1. Introduction to Win Runner.
2. Recording test in analog and context sensitive mode
3. Synchronizing test
4. Checking GUI Objects
5. Checking Bitmap Objects
6. Test with TSL
7. Creating data driven test
8. Manual Testing

Experiment No.: 1

Title: Introduction to winrunner.

Objective: Student should be able to

- Describes the benefits of automated testing
- Understand the WinRunner testing process
- Work with WinRunner user interface

Theory:

Understanding the Testing Process

The WinRunner testing process consists of 6 main phases:

1 Teaching WinRunner the objects in your application

WinRunner must learn to recognize the objects in your application in order to run tests. The preferred way to teach WinRunner your objects depends on the GUI map mode you select. The two GUI map modes are described in detail in subsequent lessons.

2 Creating additional test scripts that test your application's functionality

WinRunner writes scripts automatically when you record actions on your application, or you can program directly in Mercury Interactive's Test Script Language (TSL).

3 Debugging the tests

You debug the tests to check that they operate smoothly and without interruption.

4 Running the tests on a new version of the application

You run the tests on a new version of the application in order to check the application's behavior.

5 Examining the test results

You examine the test results to pinpoint defects in the application.

6 Reporting defects

If you have the TestDirector 7.0i, the Web Defect Manager (TestDirector 6.0), or the Remote Defect Reporter (TestDirector 6.0), you can report any defects to a database. The Web Defect Manager and the Remote Defect Reporter are included in TestDirector, Mercury Interactive's software test management tool.

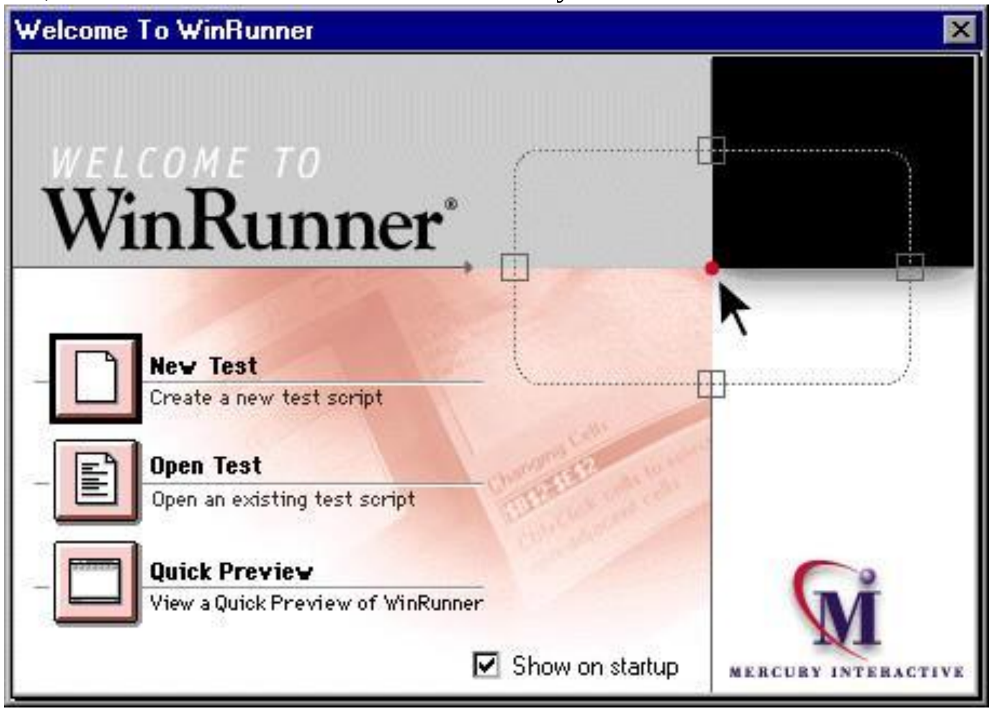
Exploring the WinRunner Window

Before you begin creating tests, you should familiarize yourself with the WinRunner main window.

To start WinRunner:

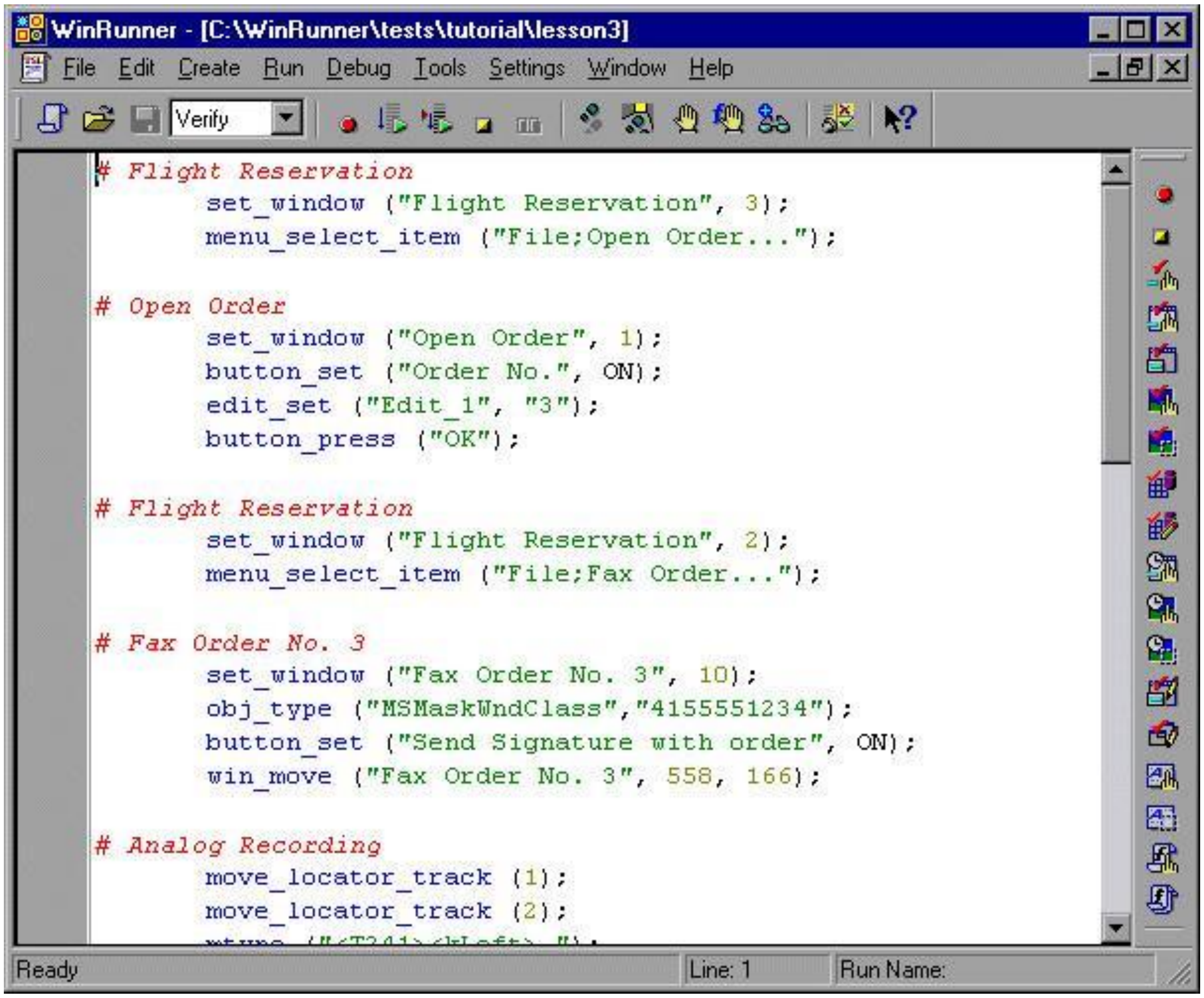
Choose **Programs** > **WinRunner** > **WinRunner** on the **Start** menu.

The first time you start WinRunner, the Welcome to WinRunner window opens. From the welcome window you can create a new test, open an existing test, or view an overview of WinRunner in your default browser.



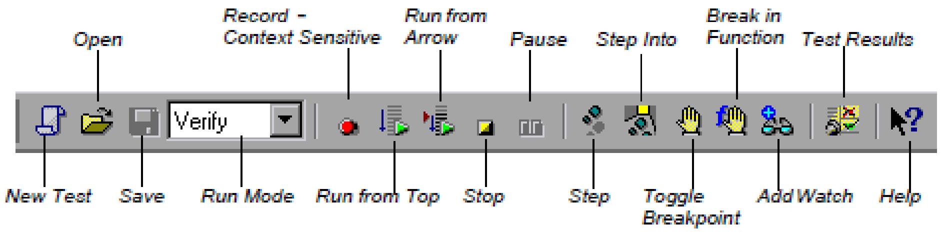
The first time you select one of these options, the WinRunner main screen opens with the "What's New in WinRunner" section of the help file on top. If you do not want the welcome window to appear the next time you start WinRunner, clear the **Show on startup** check box.

Each test you create or run is displayed by WinRunner in a test window. You can open many tests at one time.


