



WinRunner® 7.0

Testing for Year 2000



Table of Contents

Chapter 1: Introduction	4
Converting Applications for the Year 2000	5
Testing with WinRunner with Year 2000 Add-in Support	7
Assessment Testing Workflow	10
Year 2000 Compliance Testing Workflow	11
Chapter 2: Creating Year 2000 Tests	13
About Creating Year 2000 Tests	14
Setting Date Formats.....	17
Recording a Year 2000 Test Script	19
Checking Dates in a Terminal Emulator Application	20
Checking Dates in GUI Objects.....	23
Enhancing Year 2000 Test Scripts with TSL.....	27
Chapter 3: Running Year 2000 Tests	30
About Running Year 2000 Tests	31
Setting the Year 2000 Run Mode	33
Running Year 2000 Tests.....	35
Changing Year 2000 Run Mode Settings with TSL.....	37

 Books Online

 Find

 Find Again

 Help



 Top of Chapter

 Back

Chapter 4: Overriding Date Settings.....	39
About Overriding Date Settings	40
Overriding Aging of Specific Date Formats	41
Overriding Aging or Date Format of an Object	44
Overriding Date Formats and Aging with TSL.....	48
Chapter 5: Viewing Year 2000 Test Results	50
About Viewing Year 2000 Test Results	51
Viewing the Results of a Date Checkpoint	52
Viewing the Results of a GUI Checkpoint.....	57
Index	60



Introduction

Welcome to WinRunner with Year 2000 Add-in Support, Mercury Interactive's automated Year 2000 testing tool for Graphical User Interface (GUI) applications, and mainframe and AS/400 applications running on 3270, 5250, and VT100 protocol terminal emulators.

This guide describes how to test your application for Year 2000 compliance. If you are testing a GUI application, it is recommend that you first review the *WinRunner User's Guide*. If you are testing a terminal emulator application, first review the *WinRunner User's Guide* and the *Testing Terminal Emulator Applications guide*.

This chapter describes:

- **Converting Applications for the Year 2000**
- **Testing with WinRunner with Year 2000 Add-in Support**
- **Assessment Testing Workflow**
- **Year 2000 Compliance Testing Workflow**



Converting Applications for the Year 2000

In the past, programmers wrote applications using two-character fields to manipulate and store dates (for example, '75' represented 1975). Using a two-character date conserved memory and improved application performance at a time when memory and processing power were expensive.

Many of these applications are still in use today, and will continue to be in use well into the 21st century. In industries where age calculation is routinely performed, such as banking and insurance, applications using the two-character date format will generate serious errors after December 31, 1999.

For example, suppose in the year 2001 an insurance application attempts to calculate a person's current age by subtracting his birth date from the current date. If the application uses the two-character date format, a negative age will result (Age = 01 - 30 years = -29).



In order to ensure that applications can accurately process date information in the 21st century, programmers must examine millions of code lines to find date-related functions. Each instance of a two-character date format must be corrected using one of the following methods:

- **Windowing**

Programmers keep the two-character date format, but define thresholds (cut-year points) that will determine when the application recognizes that a date belongs to the 21st century. For example, if 60 is selected as the threshold, the application recognizes all dates from 0 to 59 as 21st century dates. All dates from 60 to 99 are recognized as 20th century dates.

- **Date Field Expansion**

Programmers expand two-character date formats to four-characters. For example, “98” is expanded to “1998”.

However, programmers cannot simply convert code and put an application back into operation. It is crucial that applications undergo comprehensive testing to detect any date-related defects before the year 2000.



