Online Guide

WinRunner® 7.0

Testing WAP Applications Beta

Table of Contents

Books Online
Find
Find Again
Help
Top of Chapter
Back

Books Online
Find
Find Again
Help
Top of Chapter
Back

Introduction

Welcome to WinRunner with add-in support for WAP. This guide explains how to use WinRunner to successfully test WAP applications. It should be used in conjunction with the *WinRunner User's Guide* and the *TSL Online Reference* or the *TSL Reference Guide*.

This chapter describes:

Using the WAP Add-in

The WAP Add-in is an add-in to WinRunner, Mercury Interactive's enterprise functional testing tool for Microsoft Windows applications. The WAP Add-in enables you to test the functionality of your WAP application while it runs in a WAP emulator. The terminology used in this add-in and documentation is described below:

A *WAP application* is WML code that provides content for WAP emulators (in the same way that a Web application is HTML, JavaScript, and ActiveX code, among others, that provides content for Web browsers).

WAP emulators enable users to access WAP applications, made up of WML code (in the same way that Web browsers, such as Microsoft Internet Explorer and Netscape Navigator, enable users to access Web applications). The WinRunner Add-in for WAP supports the Nokia and Phone.com emulators.

Each WAP emulator may contain multiple *devices*, or phones. For example, in the Nokia WAP Toolkit Version 2.0, you can select the following devices:

- Blueprint Phone (WAP 1.2)
- Nokia 7110 (WAP 1.1)

Similarly, for Phone.com, the following devices (configurations) are available:

- ALAV - Alcatel One Touch Pocket phone
- IM1K - Motorola iDEN phone
- SH - Samsung Duette phone
- SP01 - Mitsubishi CDPD phone
- UPG1 - Generic phone

You can create a suite of tests and then run the tests each time you change your WAP application. To create a test, use WinRunner to record the operations you perform on your emulator. As you click on objects in the emulator, WinRunner generates a test script in TSL, Mercury Interactive's C-like test script language.

You can further enhance the recorded test script by inserting bitmap and GUI checkpoints. A bitmap checkpoint can compare bitmaps of a phone screen. A GUI checkpoint can compare property values of WAP emulator objects, such as the URL, the WML source, displayed text in a phone screen, or any properties of standard GUI objects.

How the WAP Add-in Identifies WAP Emulator Objects

WinRunner learns a set of default properties for each object you operate on while recording a test. These properties enable WinRunner to obtain a unique identification for every object that you test. This information is stored in the GUI map. WinRunner uses the GUI map to help it locate objects during a test run.

WinRunner identifies each WAP emulator that it encounters as a separate window. For each emulator, the class is “window”. For all objects within a WAP emulator, the class is “object”.

For example, a key in the Nokia 7110 WAP emulator may have the following information in the GUI map:

```
{
class: object,
label: "entrykey.2"
}
```

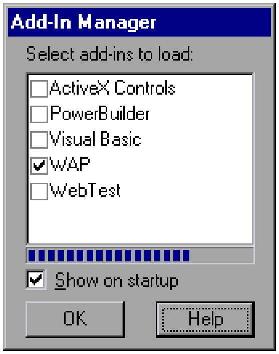
You can view the contents of your GUI map files in the GUI Map Editor, by choosing **Tools > GUI Map Editor**. The GUI Map Editor displays the logical names and the physical descriptions of objects. If the WAP application is open, highlighting an object in the GUI Map Editor also highlight the object in the WAP emulator. For additional information on GUI maps, refer to the “Understanding the GUI Map” section in the *WinRunner User’s Guide*.

Loading the WAP Add-in

Before you begin testing your WAP application, make sure that you have installed the WAP Add-in. For installation instructions, refer to the *WinRunner WAP Add-in Installation Guide*. After you install the WAP Add-in, you must load it when you start WinRunner.

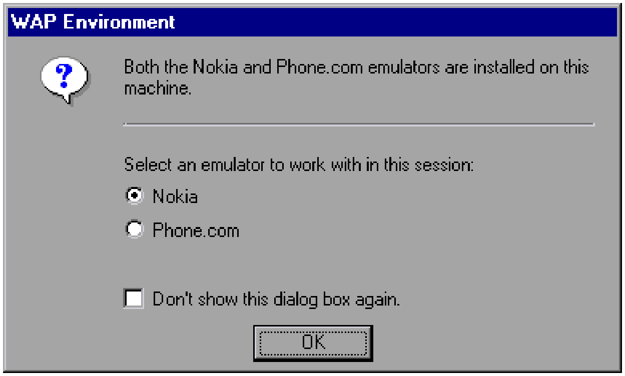
To load the WAP Add-in:

1 Click **Programs > WinRunner > WinRunner** in the Start menu. The Add-In Manager dialog box opens.



2 Select **WAP** and click **OK**.

3 If multiple supported WAP emulators are installed on your machine, the WAP Environment dialog box opens. Choose a WAP emulator to work with in the current session and click **OK**.



WAP Environment

Both the Nokia and Phone.com emulators are installed on this machine.

Select an emulator to work with in this session:

(●) Nokia

( ) Phone.com

☐ Don't show this dialog box again.

Tip: If you select the **Don’t show this dialog box again** check box, you can choose to display this dialog box in the future by setting the SHOW_WAP_DLG parameter to “1” in the *wrun.ini* file.

WinRunner opens with the WAP Add-in loaded.

For additional information on the Add-in Manager, refer to the *WinRunner User’s Guide.*

Creating Tests

You can quickly create a test script by recording the operations you perform in your WAP application.