WinRunner - [C:\WinRunner\tests\tutorial\lesson3]
File Edit Create Run Debug Tools Settings Window Help
Verify
Ready
Line: 1
Run Name:

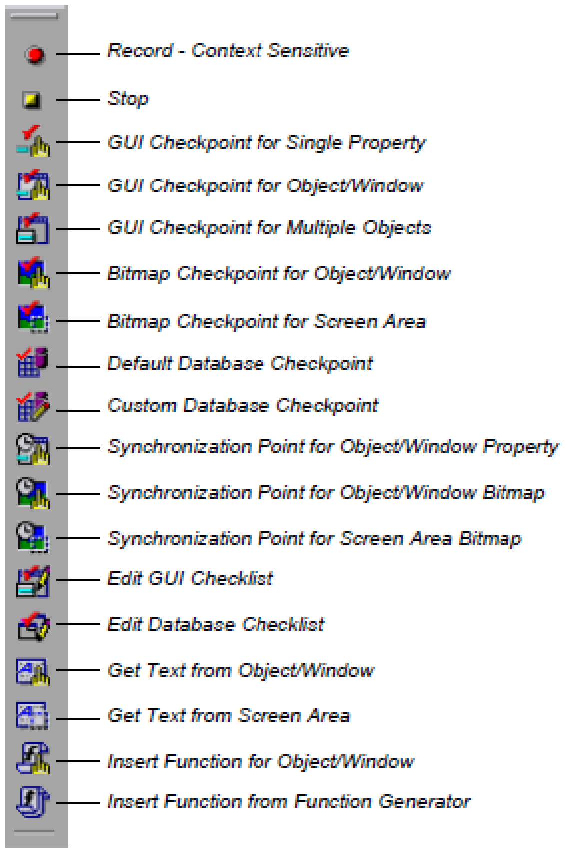
```
# Flight Reservation
        set_window ("Flight Reservation", 3);
        menu_select_item ("File;Open Order...");

# Open Order
        set_window ("Open Order", 1);
        button_set ("Order No.", ON);
        edit_set ("Edit_1", "3");
        button_press ("OK");

# Flight Reservation
        set_window ("Flight Reservation", 2);
        menu_select_item ("File;Fax Order...");

# Fax Order No. 3
        set_window ("Fax Order No. 3", 10);
        obj_type ("MSMaskWndClass","4155551234");
        button_set ("Send Signature with order", ON);
        win_move ("Fax Order No. 3", 558, 166);

# Analog Recording
        move_locator_track (1);
        move_locator_track (2);
```

The *Standard toolbar* provides easy access to frequently performed tasks, such as opening, executing, and saving tests, and viewing test results



The *User toolbar* displays the tools you frequently use to create test scripts. By default, the User toolbar is hidden.

To display the User toolbar choose **Window** > **User Toolbar**. When you create tests, you can minimize the WinRunner window and work exclusively from the toolbar.


Record - Context Sensitive
Stop
GUI Checkpoint for Single Property
GUI Checkpoint for Object/Window
GUI Checkpoint for Multiple Objects
Bitmap Checkpoint for Object/Window
Bitmap Checkpoint for Screen Area
Default Database Checkpoint
Custom Database Checkpoint
Synchronization Point for Object/Window Property
Synchronization Point for Object/Window Bitmap
Synchronization Point for Screen Area Bitmap
Edit GUI Checklist
Edit Database Checklist
Get Text from Object/Window
Get Text from Screen Area
Insert Function for Object/Window
Insert Function from Function Generator

The User toolbar is customizable. You choose to add or remove buttons using the **Settings > Customize User Toolbar** menu option. When you re-open WinRunner, the User toolbar appears as it was when you last closed it.

Experiment No. :2

Title: Recording test in analog and context sensitive mode

Objective: Student should be able to

- Describes Context Sensitive and Analog record modes
- Record a test script
- Read the test script
- Run the recorded test and analyze the results

Theory:

Choosing a Record Mode

By recording, you can quickly create automated test scripts. You work with your application as usual, clicking objects with the mouse and entering keyboard input.

WinRunner records your operations and generates statements in TSL, Mercury Interactive's Test Script Language. These statements appear as a script in a WinRunner test window.

Before you begin recording a test, you should plan the main stages of the test and select the appropriate record mode. Two record modes are available: Context Sensitive and Analog.

