellList GetTableRowWithMultipleColumnValues ClickTableCellButtonByRowColumn ClckTableRowHeader



Parameter ID *	Parameter Description *	Print/Void *	Check Type *	Pay Currency
189027	189027	Print Check	Preprinted Laser	USD
193064	193064	Print Check	Blank Laser	USD
193064_1	193064_1	Print Check	Blank Laser	USD
1A	TEST PARAM	Print Check	Preprinted Laser	USD
20	2 CUR	Print Check	Preprinted Laser	GBP
227153	227153	Print EFT Advices	Preprinted Laser	USD
246075	246075	Print Check	Z-Fold Laser Check	USD
265255	265255	Print Non-Negotiable Copies	Preprinted Laser	USD
2A	desc	Print Test Check	Preprinted Laser	USD
338368	338368	Print Non-Negotiable Copies	Blank Laser	USD
338368_T1	338368 Test1	Print Non-Negotiable Copies	Blank Laser	USD
338368_TEST3	338368 TEST3	Print Non-Negotiable Copies	Blank Laser	USD
3398368_T2	338368 TEST2	Print Non-Negotiable Copies	Z-Fold Laser Check	USD

Table

KEYWORD	Description	Parameters
GetTableRowWithColumnValue	Assigns row number that matches Value under the column ColumnHeader to VariableName	ColumnHeader Value VariableName

KEYWORD	Description	Parameters
ClickTableCellByRowColumn	Click table cell found in Row under the column ColumnHeader	Row ColumnHeader

KEYWORD	Description	Parameters
SetTableCellValue	Set the value of table cell found in Row under the column ColumnHeader	Row ColumnHeader Value

KEYWORD	Description	Parameters
VerifyTableCellValue	Verifies that table cell found in Row under column ColumnHeader has value that matches ExpectedValue	Row ColumnHeader ExpectedValue

KEYWORD	Description	Parameters
VerifyTableRowExistWithColumnValue	Verify existence/absence of table cell under column ColumnHeader with value equals Value	ColumnHeader Value ExpectedValue

KEYWORD	Description	Parameters
VerifyTableRowCount	Verifies count of table rows	ExpectedRowCount

KEYWORD	Description	Parameters
VerifyExist	Verify existence/absence of table control	ExpectedValue

KEYWORD	Description	Parameters
VerifyTableCellReadOnly	Verifies readonly state of table cell found in Row under column ColumnHeader	Row ColumnHeader ExpectedValue

KEYWORD	Description	Parameters
VerifyTableColumnHeaders	Verifies that table column headers matches ExpectedHeaderTexts	ExpectedHeaderTexts

KEYWORD	Description	Parameters
VerifyTableCellList	Verifies that items in table cell dropdown list found in Row under column ColumnHeader matches ExpectedValues	Row ColumnHeader ExpectedValues

KEYWORD	Description	Parameters
VerifyAvailableInTableCellList	Verifies existence/absence of SearchedValue in table cell dropdown list found in Row under column ColumnHeader	Row ColumnHeader SearchedValue ExpectedValue

KEYWORD	Description	Parameters
VerifyAvailableTableCellList	Verifies existence/absence of SearchedValue in table cell dropdown list found in Row under column ColumnHeader	Row ColumnHeader SearchedValue ExpectedValue

KEYWORD	Description	Parameters
GetTableRowWithMultipleColumnValues	Assigns row number that matches Values under the columns ColumnHeaders to VariableName	ColumnHeaders Values VariableName

KEYWORD	Description	Parameters
ClickTableCellButtonByRowColumn	Clicks table cell button captioned ButtonName found in Row under column ColumnHeader	Row ColumnHeader ButtonName

KEYWORD	Description	Parameters
ClickTableRowHeader	Clicks header of Row, selecting all cells in the row	Row

Link

CONTROL	Description	Keywords
Link	Refers link control	Click VerifyText VerifyExists

Link

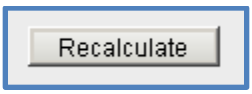
KEYWORD	Description	Parameters
Click	Click link	

KEYWORD	Description	Parameters
VerifyText	Verifies link text matches with ExpectedValue	ExpectedValue

KEYWORD	Description	Parameters
VerifyExists	Verify existence/absence of link control	ExpectedValue

Button

CONTROL	Description	Keywords
Button	Refers to button control	Click VerifyText VerifyExists VerifyReadOnly