This chapter describes:

About Creating Tests

You can create a test by recording, programming, or a combination of both methods. The easiest way to create a test is by recording. When you record a test, the operations that you perform on a Web site are recorded in the test script as statements in Test Script Language (TSL). Usually you create a script by recording, and then you use programming to enhance the recorded script.

You can further increase the power of your test scripts by adding checkpoints that compare bitmaps of the phone screen, the text displayed in a phone screen, the URL, and the WML source code in a WAP application for differences between test runs.

Planning Tests

Before you start recording, you should plan your test. You should consider the following:

- The functionality you want to test. Short tests that check specific functions of the application or complete a transaction are better than long tests that perform several tasks.
- The information you want to check during the test. A checkpoint can check for differences in the WML source code or the displayed text of a WAP application. For more information, see Chapter 1, **Checking WAP Applications**.

Notes for testing with the Phone.com emulator: When testing with the Phone.com emulator, you must work in the Global GUI Map File mode. If you work in the GUI Map File per Test mode, WinRunner cannot record or run tests properly.

Do not resize the Phone.com emulator window.

If you plan to record and run tests on multiple Phone.com devices within a single WinRunner session, you must use the **phone_GUI_load** function to change the loaded GUI map file whenever you change the Phone.com device. For additional information, see **page 23**.

For more information on planning tests, refer to the “Creating Tests” section in the *WinRunner User’s Guide*. If you are using TestDirector to organize the testing process, you can also refer to the “Test Planning” section in the *TestDirector User’s Guide.*

Recording Tests

After planning your test, you are ready to start recording your test script using WinRunner's Context Sensitive recording mode. In this mode, WinRunner records the operations you perform on your WAP application and automatically generates a test script.

To create a test script:

1 Start WinRunner with the WAP Add-in, as described in **Loading the WAP Add-in** on page 8.

Note: You must start WinRunner before you start your WAP emulator. Otherwise, WinRunner may not record and run your test script properly.

---

2 Start your WAP application.

3 In WinRunner, choose **File > New**, or click the **New** button to create a new test.

4 Choose **Create > Record–Context Sensitive**, or click the **Record** button. WinRunner starts recording your operations.

Tip: If you want to run the same test on both the Nokia and Phone.com emulators, you should record it on the Nokia emulator.

---

5 Perform the sequence in your WAP application that you want to record.

As you record, each operation you perform generates a TSL statement in your test script.

You can insert checkpoints in your test. For more information, see Chapter 1, **Checking WAP Applications**.

6 To stop recording, choose **Create > Stop Recording** or click the **Stop** button.

7 To save your test, choose **File > Save** and assign the test a name.

Notes: If you are working with the Nokia emulator in the Global GUI Map File mode, you must save the information that WinRunner learns about each object (the phone screen and keypad keys) in a GUI map file. When working with the Phone.com emulator, the GUI map files are automatically managed by WinRunner. For more information on GUI map files, refer to the “Understanding the GUI Map” section in the *WinRunner User’s Guide.*

If you recorded test scripts with WinRunner before installing the WAP Add-in, clear the GUI map before creating any WAP Add-in scripts. In WinRunner, choose **Tools > GUI Map Editor** to open the GUI Map Editor. Then choose **Edit > Clear All** and close the GUI Map Editor.

Understanding Your Test Script

As you record, each operation you perform generates a statement in Mercury Interactive’s Test Script Language (TSL), in your test script.

The following is a sample of a WinRunner test script recorded on the Phone.com WAP emulator:

```
# Phone
    set_window ("Phone", 4);
    phone_key_click("KeySoft1");
    phone_key_click("KeyDown");
    phone_key_click("KeyDown");
    phone_key_click("3");
    phone_key_click("4");
```

The following is a sample of a WinRunner test script recorded on the Nokia 7110 WAP emulator:

```
# Phone
    set_window("Phone", 8);
    phone_sync(); # http://wapsight.com/;
    phone_key_click("KeySoft1", FALSE, 131); # http://wapsight.com/;
    phone_key_click("KeySoft2", FALSE, 110); # http://wapsight.com/;
    phone_key_click("KeyNavigate", FALSE, 110); # http://wapsight.com/;
    phone_sync(); # http://wapsight.com/info/headlines.wml;
    phone_key_click("KeyNavigate", FALSE, 110); #
        http://wapsight.com/info/headlines.wml;
```

set_window Function

Each time you switch to a WAP emulator window or change a phone device, WinRunner generates a **set_window** statement. The **set_window** statement directs input to the application in the WAP emulator. It has the following syntax:

set_window (*window*, *time* **);**

window	The logical name of the window.
time	The amount of time, in seconds, added to the timeout option to give the maximum interval before the next statement is executed. If the window is found before the maximum time is reached, the test continues to run.

For example, the statement:

```
set_window ( "Phone", 3 );
```

indicates that “Phone” is the name of the current WAP emulator. WinRunner waits the default timeout value plus a maximum of 3 seconds for the WAP emulator window to open. The default timeout value is defined in the General Options dialog box (**Settings > General Options)**. For information on setting the timeout value in the General Options dialog box, refer to the “Setting Global Testing Options” chapter in the *WinRunner User’s Guide*.

For more information on the **set_window** function, refer to the *TSL Online Reference* (**Help > TSL Online Reference**).

Tip: Supported emulators for WAP add-in functions are displayed in the left margin of the book, beside the corresponding function.

Nokia
Phone.com

phone_key_click Function

WinRunner records this function when you click a phone key. It has the following syntax:

phone_key_click (*key* **[,** *delay* **[,** *timeout* **]]);**

key	The logical name of the phone key.
delay	The Boolean parameter indicating that there is an additional delay to compensate for inserting a new letter while editing. This parameter is useful when editing, as WinRunner can differentiate between typing AA and B. This parameter is recorded only for the Nokia emulator. It is always recorded as “FALSE.” If the value of this parameter is changed to “TRUE,” then WinRunner waits the additional delay, as described above.
timeout	The amount of time (in milliseconds) between pressing and releasing the key/mouse. This enables support for features in which the length of time the key is pressed is important.

