



Open2Test Test Automation Framework for SilkTest - Implementation Guide

Version 1.0

January 2010

DISCLAIMER

Verbatim copying and distribution of this entire article is permitted worldwide, without royalty, in any medium, provided this notice is preserved.

TABLE OF CONTENTS

1.	Purpose of the Document.....	3
2.	Framework Implementation in SilkTest.....	4
2.1.	Necessary Files for Keyword-Driven Scripting.....	4
2.1.1	Creating a Data-Driven Script (.g.t) File.....	4
2.1.2	Settings in Data-Driven Assistant Section.....	7
2.1.3	Settings in the Keyword Driver Section.....	10
2.2	Call to Framework.....	11
2.3	Usage of Keywords.....	12
2.4	Test Results for a Keyword-Driven Script.....	13
3.	References.....	15

1. Purpose of the Document

This document provides an overview of the prerequisites and settings required for implementing the keyword-driven framework in Microfocus SilkTest.

2. Framework Implementation in SilkTest

The keyword-driven framework is an application-independent framework that performs all possible actions and verifications on an object. Hence, the code for the same object can be used across different applications.

2.1. Necessary Files for Keyword-Driven Scripting

In the keyword-driven approach the entire script is developed using keywords. The script is developed in an external spreadsheet, which is interpreted by driver script in SilkTest.

The essential files to run a keyword script include:

1. **Object Repository (.inc) File:** The file contains the window declarations of the objects in the Application Under Test (AUT).
2. **Data-Driven Script (.g.t) File:** This file contains the main settings to run a keyword-driven script.
3. **Framework (.inc) File:** Framework file contains the Framework code which is used for keyword scripting.
4. **SpreadSheet (.xls):** This file contains the keyword script to test the particular application.
5. **Datadrivetc (.inc) File:** This file is SilkTest driver code, which drives the data-driven script file.
6. **CommonFunctions (.inc) File:** This file contains the common functions that are used in the framework code, such as create file/folder or delete file/folder operations.
7. **UserDefinedFunctions (.inc) File:** This file contains user-defined functions, which may be application-specific.

SilkTest keyword scripting includes the few simple changes in the data-driven script (.g.t) file. This file contains all the settings that have to be done to run a keyword-driven test script.

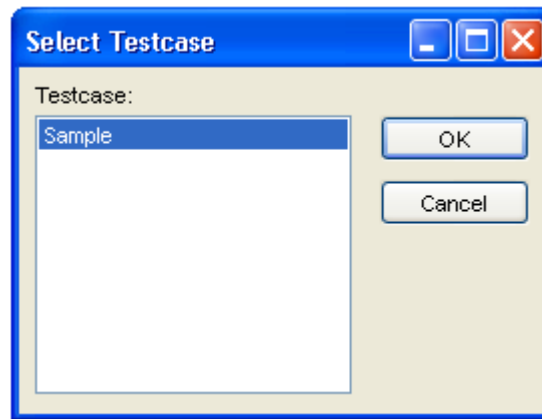
2.1.1 Creating a Data-Driven Script (.g.t) File

The following steps guide you on how to create the data-driven script file in SilkTest.

1. Create/open a 4test script file with at least one testcase.
2. Choose Tools → Data Drive Testcase or Click on “Data Drive Testcase” in the workflow bar.



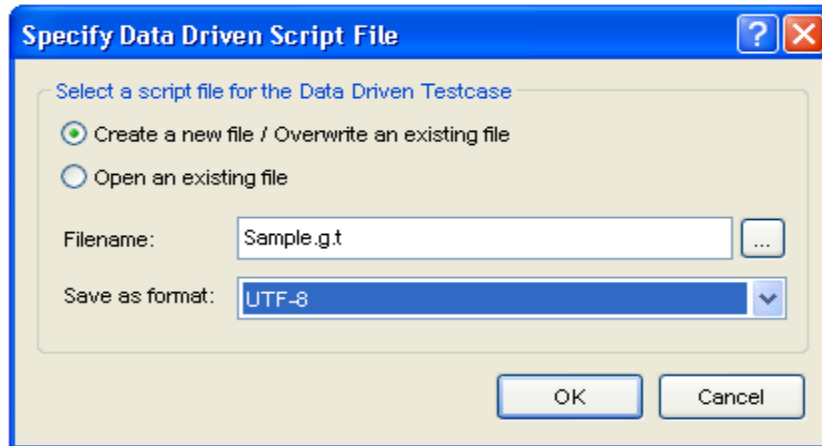
3. Select the test case name to the data drive and click “OK”.