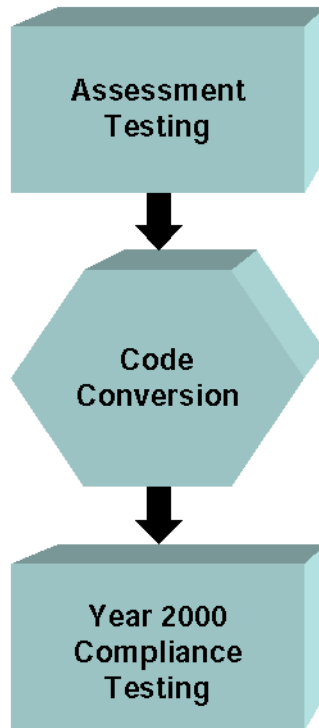
Testing with WinRunner with Year 2000 Add-in Support


WinRunner with Year 2000 Add-in Support enables you to create automated tests that check date information in GUI applications and terminal emulator applications. For GUI applications, it records your actions in terms of the GUI objects (such as windows, lists, and buttons) you select. For terminal emulator applications, it records the operations you perform in the context of screens, fields, and PF keys. As you record, WinRunner enables you to insert checkpoints into your test script that check the dates in the application.

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



WinRunner assists in two phases of the Year 2000 conversion process: assessment testing and Year 2000 compliance testing.



-  Books Online
-  Find
-  Find Again
-  Help
- 
-  Top of Chapter
-  Back

Assessment Testing

Assessment testing helps you identify Year 2000 problems in your application before converting the application. You create a baseline of tests containing checkpoints that check the dates on each screen. To create a test, you record normal business processes performed by your application.

Afterwards, you use *aging* to simulate how the application would process date information after December 31, 1999.

For more information, see [Assessment Testing Workflow](#) on page 10.

Year 2000 Compliance Testing

Year 2000 compliance testing enables you to identify defects in your application after conversion.

- If programmers use the Windowing conversion method, you simply rerun the tests using aging.
- If programmers use the Date Field Expansion method, you first run the tests created during assessment testing using the new date formats. WinRunner translates all the original dates in the test scripts to the new date format. For example, it could translate all MM/DD/YY dates to MM/DD/YYYY.

After verifying that your tests run successfully using the new format, you then run the tests again using aging to simulate conditions after December 31, 1999.

For more information, see [Year 2000 Compliance Testing Workflow](#) on page 11.



Assessment Testing Workflow

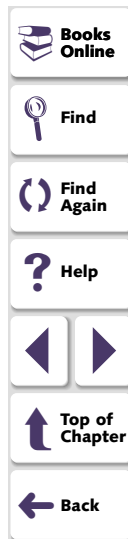
Assessment testing helps you locate date-related problems in your application before you start the Year 2000 conversion process.

The recommended workflow is as follows:

- 1 Define the date format(s) currently used in your application, for example, DD/MM/YY. For more information, see Chapter 2, [Creating Year 2000 Tests](#).
- 2 Create baseline tests. Record tests on the current version of the application (pre-year 2000 conversion). While recording, insert checkpoints that will check the dates in the application. For more information, see Chapter 2, [Creating Year 2000 Tests](#).
- 3 Run the tests (in Debug mode) to check that they run smoothly. For more information, see Chapter 3, [Running Year 2000 Tests](#).

If a test incorrectly identifies non-date fields as date fields or reads a date field using the wrong date format, you can override the automatic date recognition on selected fields. For more information, see Chapter 4, [Overriding Date Settings](#).

- 4 Run the test (in Update mode) to create expected results. For more information, see Chapter 3, [Running Year 2000 Tests](#).
- 5 Run the tests with aging (in Verify mode) to simulate conditions in the 21st century. For more information, see Chapter 3, [Running Year 2000 Tests](#).
- 6 Analyze test results to pinpoint where date-related problems exist in the application. For more information, see Chapter 5, [Viewing Year 2000 Test Results](#).



Year 2000 Compliance Testing Workflow

Year 2000 compliance testing enables you to identify defects in your application after conversion.