Books Online
Find
Find Again
Help
Top of Chapter
Back

Nokia record
and run tests

Phone.com
run tests only

phone_sync Function

WinRunner records this function on the Nokia emulator after any phone navigation. It instructs WinRunner to wait until the phone is ready to handle the next operation. This function is inserted automatically in the test scripts after a **phone_key_click** statement is recorded on a Nokia phone that included navigation. It has the following syntax:

phone_sync ([*redirect* **[,** *timeout* **]]);**

redirect	An optional Boolean parameter indicating that the phone waits an additional amount of time to redirect to another URL.
timeout	The amount of time (in seconds) that the phone waits to try to establish a connection. This is the expected period of time during which WinRunner expects the navigation to be concluded.

Tip: If your test does not run properly in the Phone.com emulator, it may not be properly synchronized.To solve this problem, you can insert a **phone_sync** statement immediately following any statement in which synchronization is a problem.

Enhancing the WAP Add-in Scripts with TSL

You can enhance your recorded test scripts by adding **phone_** TSL functions from the Function Generator. In the Function Generator, the **phone_** functions are located in the *WAP* category. For information on the Function Generator, refer to the “Generating Functions” chapter in the *WinRunner User’s Guide*.

The following **phone_** TSL functions are available. Note that you can find additional information about these functions and examples of usage in the *TSL Online Reference* (**Help > TSL Online Reference**).

Tip: Supported emulators for WAP add-in functions are displayed in the left margin of the book, beside the corresponding function.

Nokia
Phone.com

phone_append_text Function

This function appends the specified text string to the current contents of the phone editor.

Note: This function works only while the phone is in editing mode. Trying to use this function while the phone is not in editing mode will return an illegal operation error code.

This function has the following syntax:

phone_append_text (*text* **);**

text	The text string to append in the phone editor.

Nokia
Phone.com

phone_edit_set Function

This function replaces the contents of the phone editor with the specified text string.

Note: This function works only while the phone is in editing mode. Trying to use this function while the phone is not in editing mode will return an illegal operation error code.

It has the following syntax:

phone_edit_set (*text* **) ;**

text	The text string to insert in the phone editor.

Nokia
Phone.com

phone_get_name Function

This function returns the name of the device. It has the following syntax:

phone_get_name (*name* **);**

name	The name of the device.

Books Online
Find
Find Again
Help
Top of Chapter
Back

Phone.com

phone_GUI_load Function

This function unloads the currently loaded GUI map file and loads the GUI map for the specified Phone.com phone. It has the following syntax:

phone_GUI_load ([*name* **]);**

name	The name of the device.

Tips: The name of the device is the first word in the emulator window title. It is also the name of the Phone.com file with the model configuration. For example, to load the Alcatel One Touch Pocket phone device, use the ALAV name:

```
phone_GUI_load ( "ALAV" );
```

You can create buttons on the User toolbar that execute this function and switch the GUI map file for emulators. For additional information, refer to the “Customizing WinRunner’s User Interface” chapter in the *WinRunner User’s Guide*.

Note: If only the Phone.com emulator is installed on your machine, or if you choose (in the WAP Environment dialog box) to work with Phone.com in a WinRunner session, then when you work with the WAP Add-in, WinRunner automatically loads a GUI map file. Each Phone.com device has a corresponding GUI map file. Note that if you change the Phone.com configuration during a WinRunner session, you must use the **phone_GUI_load** function to load the GUI map file for the new configuration. If no device name is specified, this function automatically detects the active device and loads the appropriate GUI map file.

Nokia Phone.com

phone_navigate Function

This function directs the phone to connect to the specified site. It has the following syntax:

phone_navigate (*URL* **[,** *timeout* **]);**

URL	The URL to which the phone navigates.
timeout	The amount of time (in seconds) the phone waits while trying to establish a connection.

For more information on TSL functions, refer to the *TSL Online Reference* (**Help > TSL Online Reference**).

Checking WAP Applications

By adding GUI and bitmap checkpoints to your test scripts, you can compare the behavior of different versions of your WAP application.

This chapter describes:

About Checking WAP Applications

You can use bitmap and GUI checkpoints in your test script to help you examine your WAP application and detect defects. You can use bitmap checkpoints to check that the phone screen is displayed correctly. You can use GUI checkpoints to check the text displayed in a phone screen, the URL, and the WML source code in a WAP application. Checkpoints compare expected and actual values, so that you can check for differences between test runs.

Books Online
Find
Find Again
Help
Top of Chapter
Back

Checking Bitmaps of Phone Screens

You can create a bitmap checkpoint that compares bitmaps of a phone screen, to make sure that it is displayed properly.

Tip: You can use a bitmap checkpoint to check the displayed text in a Nokia phone screen, since the **DisplayText** property check in GUI checkpoints is not supported for the Nokia emulator.

Note: Bitmap checkpoints are not portable between WAP emulators or between different devices (configurations) of the same emulator, since the phone screens are different for each emulator.

To check bitmaps of a phone screen:

1 Choose **Create > Bitmap Checkpoint > For Object/Window**.

The WinRunner window is minimized to an icon, the mouse pointer turns into a pointing hand, and a help window opens.

2 Click the object to check:

- For a Nokia emulator, click the phone screen.
- For a Phone.com emulator, click the phone.