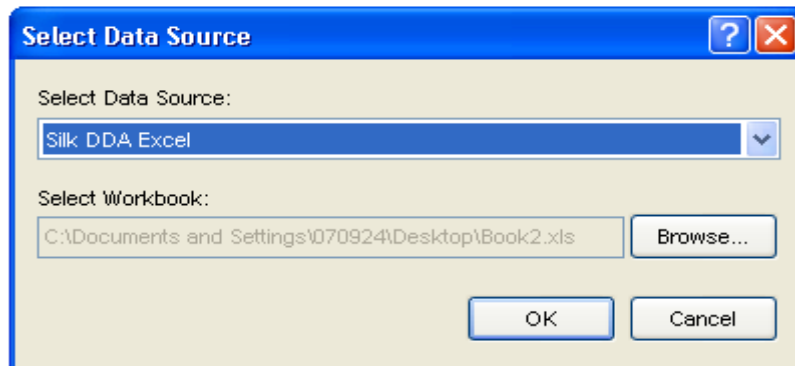
Note:

The data-driven script file can be generated only if we have a 4test script file. Let us assume that the script file is Sapmle.t file.

4. Once you select a testcase, SilkTest copies the selected testcase and creates a new data-driven testcase by adding a “DD_” prefix to the original name of the testcase.

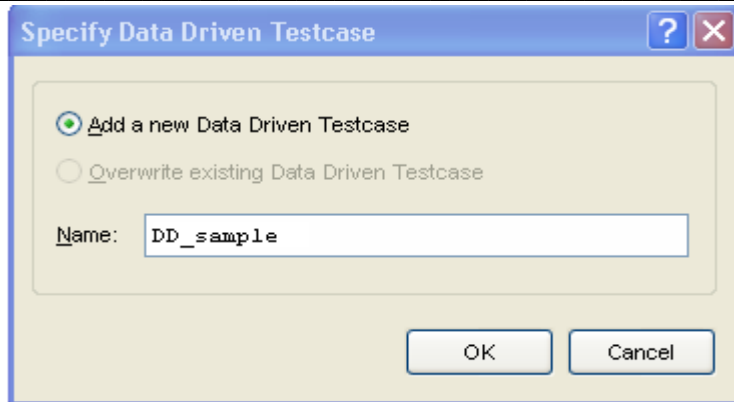


5. The Select Data Source dialog box allows you to choose either the Silk DDA Excel or the Segue DDA Excel data source. For new data-driven testcases, choose the Silk DDA Excel data source. Choose the Segue DDA Excel data source for backward compatibility. This allows existing g.t files that reference Segue DDA Excel to continue to run. Click on "OK" to continue.



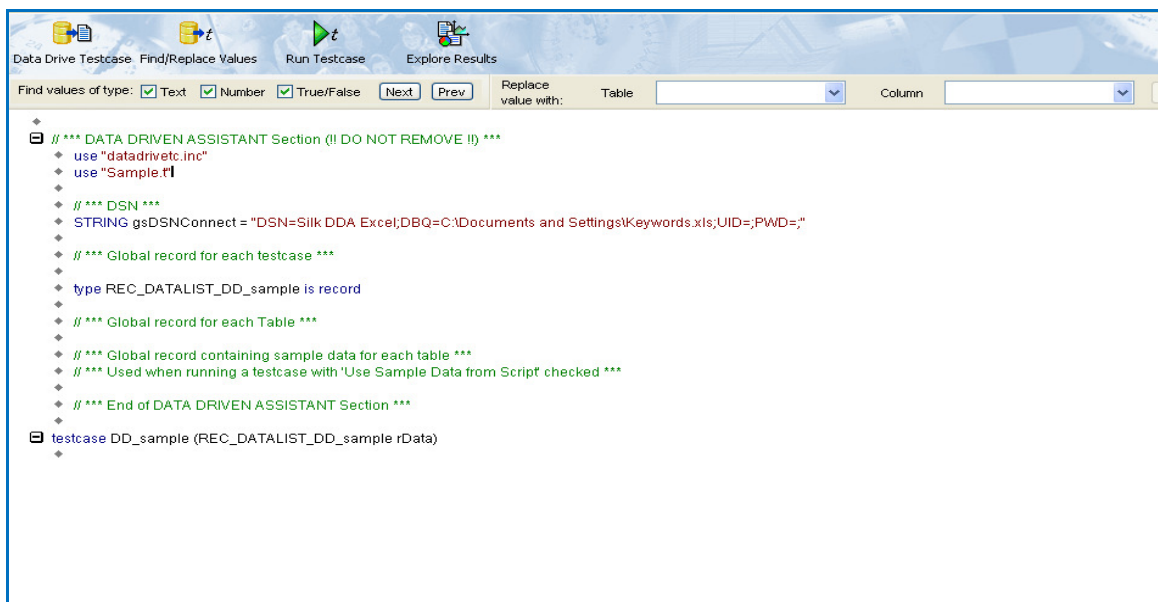
NOTE: When you installed SilkTest, the "SILK DDA EXCEL" DSN was copied to your installation computer; this is the default DSN that SilkTest uses. This DSN uses a MS EXCEL 8.0 driver and does NOT have a particular workbook (.xls file) associated with it.