Year 2000 Compliance Testing for Date Field Expansion

If programmers use the Date Field Expansion method to convert an application, the recommended workflow is as follows:

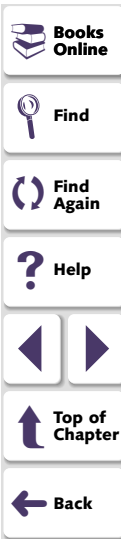
- 1 Define the new date format(s) appearing in the application after conversion, for example, MM/DD/YYYY. For more information, see Chapter 2, [Creating Year 2000 Tests](#).
- 2 Run the tests (in Debug mode) to check that they run smoothly using the new date format(s). For more information, see Chapter 3, [Running Year 2000 Tests](#).
- 3 Run tests (in Verify mode) with the new date formats *and* aging to simulate conditions in the 21st century. For more information, see Chapter 3, [Running Year 2000 Tests](#).
- 4 Analyze test results to pinpoint where date-related problems still exist in the application. For more information, see Chapter 5, [Viewing Year 2000 Test Results](#).



Year 2000 Compliance Testing for Windowing

If programmers use the Windowing method to convert an application, the recommended workflow is as follows:

- 1 Run the tests created during the assessment testing phase with aging to simulate dates in the 21st century. For more information, see Chapter 3, [Running Year 2000 Tests](#).
- 2 Analyze test results to pinpoint where date-related problems still exist. For more information, see Chapter 5, [Viewing Year 2000 Test Results](#).

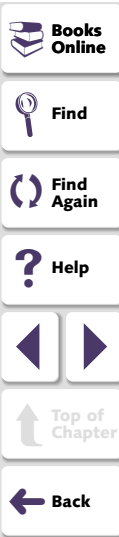


Creating Year 2000 Tests

You start the Year 2000 testing process by creating a baseline of automated tests and adding checkpoints to the test scripts to verify dates.

This chapter describes:

- **Setting Date Formats**
- **Recording a Year 2000 Test Script**
- **Checking Dates in a Terminal Emulator Application**
- **Checking Dates in GUI Objects**
- **Enhancing Year 2000 Test Scripts with TSL**



About Creating Year 2000 Tests

In order to test your application, you use recording to create a baseline of automated tests. As you record, WinRunner generates a test script in Mercury Interactive's Test Script Language (TSL). You can use programming to further enhance your recorded test script with additional TSL functions and programming elements.

To check date information in your application, you add checkpoints to your test script. When you add a checkpoint, WinRunner looks for dates in the active window or screen, captures the dates, and stores them as expected results. When you run a test, a checkpoint compares the expected date to the actual date displayed in the application.

Two types of checkpoints are used in Year 2000 test scripts:

- *Date checkpoints* capture date information displayed in the active screen of a terminal emulator application.
- *GUI checkpoints* capture date information in a GUI application.

Note that in order for WinRunner to recognize dates, you must specify the date formats used in your application (such as MMDDYY, DDMMYY or DD/MM/YY) before you add checkpoints to your application.



Sample Test for a Terminal Emulator Application

The following is a sample WinRunner test recorded on a terminal emulator application. The user presses the Enter key in the first screen of the application. WinRunner waits for the screen to change, and the user types the date 12/13/97 in a field. A date checkpoint checks that the date is correct. The comment (#) lines describe each statement in the script.

```
# Activate the terminal emulator window.
win_activate("RUMBA - DEMO");

# Press the Enter key.
TE_send_key (TE_ENTER);

# Wait for the next screen to refresh.
TE_wait_sync();

# Direct input to the Logon screen.
set_window("Logon");

# Type in the current date (Format: MMDDYY).
TE_edit_field("Date of Flight:", "121397");

# Check the date (using a date checkpoint).
Y2K_check_date ("y2k1");
```

For more information on **TE_** functions, refer to the *Testing Terminal Emulator Applications* guide and the *TSL Online Reference*.



Sample Test for a GUI Application

The following is a sample WinRunner test recorded on a GUI application used to reserve airline tickets. The user opens a new ticket order and enters the date of the flight. A GUI checkpoint checks that the date is correct. The comment (#) lines describe each statement in the script.