WinRunner captures a bitmap of the phone screen (Nokia) or the entire phone (Phone.com) and stores it in the expected results folder (*exp*) of the test. WinRunner inserts a checkpoint in your test script as an **obj_check_bitmap** statement. For information on the **obj_check_bitmap** function, refer to the *TSL Online Reference* (**Help > TSL Online Reference**). For additional information on bitmap checkpoints, refer to the “Checking Bitmaps” chapter in the *WinRunner User’s Guide*.

Checking GUI Objects in WAP Applications

You can create a GUI checkpoint that checks one or more of the following WAP properties:

- the text displayed in a phone screen (for the Phone.com emulator in Windows NT)
- the URL of the loaded WML file
- the WML source code of your WAP application

To create a GUI checkpoint for a WAP application:

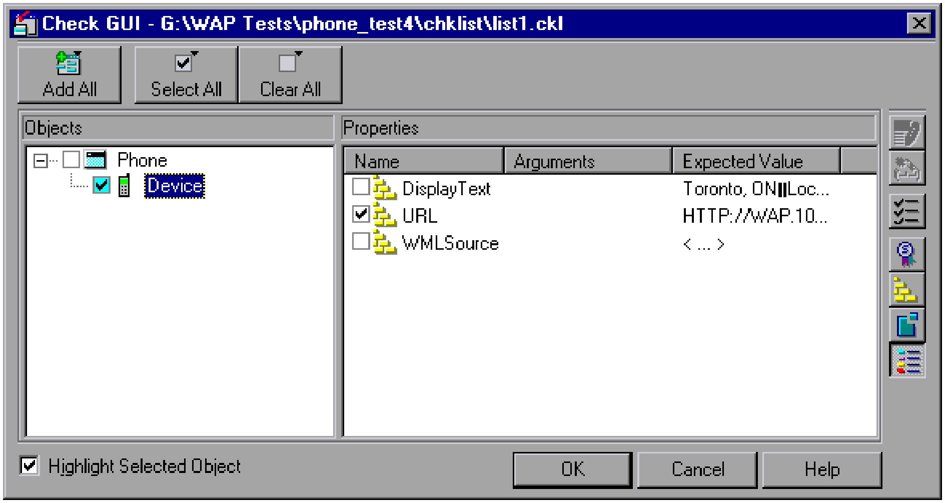
1 Choose **Create > GUI Checkpoint > For Object/Window**.

The WinRunner window is minimized to an icon, the mouse pointer turns into a pointing hand, and a help window opens.

2 Double-click the object to check:

- For a Nokia emulator, double-click the phone screen.
- For a Phone.com emulator, double-click the phone.

The Check GUI dialog box opens, and the object is highlighted.



Name	Arguments	Expected Value
☐ DisplayText		Toronto, ON‖Loc...
☑ URL		HTTP://WAP.10...
☐ WMLSource		< ... >

☑ Highlight Selected Object

Select the property or properties to check. If you want, edit the expected value. For additional information, refer to the “Checking GUI Objects” chapter in the *WinRunner User’s Guide*.

- To check the text displayed in a phone screen (for the Phone.com emulator in Windows NT), select the **DisplayText** property check. For additional information, see **Checking Text Displayed in Phone Screens** on page 32.
- To check the URL of the loaded WML file, select the **URL** property check. For additional information, see **Checking the URL of the Loaded WML File** on page 34.
- To check the WML source code of your WAP application, select the WMLSource property check. For additional information, see **Checking WML Source Code in Your WAP Application** on page 36.

3 Click **OK** to close the Check GUI dialog box.

WinRunner captures the object information and stores it in the test’s expected results folder. The WinRunner window is restored and a checkpoint appears in your test script as an **obj_check_gui** statement, for example:

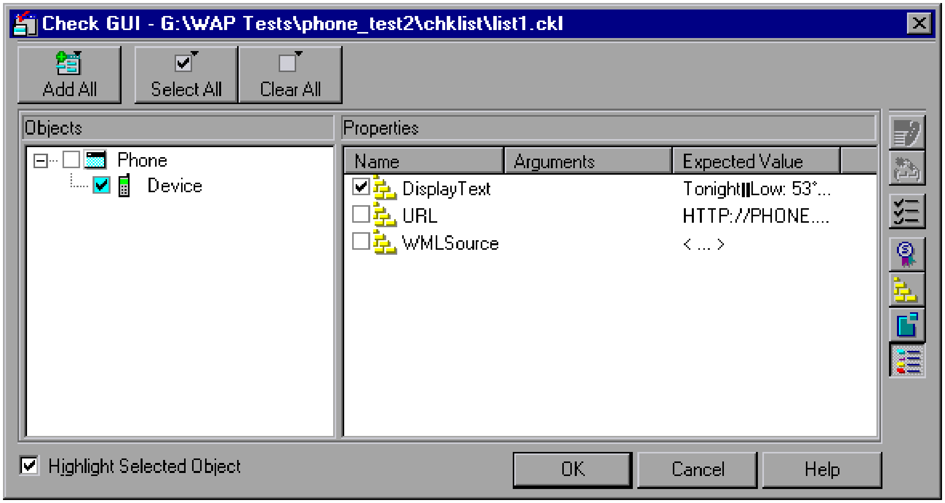
```
obj_check_gui("Device", "list1.ckl", "gui1", 1);
```

For more information on the **obj_check_gui** function, refer to the *TSL Online Reference* (**Help > TSL Online Reference**).

Checking Text Displayed in Phone Screens

Note: This feature is supported only for the Phone.com emulator on Windows NT.