6. Click on "OK" in the "Specify Data Driven Testcase" dialog.



Here SilkTest verifies that the DSN configuration is correct by connecting to the database, generates the 4Test code describing the DSN, and writes that information into the data-driven script.

7. The data-driven script file will be created with few default settings as shown below.



Data-Driven Script File

2.1.1.2 Settings in Data-Driven Assistant Section

The data-driven script file contains few lines of code automatically generated by SilkTest. The information is delivered "rolled up" (collapsed); to see the details, click on the plus sign.

1. [+] // * DATA-DRIVEN ASSISTANT Section (!! DO NOT REMOVE!!)**

The `datadrivetc.inc` files used by the original testcases and the `.t` file indicating where the testcase just came from (in this case from `Sample.t`) are displayed by default.

Some sample Settings:

```
use "datadrivetc.inc"
```

```
use "Sample.t"
```

```
use "C:\ProgramFiles\Borland\Silktest\Sample\Framework.inc" →  
framework path
```

```
use "C:\ProgramFiles\Borland\Silktest\Sample\objects.inc" →  
Object repository
```

Here we must specify the path for different files used, such as object repository file, framework file, etc.

2. [] // * DSN *****

A reference to the DSN (Data Source Name), specifying the connect string, including username and password. For example:

```
STRING$DSNConnect="DSN=SILKDDAExcel;DBQ=C:\TestExcel.xls;UID=;PW="
```

The following instructions show how to configure a machine to use a different DSN than the Silk DDA Excel default.

- a) Choose Start/Settings/Control Panel/Administrative Tools/Data Sources (ODBC).
- b) On the ODBC Data Source Administrator, click either the System DSN tab or the User DSN tab, depending on whether you want to configure this DSN for one user or for every user on this machine.
- c) Click Add.
- d) On the Create New Data Source dialog, select the driver for the data source and click Finish. If you ever need to restore SilkTest's default DSN, select the driver for Microsoft Excel Driver (*.xls).
- e) On the set-up dialog, enter the name of the data source you selected. If you ever need to restore SilkTest's default, enter Silk DDA Excel.
- f) Click OK to save your information.

3. // * Global record for each testcase *****

Each data-driven testcase takes the record as a single argument consisting of a record for each table that is used in the testcase. The record definition is automatically generated as shown here:

```
type REC_DATALIST_DD_Sample is record
```

```
    REC_Script1_ recSheet1_    //specify Sheet Name in  
excel sheet
```

```
    //Name: REC_<Testcase name>
```

```
    Field Names: Table record type with 'REC_' replaced by 'rec'
```

4. // *** Global record for each Table ***

Each table record contains the column names in the same order as in the database. Spaces in table and column names are removed. Special characters such as \$ are replaced by underscores.

Definition of record:

```
[-] type REC_Sheet1_ is record
```

```
    STRING Automate        //Automate
```

```
    STRING Action          //Action
```

```
    STRING Object          //Object
```

```
    STRING ActionValue1    //Action value1
```

```
    STRING ActionValue2    //Action value2
```

5. // *** Global record containing sample data for each table ***

```
/** Used when running a testcase with 'Use Sample Data from  
Script' checked **/
```

SilkTest writes a sample record for each table. This data is used if you opt to use sample data on the Run Testcase dialog. A value from the original testcase is inserted into the sample record.