Button

KEYWORD	Description	Parameters
Click	Click button	

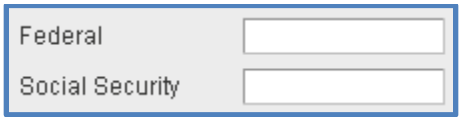
KEYWORD	Description	Parameters
VerifyText	Verifies button text matches with ExpectedValue	ExpectedValue

KEYWORD	Description	Parameters
VerifyExists	Verify existence/absence of button control	ExpectedValue

KEYWORD	Description	Parameters
VerifyReadOnly	Verify readonly state of button control	ExpectedValue

TextBox

CONTROL	**Description**	**Keywords**
TextBox	Refers to textbox control	Set VerifyText VerifyExists VerifyReadOnly ClickTextBoxButton



TextBox

KEYWORD	Description	Parameters
Set	Set value of textbox to Value	Value

KEYWORD	Description	Parameters
VerifyText	Verifies value of textbox matches with ExpectedValue	ExpectedValue

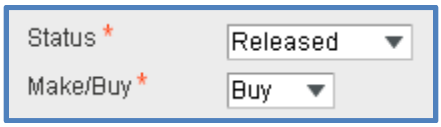
KEYWORD	Description	Parameters
VerifyExists	Verify existence/absence of textbox control	ExpectedValue

KEYWORD	Description	Parameters
VerifyReadOnly	Verify readonly state of textbox control	ExpectedValue

KEYWORD	Description	Parameters
ClickTextBoxButton	Click button inside textbox control	

ComboBox

CONTROL	Description	Keywords
ComboBox	Refers to textbox control	Select VerifyValue VerifyExists VerifyReadOnly VerifyAvailableInList VerifyList



ComboBox

KEYWORD	Description	Parameters
Select	Select Value from combobox dropdown list	Value

KEYWORD	Description	Parameters
VerifyValue	Verifies value of combobox matches with ExpectedValue	ExpectedValue

KEYWORD	Description	Parameters
VerifyExists	Verify existence/absence of combobox control	ExpectedValue

KEYWORD	Description	Parameters
VerifyReadOnly	Verify readonly state of combobox control	ExpectedValue

KEYWORD	Description	Parameters
VerifyAvailableInList	Verifies existence/absence of Item in combobox dropdown list	Item ExpectedValue

KEYWORD	Description	Parameters
VerifyList	Verifies that items in combobox dropdown list matches with ExpectedValues	ExpectedValues

CheckBox

CONTROL	Description	Keywords
CheckBox	Refers to checkbox control	Set VerifyValue VerifyExists VerifyReadOnly

CheckBox

KEYWORD	Description	Parameters
Set	Set value of checkbox to Value	Value

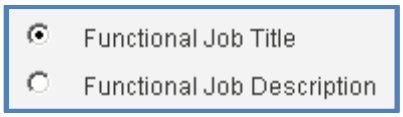
KEYWORD	Description	Parameters
VerifyValue	Verifies value of checkbox matches with ExpectedValue	ExpectedValue

KEYWORD	Description	Parameters
VerifyExists	Verify existence/absence of checkbox control	ExpectedValue

KEYWORD	Description	Parameters
VerifyReadOnly	Verify readonly state of checkbox control	ExpectedValue

RadioButton

CONTROL	Description	Keywords
RadioButton	Refers to radio button control	Select VerifyValue VerifyExists



RadioButton

KEYWORD	Description	Parameters
Select	Set selected state of radio button to Value	Value

KEYWORD	Description	Parameters
VerifyValue	Verifies value of radio button matches with ExpectedValue	ExpectedValue

KEYWORD	Description	Parameters
VerifyReadOnly	Verify readonly state of radio button control	ExpectedValue

TextArea

CONTROL	Description	Keywords
TextArea	Refers to multi-line text box (text area) control	Set VerifyText VerifyReadOnly VerifyExists AppendText

TextArea

KEYWORD	Description	Parameters
Set	Set value of textarea to Value	Value

KEYWORD	Description	Parameters
VerifyText	Verifies value of textarea matches with ExpectedValue	ExpectedValue

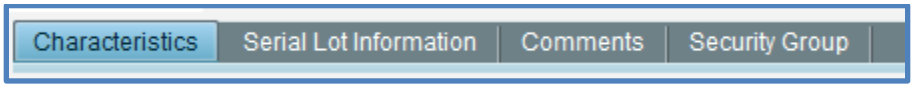
KEYWORD	Description	Parameters
VerifyReadOnly	Verify readonly state of textarea control	ExpectedValue