Context Sensitive

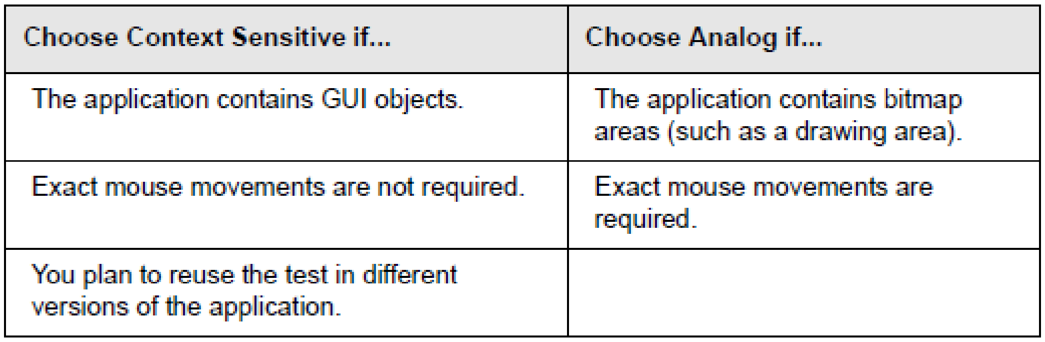
Context Sensitive mode records your operations in terms of the GUI objects in your application. WinRunner identifies each object you click (such as a window, menu, list, or button), and the type of operation you perform (such as press, enable, move, or select).

For example, if you record a mouse click on the **OK** button in the Flight Reservation Login window, WinRunner records the following TSL statement in your test script:

```
button_press ("OK");
```

When you run the script, WinRunner reads the command, looks for the **OK**
button, and presses it.

When choosing a record mode, consider the following points



Choose Context Sensitive if...	Choose Analog if...
The application contains GUI objects.	The application contains bitmap areas (such as a drawing area).
Exact mouse movements are not required.	Exact mouse movements are required.
You plan to reuse the test in different versions of the application.	

If you are testing an application that contains both GUI objects and bitmap areas, you can switch between modes as you record.

Recording a Context Sensitive Test

In this exercise you will create a script that tests the process of opening an order in the Flight Reservation application. You will create the script by recording in Context Sensitive mode.

1 Start WinRunner.

2 Open a new test.

3 Start the Flight Reservation application and log in.

4 Start recording in Context Sensitive mode.

5 Open order #3.

6 Stop recording.

7 Save the test.

Recording in Analog Mode

In this exercise you will test the process of sending a fax. You will start recording in Context Sensitive mode, switch to Analog mode in order to add a signature to the fax, and then switch back to Context Sensitive mode

1 Open the Fax Order form and fill in a fax number.

2 Select the Send Signature with Order check box.

3 Sign the fax again in Analog mode.

4 Stop Recording.

5 Save the test.

Running the Test

You are now ready to run your recorded test script and to analyze the test results. WinRunner provides three modes for running tests. You select a mode from the toolbar.

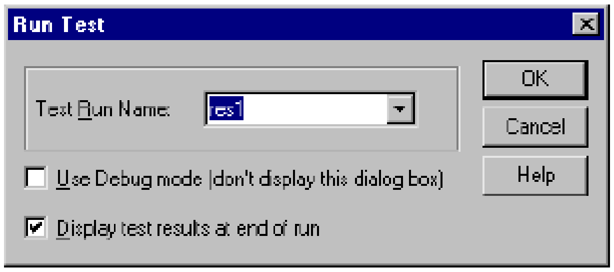
- Use *Verify mode* when running a test to check the behavior of your application, and when you want to save the test results.
- Use *Debug mode* when you want to check that the test script runs smoothly without errors in syntax. See Lesson 7 for more information.
- Use *Update mode* when you want to create new expected results for a GUI checkpoint or bitmap checkpoint. See Lessons 5 and 6 for more information

To run the test:

1 Check that WinRunner and the main window of the Flight Reservation application are open on
your desktop.
2 Make sure that the *saved* test window is active in WinRunner.
3 Make sure the main window of the Flight Reservation application is active.

4 Make sure that Verify mode is selected in the toolbar.

5 Choose Run from Top.
Choose **Run** > **Run from Top** or click the **Run from Top** button. The **Run Test** dialog box opens.



Run Test
Test Run Name: res1
OK
Cancel
Help
Use Debug mode (don't display this dialog box)
Display test results at end of run

6 Choose a Test Run name.

7 Run the test.
8 Review the test results.

Conclusion

WinRunner Results - H:\Program Files\Mercury Interactive\WinRunner\tmp\noname14

```
================================================================================
```

Expected results folder: H:\Program Files\Mercury Interactive\WinRunner\tmp\noname14\exp
Test Results Name: H:\Program Files\Mercury Interactive\WinRunner\tmp\noname14\res1
Operator Name:

Date: Tue Mar 17 12:00:21 2015

Summary:

Test Result:	OK
Total number of bitmap checks:	0
Total number of GUI checks:	0
Total Run Time:	00:00:04