Initialization of record:

```
[-] REC_Script1_ grTest_Sheet1_ = {...}
```

```
    [ ] "Automate"        // Automate
```

```
    [ ] "Action"          // Action
```

```
    [ ] "Object"          // Object
```

```
    [ ] "actionvalue1"    // Actionvalue1
```

```
[ ] "actionvalue2" //Actionvalue2
```

2.1.3 Settings in the Keyword Driver Section

The following sample settings need to be done in the keyword driver section of the data-driven script file.

Settings in the keyword driver section are done under the testcase generated by SilkTest.

Some sample Settings:

```
[-] testcase DD_Sample (REC_DATALIST_DD_Sample rData)

    ARRAY [50] OF Anytype aCellData

    ARRAY [50] OF Anytype bCellData

    ARRAY [50] OF Anytype cCellData

    ARRAY [50] OF Anytype dCellData

    ARRAY [50] OF Anytype eCellData

    //Previously declared records

    aCellData = {rData.recSheet1_.Automate}           //1st column

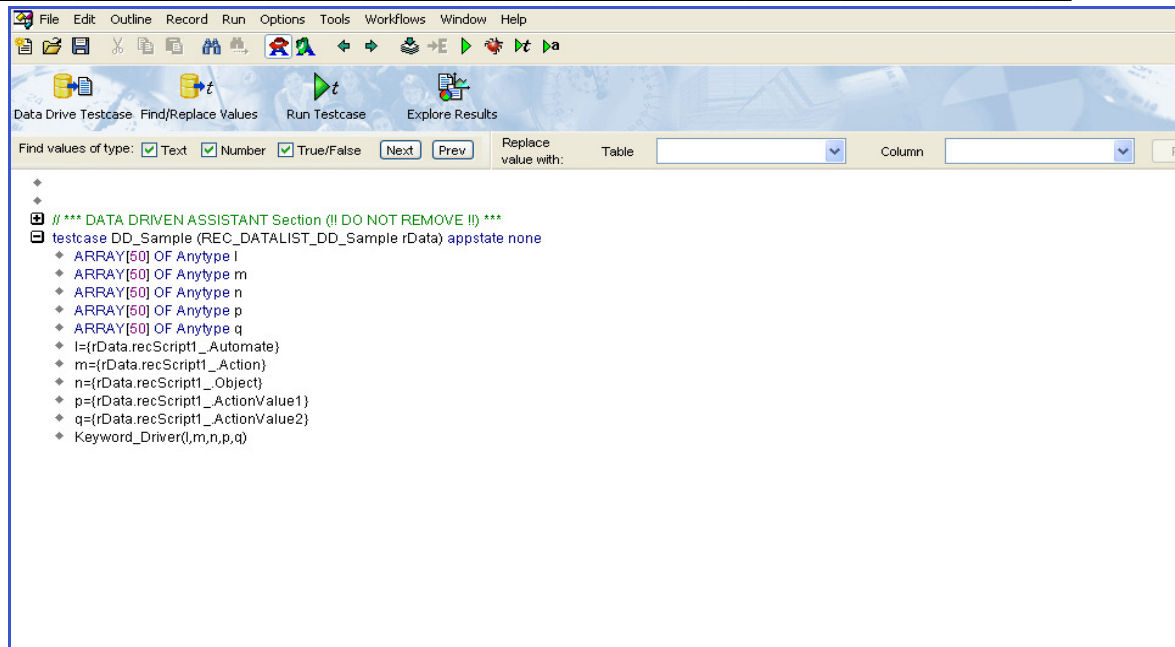
    bCellData = {rData.recSheet1_.Action}           // 2nd column
```

```
cCellData = {rData.recSheet1_.Object} // 3rd column  
  
dCellData = {rData.recSheet1_.Actionvalue1} // 4th column  
  
eCellData = {rData.recSheet1_.ActionValue2} // 5th column  
  
Keyword_Driver(aCellData, bCellData, cCellData, dCellData,  
eCellData) → //Call to Framework
```

2.2 Call to Framework

The call to `Keyword_Driver` (Parameter1, Parameter2, and so on) needs to be specified in the Keyword Driver section of the data-driven script (.g.t) file as shown below.

This calls the driver function present in the framework file associated with the test and performs the actions by interpreting the keywords specified in the keyword script file.



Call to Framework

Note:

Before calling the framework we have to make sure that we have specified the path of the framework file in

- CommonFunctions file(.inc)
- UserDefinedFunctions file (.inc)

The path for the above two files must be specified in the framework.