KEYWORD	Description	Parameters
VerifyExists	Verify existence/absence of textarea control	ExpectedValue

KEYWORD	Description	Parameters
AppendText	Poistions current cursor position within textarea control and adds text to existing text. CursorPosition can either be 'Start' or 'End'.	CursorPosition TextToAppend

Tab

CONTROL	Description	Keywords
Tab	Refers to tab control	Select VerifyExists VerifyTabs



Tab

KEYWORD	Description	Parameters
Select	Set current selected tab page to tab page with input caption	TabCaption

KEYWORD	Description	Parameters
VerifyExists	Verify existence/absence of tab control	ExpectedValue

KEYWORD	Description	Parameters
VerifyTabs	Verifies that tab pages inside tab control matches ExpectedValues. ExpectedValues items should be one single string delimited by ~	ExpectedValues

Function Keywords

Function keywords are keywords not tied up to any particular *screen* or *control*. They can be thought of as utility keywords -> keywords that provide special functionalities to help the user create better and more functional tests.

These keywords can be accessed by setting both the *screen* and *control* values to *Function*.

KEYWORD	Description	Parameters
AssignToVariable	Assigns any value to a variable for later retrieval using the *O{}* notation	VariableName Value

KEYWORD	Description	Parameters
DateFormat	Transforms an input date into user specified format such as: MM/DD/YYYY M/d/yyyyy MM-DD-YYYY	VariableName InputDate Format

KEYWORD	Description	Parameters
DateFromInputDate	Adds specific number of year, month, and/or day to input date	VariableName InputDate AdditionalYear AdditionalMonth AdditionalDay

KEYWORD	Description	Parameters
DateFromToday	Adds specific number of year, month, and/or day to current date	VariableName AdditionalYear AdditionalMonth AdditionalDay

KEYWORD	Description	Parameters
DateToday	Assigns current date to a variable	VariableName

KEYWORD	Description	Parameters
IfThenElse	Performs a value comparison test that will determine next step script will execute	VariableValue Operator ValueToTest IfGoToStep ElseGoToStep

KEYWORD	Description	Parameters
LogComment	Displays a row of comment in the Test Editor	CommentToLog

KEYWORD	Description	Parameters
SendKeys	Sends a keyboard key or combination of keys to the test browser	Keys

KEYWORD	Description	Parameters
TextConcat	Concatenates two input texts and assigns to variable	VariableName OriginalText TextToConcatenate

KEYWORD	Description	Parameters
TextReplace	Replaces all instances of input subtext in an input text with another input text, then assign resulting text to variable	VariableName InputText TextToReplace TextReplacement

KEYWORD	Description	Parameters
TextSubstring	Assigns a portion of an input text to variable starting from input index up to input length	VariableName InputText StartIndex Length

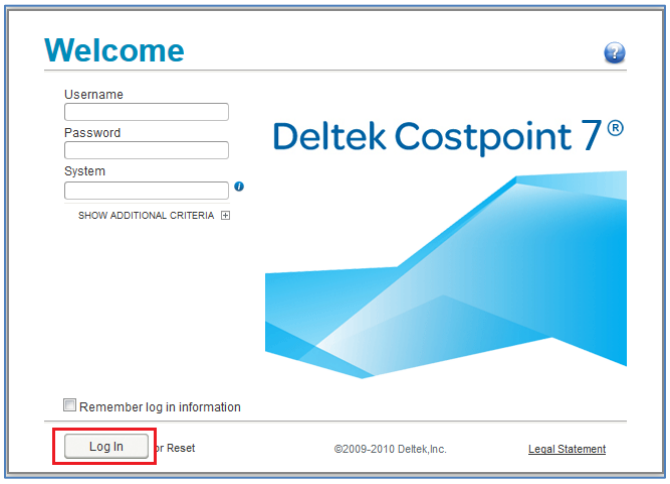
KEYWORD	Description	Parameters
TurnOffTestSteps	Skips execution of steps denoted by the ff notation: Ex: 4~5~6~7 In example, steps 4,5,6,7 will not be executed	StepsToTurnOff

Appendix E: Test Capture Limitations

1) 'Clickable' controls that disappear from screen on click

The test capture feature works with the use of injection of *javascript* (*JS*) codes to the *Document Object Model* (*DOM*). These *JS* codes are designed to identify web elements within the *DOM* that the user clicked or typed into. With buttons and other controls that 'disappear' upon user click, the *JS* code sometimes fail to retrieve the web element before it is removed from the *DOM* due to the web page transitioning to another state because of the user click.