Detailed Results Description

Line	Event	Result	Details	Time

```
--------------------------------------------------------------------------------

3     start run         run              noname14         00:00:00

23    stop run          pass             noname14         00:00:04

                    WinRunner        Results        -        H:\Program        Files\Mercury
Interactive\WinRunner\tmp\noname14


==================================================================
=========================

Expected results folder: H:\Program                                          Files\Mercury
Interactive\WinRunner\tmp\noname14\exp
Test Results Name:                  H:\Program                               Files\Mercury
Interactive\WinRunner\tmp\noname14\res1
Operator Name:

Date:                        Tue Mar 17 12:00:21 2015

Summary:
----------------

Test Result:                                          OK
Total number of bitmap checks:                        0
Total number of GUI checks:                           0
Total Run Time:                                       00:00:04


                              Detailed Results Description

Line     Event          Result         Details            Time
--------------------------------------------------------------------------------

3     start run         run              noname14         00:00:00

23    stop run          pass             noname14         00:00:04
```

Experiment No.: 3

Title: . To perform Synchronization test

Objective: Student should be able

- Describes when you should synchronize a test
- Synchronize a test
- Run the test and analyze the results

When Should You Synchronize?

When you run tests, your application may not always respond to input with the same speed. For example, it might take a few seconds:

- to retrieve information from a database
- for a window to pop up
- for a progress bar to reach 100%
- for a status message to appear

WinRunner waits a set time interval for an application to respond to input. The default wait interval is up to 10 seconds. If the application responds slowly during a test run, WinRunner's default wait time may not be sufficient, and the test run may unexpectedly fail.

If you discover a synchronization problem between the test and your application,
you can either:

- Increase the default time that WinRunner waits. To do so, you change the value of the **Timeout for Checkpoints and CS Statements** option in the Run tab of the General Options dialog box (**Settings** > **General Options**). This method affects all your tests and slows down many other Context Sensitive operations.
- Insert a *synchronization point* into the test script at the exact point where the problem occurs. A synchronization point tells WinRunner to pause the test run in order to wait for a specified response in the application. *This is the recommended method for synchronizing a test with your application.*

In the following exercises you will:
✓ create a test that opens a new order in the Flight Reservation application and
inserts the order into the database
✓ change the synchronization settings
✓ identify a synchronization problem
✓ synchronize the test
✓ run the synchronized test

Creating a Test

In this first exercise you will create a test that opens a new order in the Flight Reservation application and inserts the order into a database

1 Start WinRunner and open a new test.

Start the Flight Reservation application and log in.

3 Start recording in Context Sensitive mode.

4 Create a new order.

5 Fill in flight and passenger information.

6 Insert the order into the database.

7 Delete the order.

8 Stop recording.

9 Save the test.

Changing the Synchronization Setting

1 Open the General Options dialog box.

2 Click the Run tab.

3 Change the value to 1000 milliseconds (1 second).

4 Click OK to close the dialog box.

Identifying a Synchronization Problem

1 Make sure that the *lesson4* test window is active in WinRunner.

2 Choose Run from Top.

3 Run the test.

4 Click Pause in the WinRunner message window.

Synchronizing the Test

1 Make sure that the *lesson4* test window is active in WinRunner.

2 Place the cursor at the point where you want to synchronize the test.

3 Synchronize the test so that it waits for the “Insert Done” message to appear in the status bar.

4 Manually change the 1 second wait in the script to a 10 second wait.

5 Save the test

Conclusion

```
WinRunner Results - H:\Documents and Settings\JNEC-12\Desktop\sytest
==========================================================================

Expected results folder:   H:\Documents and Settings\JNEC-12\Desktop\sytest\exp
Test Results Name:         H:\Documents and Settings\JNEC-12\Desktop\sytest\res3
Operator Name:             Date:          Tue Mar 17 11:57:53 2015

 Summary:
----------------

Test Result:                          OK
Total number of bitmap checks:    0
Total number of GUI checks:           0
Total Run Time:                   00:00:07


              Detailed Results Description


Line   Event        Result      Details       Time


------------------------------------------------------------------------

3   start run      run         sytest       00:00:00

23   wait for bitmap   OK        Img1         00:00:07

31   stop run       pass        sytest       00:00:07
```

Experiment No.:4

Title: Checking GUI Objects.

Objective: Student should be able to

- Explain how to check the behavior of GUI objects
- Create a test that checks GUI objects
- Run the test on different versions of an application and examine the results

How Do You Check GUI Objects?

