# Open2Test Test Automation Framework for SilkTest – Tips

**Version 1.0**

**January 2010**

# TABLE OF CONTENTS

# 1 INTRODUCTION

## 1.1 Purpose

This "Tips" document provides an overview for handling frequently encountered scripting problems, as well as some valuable do's and don'ts to maximize the productivity of the Open source Test Automation Framework.

## 1.2 Scope

This document is solely for understanding and reference. There might be a need to tweak the solutions provided before implementation, depending on the actual scenario.

This document requires prior knowledge and working experience with the Open Source Test Automation Framework. For understanding the keywords and syntaxes, please refer to the Scripting Standards document.

# 2 CONDITIONS

Conditions are basically implemented when we need a specific code to be executed only when a condition is satisfied. For example, we want a 'pass' report to be sent to the log if variable 'A' is equal to variable 'B'. Otherwise, we want a 'Fail' report to be sent.

Syntax – r|condition|<expression1>;Conditional Operation;<expression2>|<true start row>:<false start row>

## 2.1 Problem Scenarios

You may encounter the following two problem scenarios during the implementation of conditions using the Open Source Test Automation Framework.

### 2.1.1 Handling Nested Conditions

At times, the flow might not be restricted to just one condition but instead require multiple conditions before code is executed. In other words, at times nested conditions might be required. In such cases, the second condition should be placed after the first condition and should fall within the range of the first condition.

The following example in Table 1 shows a simple way to do this. The first condition at line 10 has the start and end rows as 11 and 13, respectively. The second condition is placed at line 11 and has the start row as line 12, where the third condition is placed. Therefore, if the first condition is satisfied, only then will the control go to the second condition. And if the second condition is satisfied, the control would go to the third condition. Thus, during actual execution, all three conditions would have to be satisfied before the code at line 13 is executed.

Below is the simple explanation for a Condition Statement.

| Line | Automate | Action | Object | ActionValueOne |
|---|---|---|---|---|
| 10 | r | Condition | Var:A;equals;Var:B | 11;13 |
| 11 | r | Condition | Var:B;equals;Var:C | 12;13 |
| 12 | r | Condition | Var:D;Greaterthan;Var:E | 13;13 |
| 13 | r | report | Pass;Expected:Expected Value;Actual:ActualValue | |

**Table 1: Nested Conditions**

# 3 Base State

It is very important that the automation scripts developed should not have any dependency on the state of the application. They should be able to execute from any screen in the application. The importance of this is that when these scripts run in a suite and a script fails, the following script might not find the application in the home screen. In such circumstances, the second script should not fail because it did not find the application in the required state.

The best way to resolve this is to set the recovery system. The recovery system ensures that each testcase begins and ends with the application in its intended state. SilkTest refers to this intended application state as the BaseState. The recovery system allows you to run tests unattended. When your application fails, the recovery system restores the application to the BaseState, so that the rest of your tests can continue to run unattended.

We use the Set Recovery System dialog to identify the starting point of the application you are testing. SilkTest's recovery system will return your application to this BaseState: before running a testcase; during a testcase, if an error occurs; and after a testcase completes.

# 4 Message Box

Maintenance is an integral part of the automation lifecycle. Once we have developed the scripts, we have to maintain them to keep the script updated and in sync with the application.

The "message box" functionality is very handy for maintaining scripts. You can use the "MsgBox" keyword to invoke a message box with the value of the dynamic variables during the execution of the script. To some extent, this helps in pinpointing the error points during execution.

In the example in Table 3, at line 10, 'Text' property of a 'textbox' object with the textfield tag name is stored in a variable 'strName' for later use at line 12, where it is being checked to be equal to "Smith". During maintenance, we can actually see what is getting stored in the variable 'strName' by inserting a simple 'msgbox' keyword with 'strName' as the argument.

| Line | Automate | Action | Object | ActionValueOne |
|---|---|---|---|---|
| 10 | r | Storevalue | Textbox;<textfield tag name> | Textbox:strName |
| 11 | r | Msgbox | Var:strName | |
| 12 | r | Condition | Var:strName;equals;Smith | 13;13 |
| 13 | r | Report | Pass;Expected:Expected Value::Actual:ActualValue | |

**Table 2: Message Box 'msgbox' Keyword.**

# 5 EXTERNAL DATA

The data from the external data sheet is used to type, select, or verify the items present in ComboBox, ListBox, and TextBox.

**1** TextBox

Type_dt: Types the value present in the external data sheet in the current object.

**2** ListBox:

Verifylist_dt: Verifies the list of items present in the external data sheet in the current object.

**3** ComboBox:

- Select_dt: Selects an item present in the external data sheet in the current object.
- Verifylist_dt: Verifies the list of items present in the external data sheet in the current object.

Note: The data sheet path is specified in the ActionValueTwo column of the keyword script.

Here the syntax is provided for TextBox, ListBox, and ComboBox.

| Automate | Action | Object | ActionValueOne | ActionValueTwo |
|---|---|---|---|---|
| r | Perform | Textbox;<textfield tag name> | Type_dt:<Row Number>:<Column Name> | <Data Sheet Path> |
| r | Check | Listbox;<listbox tag name> | Verifylist_dt:< Row Number>:<Column Name> | <Data Sheet Path> |
| r | Perform | Combobox;<combobox tag name> | Select_dt:<Row Number>:<Column Name> | <Data Sheet Path> |
| r | Check | Combobox;<combobox tag name> | Verifylist_dt:< Row Number>:<Column Name> | <Data Sheet Path> |

**Table 3: Syntax for External Data**