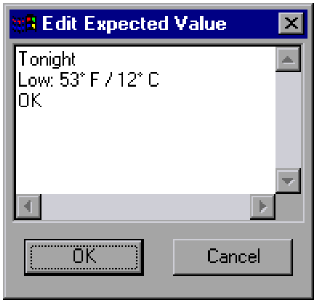
You can create a GUI checkpoint that checks the text displayed in a phone screen, as described in **Checking GUI Objects in WAP Applications** on page 29. In the Check GUI dialog box, you select the **DisplayText** property check.


Check GUI - G:\WAP Tests\phone_test2\chklist\list1.ckl
Add All
Select All
Clear All
Objects
Phone
Device
Properties
Highlight Selected Object
OK
Cancel
Help

Name	Arguments	Expected Value
✓ DisplayText		Tonight‖Low: 53°...
URL		HTTP://PHONE....
WMLSource		< ... >

The text in the phone screen is displayed under the Expected Value column.

If the entire text in the phone screen is not displayed, or to edit the expected value of the displayed text, you highlight **DisplayText** and click the **Edit Expected Value** button or double-click the value in the **Expected Value** column. The Edit Expected Value edit box opens, in which you can view the entire displayed text and edit its expected value.

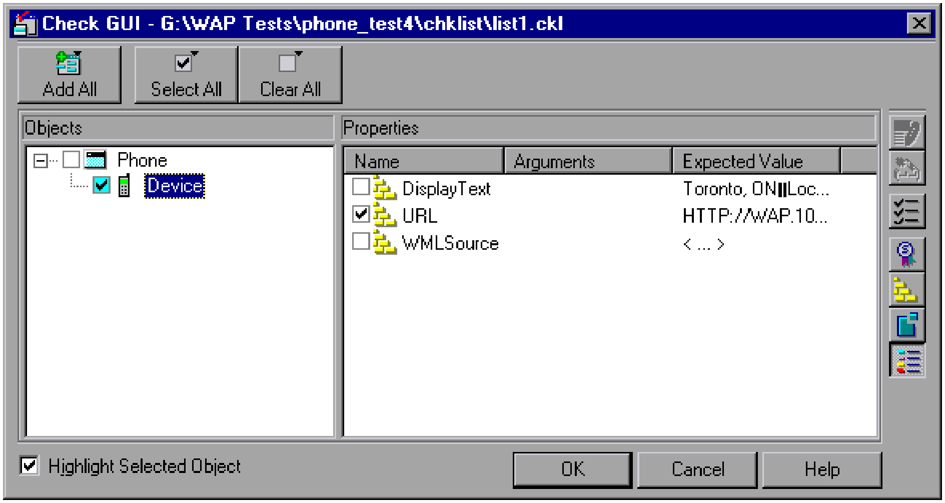


Tonight
Low: 53° F / 12° C
OK

If you want, edit the expected value, and click **OK** to close the Edit Expected Value dialog box and return to the Check GUI dialog box.

Checking the URL of the Loaded WML File

You can create a GUI checkpoint that checks the URL of the loaded WML file, as described in **Checking GUI Objects in WAP Applications** on page 29. In the Check GUI dialog box, **URL** is selected as the default property check for all WAP emulator devices.

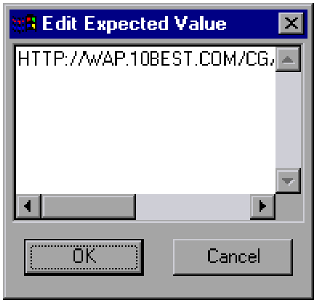

Check GUI - G:\WAP Tests\phone_test4\chklist\list1.ckl
Add All
Select All
Clear All
Objects
Phone
Device
Properties
Highlight Selected Object
OK
Cancel
Help

Name	Arguments	Expected Value
☐ DisplayText		Toronto, ON‖Loc...
☑ URL		HTTP://WAP.10...
☐ WMLSource		< ... >

The value of the URL is displayed under the Expected Value column.

Books Online
Find
Find Again
Help
Top of Chapter
Back

If the entire URL is not displayed, or to edit the expected value of the URL, you can highlight the **URL** property check and click the **Edit Expected Value** button or double-click the value in the **Expected Value** column. The Edit Expected Value edit box opens, in which you can view the entire URL and edit its expected value.

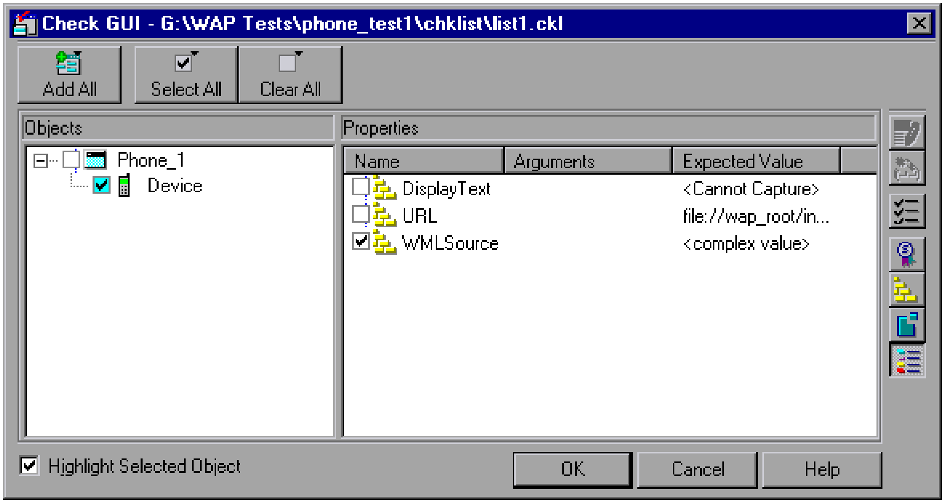


HTTP://WAP.10BEST.COM/CG/

If you want, edit the expected value, and click **OK** to close the Edit Expected Value dialog box and return to the Check GUI dialog box.