When working with an application, you can determine whether it is functioning properly according to the behavior of its GUI objects. If a GUI object does not respond to input as expected, a defect probably exists somewhere in the application’s code. You check GUI objects by creating *GUI checkpoints*. A GUI checkpoint examines the behavior of an object’s properties. For example, you can check: the content of a field whether a radio button is on or off whether a pushbutton is enabled or disable

Adding GUI Checkpoints to a Test Script

1 Start WinRunner and open a new test.

2 Start the Flight Reservation application and log in.

3 Start recording in Context Sensitive mode.

4 Open the Open Order dialog box.

5 Create a GUI checkpoint for the Order No. check box.

6 Enter “4” as the Order No.

7 Create another GUI checkpoint for the Order No. check box.

8 Create a GUI checkpoint for the Customer Name check box.

9 Click OK in the Open Order dialog box to open the order.

10 Stop recording.

11 Save the test.

Conclusion

```
        WinRunner Results - H:\Program Files\Mercury Interactive\WinRunner\tmp\noname15

==============================================================================
=========

Expected results folder: H:\Program Files\Mercury Interactive\WinRunner\tmp\noname15\exp
Test Results Name:              H:\Program Files\Mercury Interactive\WinRunner\tmp\noname15\res2
Operator Name:

Date:                   Tue Mar 17 12:06:04 2015

 Summary:
----------------

Test Result:                                    OK
Total number of bitmap checks:                  0
Total number of GUI checks:                     3
Total Run Time:                                 00:00:02


                        Detailed Results Description

Line    Event       Result      Details       Time
-------------------------------------------------------------------------------

3    start run       run          noname15       00:00:00

12   start GUI checkpoint---          gui1          00:00:01

12   end GUI checkpointOK             gui1          00:00:01

23   start GUI checkpoint---          gui2          00:00:02

23   end GUI checkpointOK             gui2          00:00:02

24   start GUI checkpoint---          gui3          00:00:02

24   end GUI checkpointOK             gui3          00:00:02

25   stop run        pass         noname15       00:00:02
```

Experiment No.:5

Title: . Checking Bitmap Objects

Objective: Student should be able to

- Explains how to check bitmap images ina application
- Create a test that checks bitmaps
- Run the test in order to compare bitmaps in different versions of an application
- Analyze the results

Theory:

How Do You Check a Bitmap?

If your application contains bitmap areas, such as drawings or graphs, you can check these areas using a *bitmap checkpoint*. A bitmap checkpoint compares captured bitmap images pixel by pixel.

Adding Bitmap Checkpoints to a Test Script

1 Start WinRunner and open a new test.

2 Start the Flight Reservation application and log in.

3 Start recording in Context Sensitive mode.

4 Open order #6.

5 Open the Fax Order dialog box.

6 Enter a 10-digit fax number in the Fax Number box.

7 Move the Fax Order dialog box.

8 Switch to Analog mode.

9 Sign your name in the Agent Signature box.

10 Switch back to Context Sensitive mode.

11 Insert a bitmap checkpoint that checks your signature.

12 Click the Clear Signature button.

13 Insert another bitmap checkpoint that checks the Agent Signature box.

14 Click the Cancel button on the Fax Order dialog box.

15 Stop recording.

16 Save the test.

Conclusion:

```
                    WinRunner Results - H:\Documents and Settings\JNEC-12\Desktop\bit

================================================================================

Expected results folder: H:\Documents and Settings\JNEC-12\Desktop\bit\exp

Test Results Name:              H:\Documents and Settings\JNEC-12\Desktop\bit\res3
Operator Name:

Date:                  Tue Mar 17 11:44:42 2015

Summary:
----------------

Test Result:                                    OK
Total number of bitmap checks:                  2
Total number of GUI checks:                     0
Total Run Time:                                 00:00:38


                              Detailed Results Description

Line    Event        Result       Details        Time


-------------------------------------------------------------------------------

3    start run      run           bit                  00:00:00

58    bitmap checkpoint OK             Img1          00:00:37

67    bitmap checkpoint OK             Img2          00:00:38

76    stop run       pass          bit                  00:00:38
```

Experiment No.:6

Title: Testing with TSL**.**

Objective: Student should be able to

- Use visual programming to add functions to The recorded test scripts
- Add decision-making logic to a test script
- Debug a test script
- Run a test on a new version of an application and analyze the result

Steps:

1 Start WinRunner and open a new test.

2 Start the Flight Reservation application and log in.

3 Start recording in Context Sensitive mode.

4 Open order #4.

5 Open the Fax Order dialog box.

6 Click Cancel to close the dialog box.

7 Stop recording.

8 Save the test

9 Insert a blank line above the button_press ("Cancel"); **statement and place the cursor at the beginning of this line.**

10Open the Fax Order dialog box.

11 Query the # Tickets field

12 Query the Ticket Price field.

13 Query the Total field.

14Close the Fax Order dialog box.