Direct input to the main Flight Reservation window.

```
set_window ("Flight Reservation", 10);
```

Open a new ticket order.

```
menu_select_item ("File; New Order");
```

Enter the flight date in the Date of Flight field (Format: MMDDYY).

```
edit_set_insert_pos ("Date of Flight:", 0, 0);
```

```
obj_type ("121397");
```

Check the date (using a GUI checkpoint).

```
obj_check_gui("Date of Flight:", "list1.ckl", "gui1", 1);
```

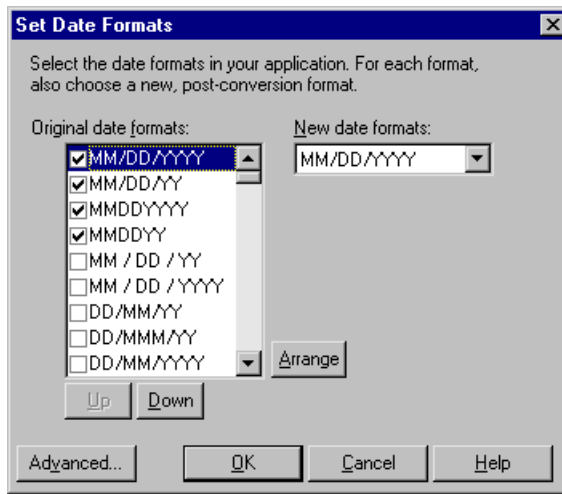


Setting Date Formats

WinRunner supports a wide range of date formats. Before you begin creating tests, you should specify the date formats currently used in your application. This enables WinRunner to recognize date information when you insert checkpoints into a test script and run tests.

To specify date formats:

- 1 Choose **Year2000 > Set Date Formats**. The Set Date Formats dialog box opens.



2 In the **Original Date Formats** list, select the check box next to each date format used in your application.

3 Click **Arrange** to move all selected date formats to the top of the list. You can also use the **Up** and **Down** buttons to rearrange the formats.

Note that you should move the most frequently-used date format in your application to the top of the list. WinRunner considers the top date format first.

4 If you are performing Year 2000 compliance testing, select the post-conversion date format used in your application from the **New Date Formats** list.

Note that if the windowing conversion method was used, the original date format is identical to the new date format.

5 Click **OK**.



Recording a Year 2000 Test Script

You use the Context Sensitive recording mode to create a Year 2000 test script.

To record a Year 2000 test script:



- 1 Choose **Create > Record - Context Sensitive** to start recording, or click the **Record - Context Sensitive** button.
- 2 Use the mouse and keyboard to perform operations in your application.
- 3 While recording, insert checkpoints into the test script to check dates. For more information, see [Checking Dates in a Terminal Emulator Application](#) on page 20, and [Checking Dates in GUI Objects](#) on page 23.



- 4 Choose **Create > Stop Recording** to stop the test, or click the **Stop** button.
- 5 Enhance the recorded test script with additional Year 2000 functions (optional).



- 6 Choose **File > Save**, or click the **Save** button to save the test script.



Checking Dates in a Terminal Emulator Application

You can check all the dates in a terminal emulator screen, or a date in a specific area of the screen. When you run the test, WinRunner compares the original dates captured when you created the test to the actual dates in the application.

Checking All Dates in a Screen

You can manually or automatically add a date checkpoint to your test script that checks all the dates in a screen.

Inserting Date Checkpoints Manually

You can manually add a date checkpoint to your test script while recording. To do so, press the CHECK DATE softkey (Left Ctrl + F2 for AS/400 and VT100 applications and Left Ctrl + PgDn for mainframe applications).

WinRunner inserts the following statement into the test script:

```
Y2K_check_date ( filename );
```

The *filename* parameter is the name of the date checkpoint and is assigned sequentially by WinRunner.