Checking WML Source Code in Your WAP Application

You can create a GUI checkpoint that checks the WML source code of your WAP application, as described in **Checking GUI Objects in WAP Applications** on page 29. In the Check GUI dialog box, you select the **WMLSource** property check. WinRunner takes a few seconds to capture the value of the WML source code, and "complex value" is displayed under the Expected Value column. The dialog box is displayed as follows:


Check GUI - G:\WAP Tests\phone_test1\chklist\list1.ckl
Add All
Select All
Clear All
Objects
Phone_1
Device
Properties
Highlight Selected Object
OK
Cancel
Help

Name	Arguments	Expected Value
☐ DisplayText		<Cannot Capture>
☐ URL		file://wap_root/in...
✓ WMLSource		<complex value>

Books Online
Find
Find Again
Help
Top of Chapter
Back

To view or to edit the expected value of the WML source code, you highlight **WMLSource** and click the **Edit Expected Value** button or double-click the value in the **Expected Value** column. Notepad opens and displays the WML source code. If you edit the source code, click **File > Save** and **File > Exit** when you are done to save your changes as the expected results.

Running Tests

Once you have created a test script, you run the test to check the behavior of your WAP application.

About Running Tests

When you run a test, WinRunner interprets your test script, line by line, and performs the operations on your WAP application.

Use WinRunner's Run commands to run your tests. You can run an entire test or a portion of a test. For more information, refer to the "Running Tests" chapter in the *WinRunner User's Guide*.

Note: If you are working with the Nokia emulator in the Global GUI Map File mode, you must load the appropriate GUI map files before you run your tests. For more information, refer to the "Understanding the GUI Map" section in the *WinRunner User's Guide*.

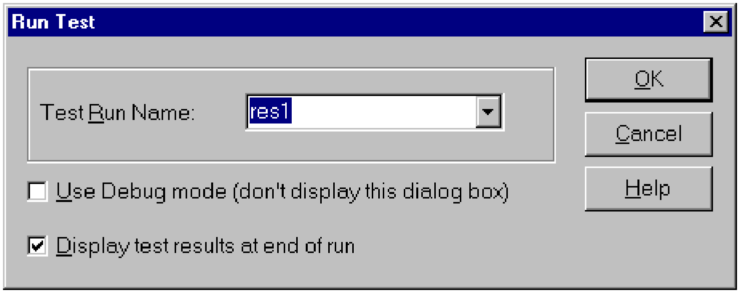
Notes for testing with the Phone.com emulator: If your test does not run properly, it may not be properly synchronized. To solve this problem, you can insert a **phone_sync** statement immediately following any statement in which synchronization is a problem. For additional information on the **phone_sync** function, see **Enhancing the WAP Add-in Scripts with TSL** on page 20 or refer to the *TSL Online Reference*. In addition, in the Advanced Run Options dialog box, you can change the value of the **Delay between execution of CS statements** option from 0 to 1000 milliseconds. For additional information on this option, refer to the “Setting Global Testing Options” chapter in the *WinRunner User’s Guide*.

Running a Test to Check Your WAP Application

When you run a test to check the behavior of your WAP application, WinRunner compares the current results with the expected results. You specify the folder in which the verification results for the test are saved.

To run a test to check your WAP application:

1. Open the test if it is not already open.
2. Make sure that **Verify** is selected from the dropdown list of run modes on the toolbar.
3. Choose **Run > Run from Top**, or click the **Run from Top** button.
4. In the Run Test dialog box, assign a name to the folder that will store the results, or accept the default name “res1”.



Test Run Name: res1

☐ Use Debug mode (don't display this dialog box)

☑ Display test results at end of run

Books Online
Find
Find Again
Help
Top of Chapter
Back

5 To instruct WinRunner to display the test results automatically following the test run (the default), select the **Display test results at end of run** check box.

6 Click **OK**. The Run Test dialog box closes and WinRunner runs the test.

Analyzing Test Results

After you run a test, you can view a report of all the major events that occurred during the test run in order to determine its success or failure.

This chapter describes:

About Analyzing Test Results

When a test run is completed, you can view detailed test results in the WinRunner Test Results window. The window contains a description of the major events that occurred during the test run, such as errors and checkpoints. You can view expected, debug, and verification results in the Test Results window. By default, the Test Results window displays the results of the most recently executed test run. For more information, refer to the “Analyzing Test Results” chapter in the *WinRunner User’s Guide*.

Viewing Results of a Test Run

When a test run is completed, test results are displayed in the WinRunner Test Results window.

To view test results:

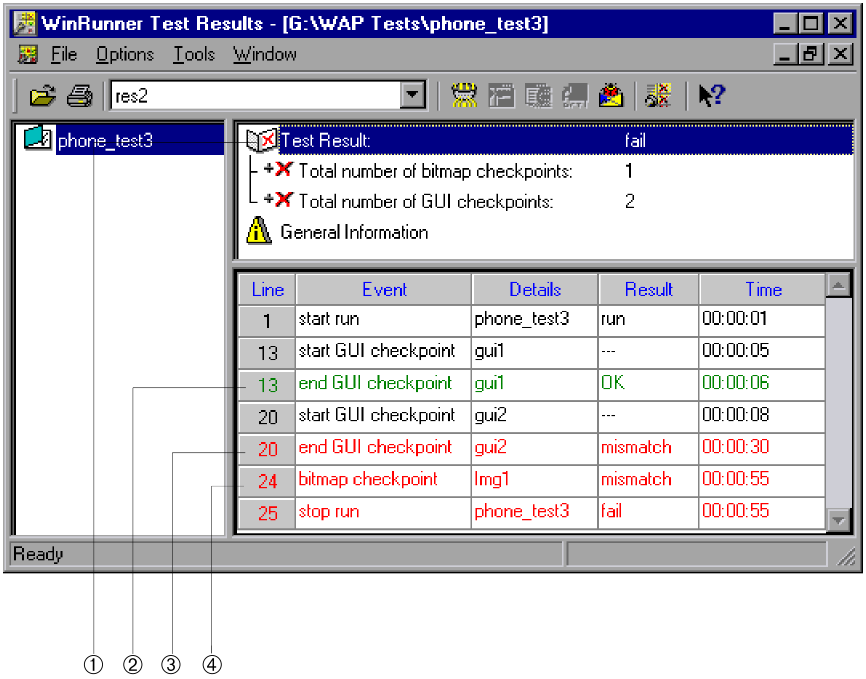
1 To open the WinRunner Test Results window, choose **Tools > Test Results** or click the **Test Results** button. Note that if the **Display test results at end of run** check box was selected in the Run Test dialog box before you ran the test, the Test Results window opens automatically.

1 *Indicates whether the test passed or failed, and lists all checkpoints performed during the test run.*

2 *Successful GUI checkpoint.*

3 *Failed GUI checkpoint.*

4 *Failed bitmap checkpoint that checks the phone screen.*



Test Result: fail
Total number of bitmap checkpoints: 1
Total number of GUI checkpoints: 2
General Information

Line	Event	Details	Result	Time
1	start run	phone_test3	run	00:00:01
13	start GUI checkpoint	gui1	---	00:00:05
13	end GUI checkpoint	gui1	OK	00:00:06
20	start GUI checkpoint	gui2	---	00:00:08
20	end GUI checkpoint	gui2	mismatch	00:00:30
24	bitmap checkpoint	Img1	mismatch	00:00:55
25	stop run	phone_test3	fail	00:00:55

- In the Test Results section you can see whether the test passed or failed, and how many checkpoints were included in the test.
- In the test log, look for GUI checkpoints statements. Failed GUI checkpoints are displayed in red; passed GUI checkpoints are displayed in green.
- In the test log, failed bitmap checkpoint statements are displayed in red; passed bitmap checkpoint statements are displayed in green.

2 By default, the Test Results window displays the results of the most recently executed test run.

- To view other test run results, click the **Results** box in the toolbar and select a test run.
- To view a text version of the test results, choose **Tools > Text Report** from the Test Results window. The report opens in Notepad.
- To view only specific types of results in the events column in the test log, choose **Options > Filters** or click the **Filters** button.
- To print test results directly from the Test Results window, choose **File > Print** or click the **Print** button.

  In the Print dialog box, choose the number of copies you want to print and click **OK**. Test results print in a text format.

3 To close the Test Results window, choose **File > Exit**.

Viewing Bitmap Checkpoint Results

You can view images of the expected and actual results of a bitmap checkpoint. If a mismatch occurs during a test run, you can also view an image showing the differences between the expected and actual results.

To view the results of a bitmap checkpoint:

1. In the Test Results window, look for a **bitmap checkpoint** entry in the Event column in the test log.
2. Double-click a **bitmap checkpoint** entry in the Event column.

   For a mismatch during a test run in Verification or Debug mode, the expected, actual, and difference bitmaps are displayed. For a mismatch during a test run in Update mode, only the expected bitmaps are displayed. For additional information on test run modes, refer to the “Running Tests” chapter in the *WinRunner User’s Guide*.

   For additional information on bitmap checkpoints, refer to the “Checking Bitmaps” chapter in the *WinRunner User’s Guide*.

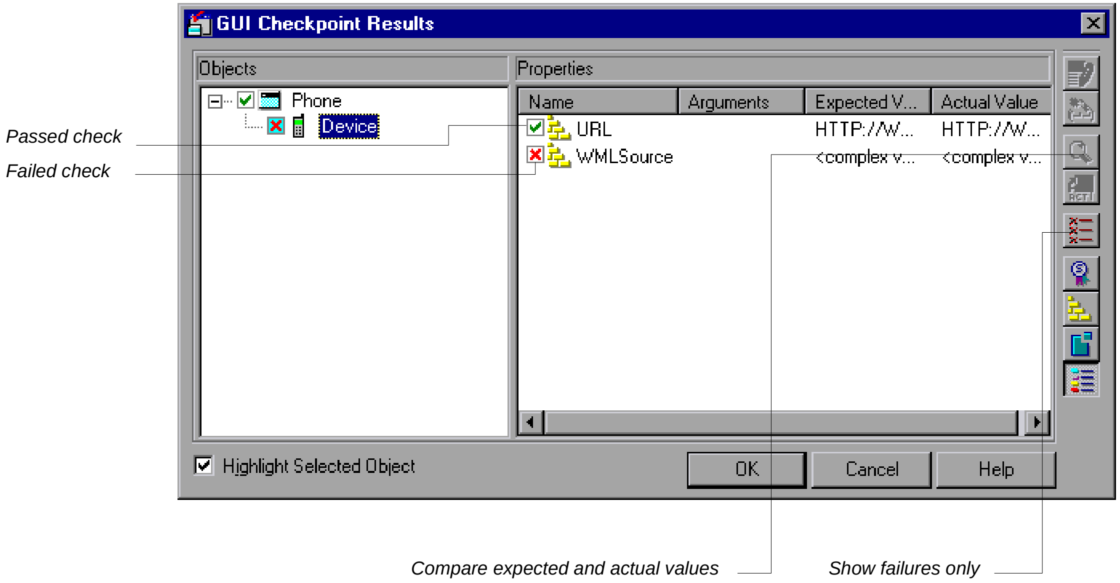
Viewing GUI Checkpoint Results

You can view the results of a GUI checkpoint using the GUI Checkpoint Results dialog box.