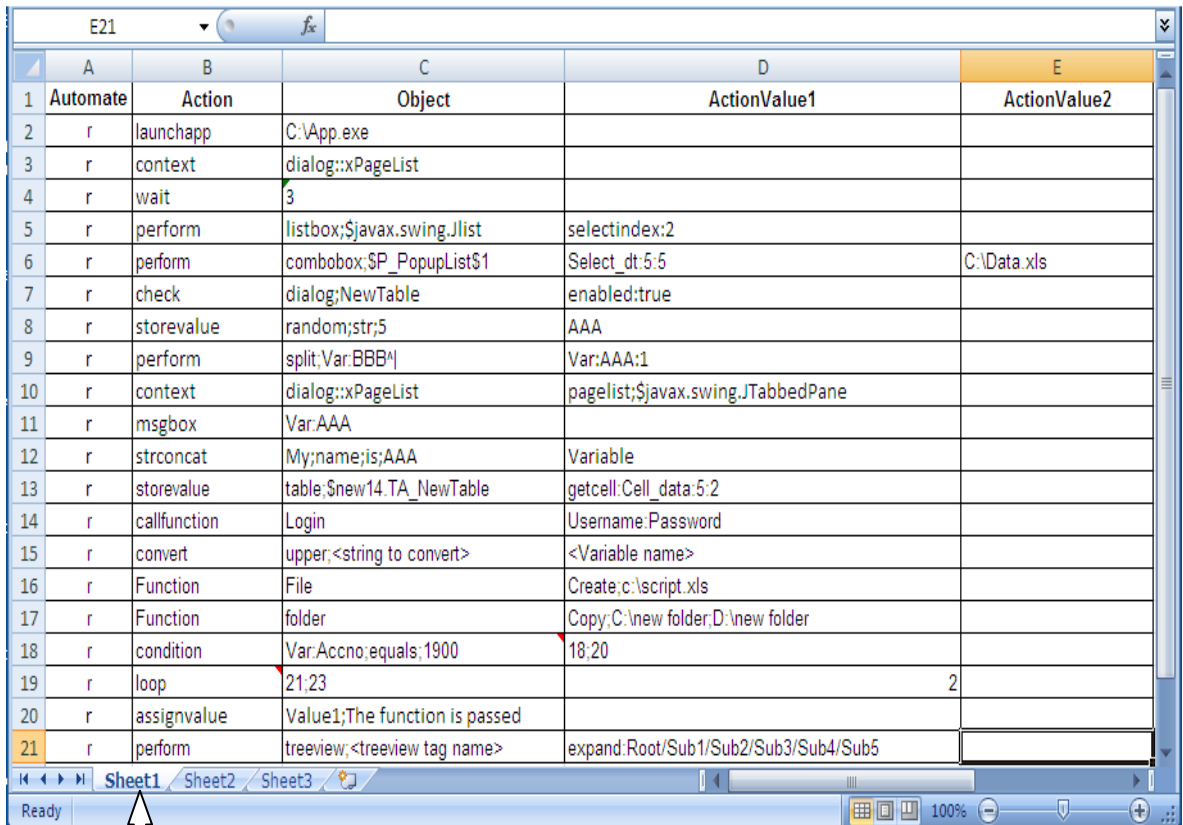


Details for specifying the path will be discussed in the framework description document.

2.3 Usage of Keywords

The keywords should be entered in the keyword script sheet according to the syntax.

Below is an example of keyword-driven scripting:



	A	B	C	D	E
1	Automate	Action	Object	ActionValue1	ActionValue2
2	r	launchapp	C:\App.exe		
3	r	context	dialog::xPageList		
4	r	wait	3		
5	r	perform	listbox;\$javax.swing.Jlist	selectindex:2	
6	r	perform	combobox;\$P_PopupList\$1	Select_dt.5:5	C:\Data.xls
7	r	check	dialog;NewTable	enabled:true	
8	r	storevalue	random;str;5	AAA	
9	r	perform	split;Var.BBB^]	Var:AAA:1	
10	r	context	dialog::xPageList	pagelist;\$javax.swing.JTabbedPane	
11	r	msgbox	Var.AAA		
12	r	strconcat	My;name;is;AAA	Variable	
13	r	storevalue	table;\$new14.TA_NewTable	getcell.Cell_data.5:2	
14	r	callfunction	Login	Username:Password	
15	r	convert	upper;<string to convert>	<Variable name>	
16	r	Function	File	Create:c:\script.xls	
17	r	Function	folder	Copy:C:\new folder;D:\new folder	
18	r	condition	Var.Accno;equals;1900	18:20	
19	r	loop	21:23		2
20	r	assignvalue	Value1;The function is passed		
21	r	perform	treeview;<treeview tag name>	expand:Root/Sub1/Sub2/Sub3/Sub4/Sub5	

Using the Keyword in a Keyword Script

“Sheet1” is the name of the sheet as shown above. This is used while scripting in .g.t file. The name “sheet1” can be renamed, but the same name must be updated in the data-driven script (.g.t) file.