For example, the statement below is the first date checkpoint in the script.

```
Y2K_check_date("y2k1");
```



Note: You can change the softkey configuration of a date checkpoint using the Softkey Configuration utility. For more information, refer to the *WinRunner User's Guide*.

For more information on **Y2K_check_date**, refer to the *TSL Online Reference*.

Inserting Date Checkpoints Automatically

You can set WinRunner to automatically insert a date checkpoint into the test script each time you change screens in your terminal emulator application. To do so, add the following function at the top of a test script:

```
Y2K_set_auto_date_verify ( ON/OFF );
```

For more information on **Y2K_set_auto_date_verify**, refer to the *TSL Online Reference*.

Checking a Date in a Specific Area of the Screen

While recording, you can add a date checkpoint to your test script that checks a specific area of the screen.



To insert a date checkpoint for a specific area:

- 1 Press the CHECK PARTIAL DATE softkey (Left Alt + End for mainframe applications, Left Ctrl + F9 for Extra mainframe applications, and Left Ctrl + F8 for AS/400 and VT100 applications).

WinRunner is minimized, the mouse pointer becomes a crosshairs pointer, and a help window opens.

- 2 Mark the date to be captured: click the left mouse button and drag the mouse pointer until a rectangle encloses the area, then release the mouse button.
- 3 Click the right mouse button to complete the operation. WinRunner is restored and the following statement is inserted in the test script:

```
Y2K_check_date ( filename [ , x1, y1, x2, y2 ] );
```

The *filename* parameter is the name of the date checkpoint.

The [x1,y1,x2,y2] parameter defines the location of the date, relative to the specified window. The coordinate pairs can designate any two diagonally opposite corners of a rectangle. WinRunner searches for the date in the area defined by the rectangle. For example:

```
Y2K_check_date("y2k2", 13, 6, 24, 7);
```

Note: You can change the softkey configuration of a date checkpoint using the Softkey Configuration utility. For more information, refer to the *WinRunner User's Guide*.



Checking Dates in GUI Objects

You can use GUI checkpoints to check dates in GUI objects (such as edit boxes or static text fields). In addition you can check dates in the contents of PowerBuilder, Visual Basic, and ActiveX control tables.

When you create a GUI checkpoint, you can use the default check for an object or you can specify which properties to check. The default check for edit boxes and static text fields is the date. For a table, the default check performs a case-sensitive check on the entire contents of the table, and checks all the dates in the table.

Note that you can also use the **Create > GUI Checkpoints > For Multiple Objects** command to check multiple objects in window. For more information about this command, refer to Chapter 9, “Checking GUI Objects” in the *WinRunner User’s Guide*.

Checking Dates with the Default Check

You can use the default check to check dates in edit boxes, static text fields, and table contents.



To check the date in a GUI object:



- 1 Choose **Create > GUI Checkpoint > For Object/Window**, or click the **GUI Checkpoint for Object/Window** button on the User toolbar.

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

- 2 Click the object containing the date.
- 3 WinRunner captures the current date and stores it in the test's expected results folder. If you click in a table, WinRunner also captures the table contents. The WinRunner window is restored and a GUI checkpoint is inserted into the test script as an **obj_check_gui** statement. For more information on **obj_check_gui**, refer to the *TSL Online Reference*.

For additional information on creating GUI checkpoints, refer to Chapter 9, "Checking GUI Objects" and Chapter 12, "Checking Table Contents" in the *WinRunner User's Guide*.

Creating a GUI Checkpoint by Specifying which Properties to Check

You can create a GUI Checkpoint by specifying which properties of an object to check.