15 Save the test.

16 Place the cursor below the last edit_get_text statement in the *saved* script.

17 Add the following statements to the test script exactly as they appear below. Note that the tabs or spaces at the beginning of the second and fourth lines are optional.

```
if (tickets*price == total)
tl_step ("total", 0, "Total is correct.");
```

else
tl_step ("total", 1, "Total is incorrect.");
these statements mean: “If *tickets* multiplied by *price* equals *total*, report that the total is correct, otherwise (else) report that the total is incorrect.

18 Save the test.

Conclusion:

```
                    WinRunner Results - H:\Documents and Settings\JNEC-12\Desktop\tsl

=======================================================================

Expected results folder: H:\Documents and Settings\JNEC-12\Desktop\tsl\exp
Test Results Name:            H:\Documents and Settings\JNEC-12\Desktop\tsl\res1
Operator Name:

Date:                 Fri Mar 27 16:14:44 2015

 Summary:
---------------

Test Result:                                    OK
Total number of bitmap checks:                  0
Total number of GUI checks:                     0
Total Run Time:                                 00:00:03


                         Detailed Results Description

Line    Event       Result      Details       Time


--------------------------------------------------------------------------------

3    start run      run         tsl          00:00:00

26    tl_step       ---         Step: total, Status: Pass, Description: Total is
correct00:00:03

30    stop run      pass        tsl          00:00:03
```

Experiment No.:7

Title: Creating data driven test

Objective: Student should be able to

- Use the DataDriver Wizard to create a data-driven test
- Use regular expressions for GUI object names that vary with each iteration of a test
- Run a test with several iterations and analyze the results

Theory:

Once you have successfully debugged and run your test, you may want to see how the same test performs with multiple sets of data. To do this, you convert your test to a data-driven test and create a corresponding data table with the sets of data you want to test.

Converting your test to a data-driven test involves the following steps:

- Adding statements to your script that open and close the data table.
- Adding statements and functions to your test so that it will read from the data table and run in a loop while it applies each set of data.
- Replacing fixed values in recorded statements and checkpoint statements with parameters, known as *parameterizing* the test.

You can convert your test to a data-driven test using the DataDriver Wizard or you can modify your script manually.

When you run your data-driven test, WinRunner runs the parameterized part(s) of the test one time (called an *iteration*) for each set of data in the data table, and then displays the results for all of the iterations in a single Test Results window.

In Lesson 7 you created a test that opened a specific flight order and read the number of tickets, price per ticket, and total price from a fax order dialog box in order to check that the total price was correct. In this lesson you will create a test that performs the same check on several flight orders in order to check that your application computes the correct price for various quantities and prices of tickets.

1 Create a new test from the test experiment 6.

2 Run the DataDriver Wizard.

3 Create a data table for the test.

4 Assign a table variable name.

5 Select global parameterization options.

6 Select the data to parameterize.

7 Open the data table.

8 Add data to the table.

9 Save and close the table.

10Save the test.

11Locate the Fax Order window in the *flight1a.gui* GUI map file.

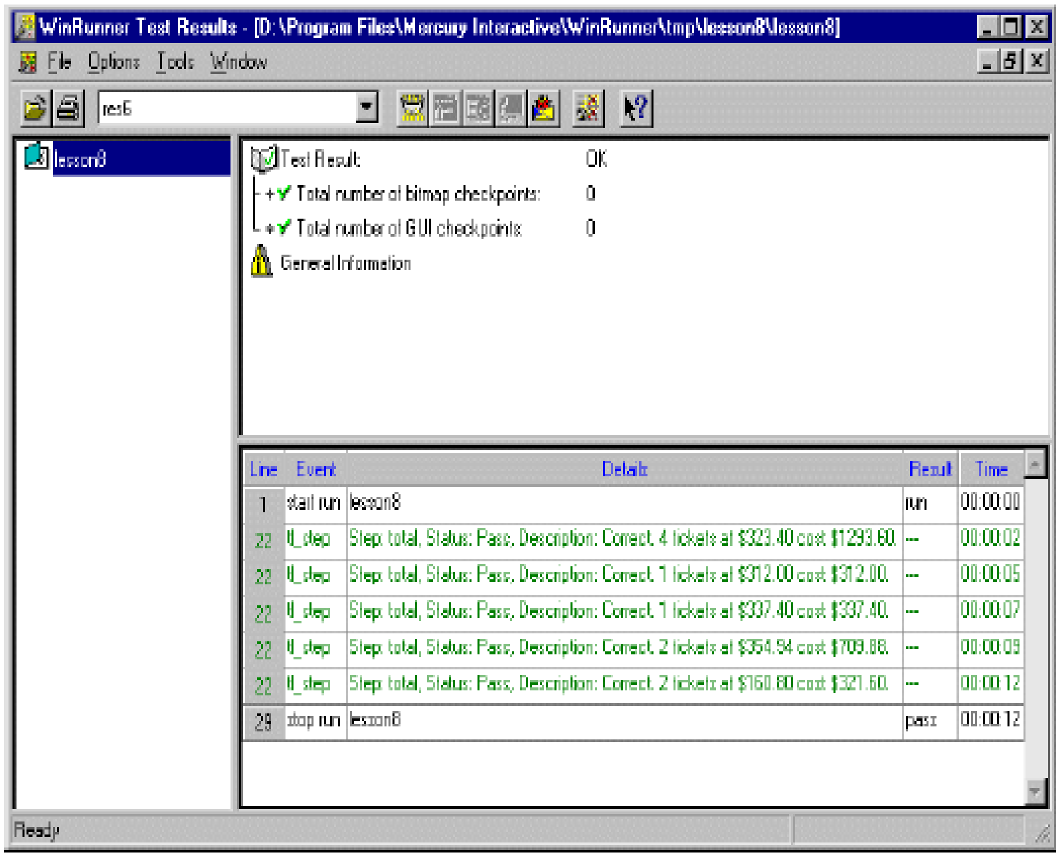
12 Modify the window label with a regular expression.