An example is the *Login* button:



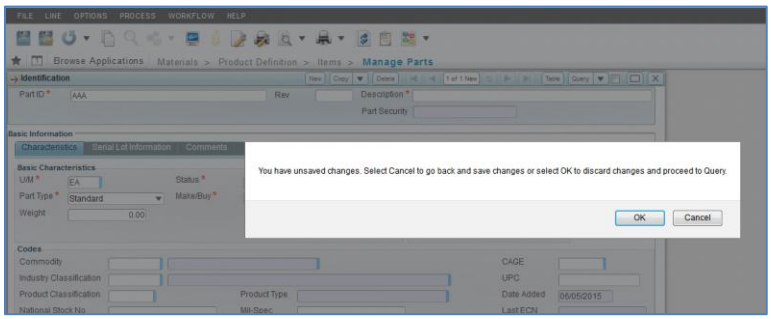
Login button

When the user clicks on the *Login* button, the page transitions from the *Login* page to the *Main* page of *Costpoint*, refreshing the browser *DOM*. This refreshed *DOM* no longer contains the *Login* button web element, thus the recorder fails to record the corresponding user action performed on the login button.

As an alternative, the user can press the **left mouse button and not release (mouse-down)** until the *Click* step is recorded in the *Script Grid*, and then **move the mouse cursor away from the control** to prevent firing of the *Click* and any attached actions programmed into the control when clicked. The user can then re-click the control to proceed with recording.

2) Unexpected dialogs

Since the test capture feature implements an engine that continuously monitors web page activity, it is essential that the browser *DOM* is always accessible. Unexpected modal or blocking dialogs prevent the tool from accessing the *DOM* and record user actions. Below is an example:

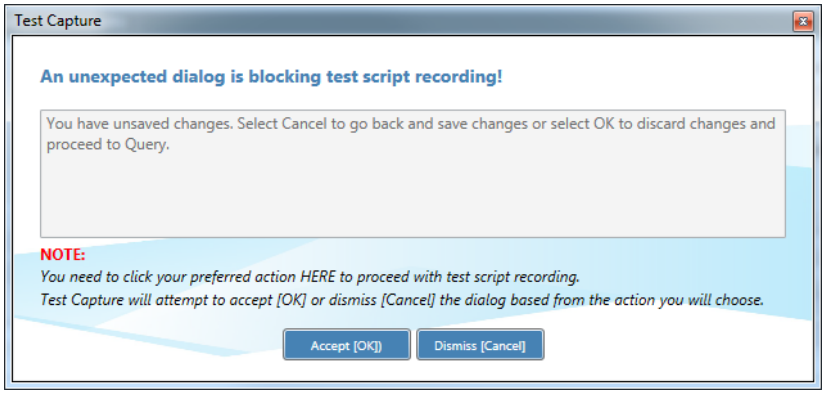


Unexpected Dialog

Some examples of controls/actions in *Costpoint* that trigger this include but are not limited to:

- Clicking on *File->Close Application* when there is a pending change to current record
- Clicking on *Query* when there is a pending change to current record

Once an unexpected dialog (*Unexpected Dialog*) is encountered, the tool will display a dialog message of its own (*Test Capture Dialog*), which mirrors the message of the unexpected dialog (*Unexpected Dialog*):

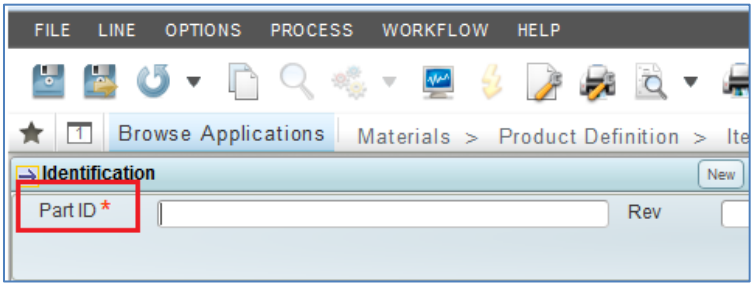


Test Capture Dialog

The user should use *Test Capture Dialog* instead of *Unexpected Dialog* to click on the desired response for the recording to proceed. Failure to follow this routine will end the test recording.

3) Label controls not supported.. yet

Currently, actions performed against *Label* controls are not yet supported for recording.



Label Control