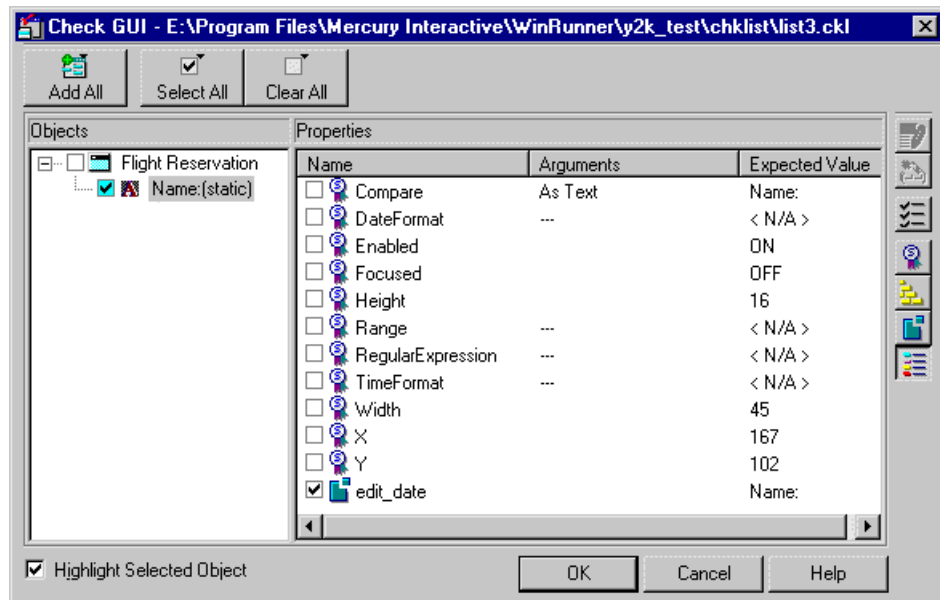
To create a GUI Checkpoint by specifying which properties to check:



- 1 Choose **Create > GUI Checkpoint > For Object/Window**, or click the **GUI Checkpoint for Object/Window** button on the User toolbar.

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

- 2 Double-click the object containing the date. The Check GUI dialog box opens.



- 3 Highlight the object name in the **Objects** pane. The **Properties** pane lists all the properties for the selected object.

Note that if you chose an edit box or a static text field, the date check is listed in the **Properties** column as **edit_date**. If you chose a table, the date check is listed as **Y2K Verification**.

- 4 Select the properties you want to check. For more information on selecting properties, refer to Chapter 9, “Checking GUI Objects” and Chapter 12, “Checking Table Contents” in the *WinRunner User’s Guide*.



Note that you can edit the expected value of a property. To do so, first select it in the **Properties** column. Next either click the **Edit Expected Value** button, or double-click the value in the **Expected Value** column. For an edit box or a static text field, an edit field opens in the Expected Value column, where you can change the value. For a table, the Edit Check dialog box opens. In the **Edit Expected Data** tab, edit the table contents.

- 5 Click **OK** to close the Check GUI dialog box.

An **obj_check_gui** statement is inserted into your test script. For more information on the **obj_check_gui** function, refer to the *TSL Online Reference*.



Enhancing Year 2000 Test Scripts with TSL

You can enhance your recorded Year 2000 test scripts by adding the following TSL functions:

- The **Y2K_calc_days_in_field** function calculates the number of days between two date fields. It has the following syntax:

```
Y2K_calc_days_in_field ( field_name1, field_name2 );
```

- The **Y2K_calc_days_in_string** function calculates the number of days between two numeric strings. It has the following syntax:

```
Y2K_calc_days_in_string ( string1, string2 );
```

- The **Y2K_field_to_Julian** function translates the contents of a date field to a Julian number. It has the following syntax:

```
Y2K_field_to_Julian ( date_field );
```

- The **Y2K_is_date_field** function determines whether a field contains a valid date. It has the following syntax:

```
Y2K_is_date_field ( field_name, min_year, max_year );
```



- The **Y2K_is_date_string** function determines whether a numeric string contains a valid date. It has the following syntax:

```
Y2K_is_date_string ( string, min_year, max_year );
```

- The **Y2K_is_leap_year** function determines whether a year is a leap year. It has the following syntax:

```
Y2K_is_leap_year ( year );
```