To view the results of a GUI checkpoint:

1 In the Test Results window, look for an **end GUI checkpoint** entry in the Event column in the test log.

2 Double-click an **end GUI checkpoint** entry in the Event column. The GUI Checkpoint Results dialog box opens.



The Objects column lists all the objects that WinRunner checked in the specified phone. The Properties columns lists the properties of the phone that WinRunner checked.

A green mark indicates a passed check. A red mark indicates a failed check.

3 The **Properties** column lists the object properties checked and the results of the check.

- **Name** indicates the name of each object property that WinRunner checked.
- **Arguments** indicates the specified arguments.
- **Expected Value** indicates the expected value of object property that WinRunner checked.
- **Actual Value** indicates the actual value of each object property that WinRunner checked.

4 In the **Properties** column, select a property and click the **Compare Expected and Actual Values** button.

- When viewing the results of a **DisplayText** or **URL** check, the Compare Values box opens, which displays the expected and actual values. For more information refer to the “Analyzing Test Results” chapter in the *WinRunner User’s Guide*.
- When viewing the results of a **WMLSource** check, the WDiff utility opens which displays the expected and actual values and highlights the differences. For more information refer to the “Analyzing Test Results” chapter in the *WinRunner User’s Guide*.

5 Click **OK** to close the GUI Checkpoint Results dialog box.

Guidelines for Working with WAP Add-In Support

This chapter includes the tips and guidelines mentioned throughout the book for working with the WAP Add-in generally and for working with specific emulators.

This chapter describes:

General Guidelines

- You must start WinRunner before you start your WAP emulator. Otherwise, WinRunner may not record and run your test script properly.
- Bitmap checkpoints are not portable between emulators or among devices of the same emulator.

Testing with both the Nokia and Phone.com Emulators Installed

- If both WAP emulators are installed on your machine, the WAP Environment dialog box opens when you start WinRunner with the WAP Add-in loaded. You are prompted to choose a WAP emulator to work with in the current session. If you choose not to display this dialog box again, you can change this setting back to the default by setting the SHOW_WAP_DLG parameter to "1" in the *wrun.ini* file.
- If you want to run the same test on both the Nokia and Phone.com emulators, you should record it on the Nokia emulator.

Testing with the Nokia Emulator

- If you are working with the Nokia emulator in the Global GUI Map File mode, you must save the information that WinRunner learns about each object (the phone screen and keypad keys) in a GUI map file. For more information on GUI map files, refer to the "Understanding the GUI Map" section in the *WinRunner User's Guide.*
- You can use a bitmap checkpoint to check the displayed text in a Nokia phone screen, since the **DisplayText** property check in GUI checkpoints is not supported for the Nokia emulator.
- If you are working with the Nokia emulator in the Global GUI Map File mode, you must load the appropriate GUI map files before you run your tests. For more information, refer to the "Understanding the GUI Map" section in the *WinRunner User's Guide*.

Books Online
Find
Find Again
Help
Top of Chapter
Back

Testing with the Phone.com Emulator

- Do not resize the Phone.com emulator window.
- When testing with the Phone.com emulator, you must work in the Global GUI Map File mode. If you work in the GUI Map File per Test mode, WinRunner cannot record or run tests properly.
- If you plan to record and run tests on multiple Phone.com devices within a single WinRunner session, you must use the **phone_GUI_load** function to change the loaded GUI map file whenever you change the Phone.com device. For additional information, see **page 23**.
- When working with the Phone.com emulator, the GUI map files are automatically managed by WinRunner. For more information on GUI map files, refer to the "Understanding the GUI Map" section in the *WinRunner User's Guide.*

Books Online
Find
Find Again
Help
Top of Chapter
Back

- If only the Phone.com emulator is installed on your machine, or if you choose (in the WAP Environment dialog box) to work with Phone.com in a WinRunner session, then when you work with the WAP Add-in, WinRunner automatically loads a GUI map file. Each Phone.com device has a corresponding GUI map file. Note that if you change the Phone.com configuration during a WinRunner session, you must use the **phone_GUI_load** function to load the GUI map file for the new configuration. If no device name is specified, this function automatically detects the active device and loads the appropriate GUI map file.

  The name of the device is the first word in the emulator window title. It is also the name of the Phone.com file with the model configuration. For example, to load the Alcatel One Touch Pocket phone device, use the ALAV name:

  ```
  phone_GUI_load ( "ALAV" );
  ```

- If your test does not run properly, it may not be properly synchronized. To solve this problem, you can insert a **phone_sync** statement immediately following any statement in which synchronization is a problem. For additional information on the **phone_sync** function, see **Enhancing the WAP Add-in Scripts with TSL** on page 20 or refer to the *TSL Online Reference*. In addition, in the Advanced Run Options dialog box, you should change the value of the **Delay between execution of CS statements** option from 0 to 1000 milliseconds. For additional information on this option, refer to the “Setting Global Testing Options” chapter in the *WinRunner User’s Guide*.

Index

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Books Online
Find
Find Again
Help
Top of Chapter
Back

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

WinRunner - Testing WAP Applications, Version 7.0 Beta

If you have any comments or suggestions regarding this document, please send them via e-mail to documentation@mercury.co.il.

Mercury Interactive Corporation
1325 Borregas Avenue
Sunnyvale, CA 94089
Tel. (408) 822-5200 (800) TEST-911
Fax. (408) 822-5300

WRWAPUG7.0BETA/01