13 Close the Modify dialog box.

14 Modify the tl_step statements.
Locate the first **tl_step** statement in your script. Delete the words “total is correct.” and replace them with, “Correct. "tickets" tickets at $"price" cost $"total".”
tl_step("total",0, "Correct. "tickets" tickets at $"price" cost $"total".");
Use the same logic to modify the next **tl_step** statement to report an incorrect result. For example:
tl_step("total", 1, "Error! "tickets" tickets at $"price" does not equal $"total".");
Now you will be able to see which data is used in each iteration when you view the results.

15 Save the test.

conclusion


WinRunner Test Results - [D:\Program Files\Mercury Interactive\WinRunner\tmp\lesson8\lesson8]
File Options Tools Window
res6
lesson8
Ready

Test Result OK
Total number of bitmap checkpoints: 0
Total number of GUI checkpoints: 0
General Information

Line	Event	Details	Result	Time
1	start run	lesson8	run	00:00:00
22	tl_step	Step: total, Status: Pass, Description: Correct. 4 tickets at $323.40 cost $1293.60.	---	00:00:02
22	tl_step	Step: total, Status: Pass, Description: Correct. 1 tickets at $312.00 cost $312.00.	---	00:00:05
22	tl_step	Step: total, Status: Pass, Description: Correct. 1 tickets at $337.40 cost $337.40.	---	00:00:07
22	tl_step	Step: total, Status: Pass, Description: Correct. 2 tickets at $354.94 cost $709.88.	---	00:00:09
22	tl_step	Step: total, Status: Pass, Description: Correct. 2 tickets at $160.80 cost $321.60.	---	00:00:12
29	stop run	lesson8	pass	00:00:12

Experiment No.: 8

Title: Manual Testing

This experiment helps you write manual Testcases for Login Form as below

TEST CASE ID	TEST DESCRIPTION	TEST PREREQUISITE	TEST INPUTS	TEST RESULTS
1.	**USERNAME**	Should preceed with capital letter followed by small case	**Jayshri**	Accepted
2.	**USERNAME**	Should contain only alphabets	**Jay123**	Error
3.	**USERNAME**	Special characters not allowed	**#jay**	Error
4.	**USERNAME**	Should not preceed with digits	**123jaydp**	Error
5.	**USERNAME**	Blankspace or tab not allowed	**Cidco aurangabad**	Error
6.	**PASSWORD**	It should contain minimum 8 characters	**cidcoabd**	Accepted
7.	**PASSWORD**	Combination of digits and alphabets with special characters allowed	**cidcoabd123#$**	Accepted
8.	**PASSWORD**	Should not preceed with special characters	**@cidcoabd**	Error
9.	**PASSWORD**	One digit and one alphabet is compulsory	**cidcoabd123#$**	Accepted
10.	**PASSWORD**	Should not exceed 16 characters	**Ac3ysj2#$%1@0klef$**	Error
11.	**SUBMIT**	Click once to login	**Single click**	Accepted
12.	**SUBMIT**	Double click ,no action will be	**Double click**	No action

		performed		
13.	**SUBMIT**	Moving the cursor towards button , it gets highlighted	**Move the cursor on button**	Button highlighted
14.	**CANCEL**	Single click ,login gets cancelled	**Single click**	Cancelled
15.	**CANCEL**	Double click,no action will be performed	**Double click**	No action
16.	**CANCEL**	Moving the cursor on button,highlights it	**Move the cursor on button**	highlighted