It is used under the “Global record for each testcase” section in (.g.t) file.

2.4 Test Results for a Keyword-Driven Script

Test execution results can be viewed and analyzed as soon as the run session ends in the Results (.res) file. To access the test results, click on the Explore Results icon on the Data Driven Work Flow Bar and choose the file. The results window will be displayed.

The results file also provides information on:

- Machine on which the script run
- Start date and time
- Elapsed time
- Number of tests passed with percentage of Passed
- Number of tests failed with percentage of Failed
- Total number of tests run, as well as the number of Errors and Warnings

```
Script todupd.g.t - 2 errors
Machine: (local)
Started: 11:46:37AM on 25-Nov-2009
Elapsed: 0:00:09
Passed: 54 tests (98%)
Failed: 1 test (2%)
Totals: 55 tests, 2 errors, 0 warnings

Testcase DD_todupd {{{NULL,"context","window:TestApplication",NULL,NULL}}} - Passed
Testcase DD_todupd {{{NULL,"wait","3",NULL,NULL}}} - Passed
Testcase DD_todupd {{{NULL,"perform","window:TestApplication","Activated",NULL}}} - Passed
Testcase DD_todupd {{{NULL,"wait","3",NULL,NULL}}} - Passed
Testcase DD_todupd {{{NULL,"perform","window:TestApplication","minimized",NULL}}} - Passed
Testcase DD_todupd {{{NULL,"wait","3",NULL,NULL}}} - Passed
Testcase DD_todupd {{{NULL,"perform","window:TestApplication","Restored",NULL}}} - Passed
Testcase DD_todupd {{{NULL,"wait","3",NULL,NULL}}} - Passed
Testcase DD_todupd {{{NULL,"perform","window:TestApplication","maximized",NULL}}} - Passed
Testcase DD_todupd {{{NULL,"wait","3",NULL,NULL}}} - Passed
Testcase DD_todupd {{{NULL,"perform","window:TestApplication","closed",NULL}}} - Passed
Testcase DD_todupd {{{NULL,NULL,NULL,NULL,NULL}}} - Passed
Testcase DD_todupd {{{NULL,"context","dialog:JCheckBox",NULL,NULL}}} - Passed
Testcase DD_todupd {{{NULL,"wait","3",NULL,NULL}}} - Passed
Testcase DD_todupd {{{NULL,"perform","dialog:JCheckBox","closed",NULL}}} - Passed
Testcase DD_todupd {{{NULL,"wait","3",NULL,NULL}}} - Passed
Testcase DD_todupd {{{NULL,"storevalue","checkbox;$javax.swing.JCheckBox[1]","getstate:Test",NULL}}} - Passed
Testcase DD_todupd {{{NULL,"wait","3",NULL,NULL}}} - Passed
Testcase DD_todupd {{{NULL,"msgbox","Var:Test",NULL,NULL}}} - Passed
Testcase DD_todupd {{{NULL,"check","checkbox;$javax.swing.JCheckBox[1]","enabled:true/false",NULL}}} - Passed
Testcase DD_todupd {{{NULL,"wait","3",NULL,NULL}}} - Passed
Testcase DD_todupd {{{NULL,"check","checkbox;$javax.swing.JCheckBox[1]","enabled:false/true",NULL}}} - Passed
Testcase DD_todupd {{{NULL,"wait","3",NULL,NULL}}} - Passed
Testcase DD_todupd {{{NULL,"check","checkbox;$javax.swing.JCheckBox[1]","exist:true/false",NULL}}} - Passed
Testcase DD_todupd {{{NULL,"wait","3",NULL,NULL}}} - Passed
```

Results in SilkTest

NOTE: If check keyword or export result is used in the keyword script, the results will be generated in the Excel sheet with a special format.

For more details on this, refer to the "Silk test Quick Start Guidelines" document.

3. References

Microfocus SilkTest Help document.

COPYRIGHT

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.