- The **Y2K_month_language** function sets the language used for month names. It has the following syntax:

```
Y2K_month_language ( language );
```

- The **Y2K_set_attr** function sets the record configuration mode for a field in a terminal emulator application. It has the following syntax:

```
Y2K_set_attr ( index );
```

- The **Y2K_set_capture_mode** function determines how WinRunner captures dates in terminal emulator applications. It has the following syntax:

```
Y2K_set_capture_mode ( mode );
```



- The **Y2K_string_to_Julian** function translates the contents of a date string to a Julian number. It has the following syntax:

Y2K_string_to_Julian (*string*);

For more information on Year 2000 TSL functions, refer to the *TSL Online Reference*.



Running Year 2000 Tests

Once you have developed a baseline of tests, you run the tests to check how your application calculates date information.

This chapter describes:

- **Setting the Year 2000 Run Mode**
- **Running Year 2000 Tests**
- **Changing Year 2000 Run Mode Settings with TSL**



About Running Year 2000 Tests

When you run a Year 2000 test, WinRunner interprets the test script line-by-line and performs the required operations on your application. At each checkpoint in the test script, it compares the expected dates with the actual dates in your application.

To run a Year 2000 test script, you first specify Year 2000 settings and the general run mode of the script.

Year 2000 run mode settings specify:

- *Date format*, to determine whether to use the script's original date formats or to convert dates to new formats. Usually, you will modify dates formats during Year 2000 compliance testing—if the application was converted using date field expansion.
- *Aging*, to determine whether or not to age the dates in the script. Usually you age dates in tests during Year 2000 compliance testing, to check that the converted application supports Year 2000 dates.

You can age dates incrementally (by specifying the years, months, and days by which you want to age the dates) or statically (by defining a specific date).



The general run mode settings are:

- *Verify* mode enables you to check your application. WinRunner compares the current response of your application to its expected response. Any discrepancies between the current and expected responses are captured and saved as verification results. Each time you run the script, a new verification results folder is created.
- *Debug* mode helps you debug the test script. Running a test in Debug mode is the same as running a test in Verify mode, except that debug results are always saved in the *debug* directory, overwriting the previous results.
- *Update* mode enables you to update the expected results of a test. Note that during a test run in Update mode, dates in the script are not aged or translated to a new format.

For more information about the general run modes, refer to the *WinRunner User's Guide*.

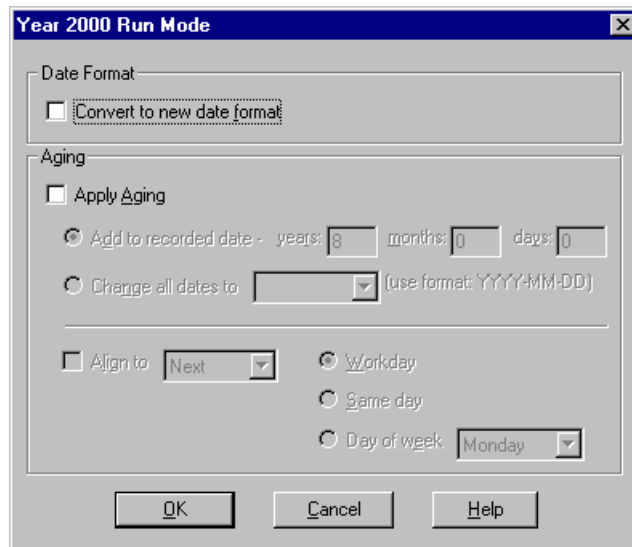


Setting the Year 2000 Run Mode

Before you run a Year 2000 test script, you set the Year 2000 run mode.

To set the Year 2000 run mode:

- 1 Choose **Year2000 > Run Mode**. The Year 2000 Run Mode dialog box opens.



You can also reach this dialog box from the Run Year 2000 Test dialog box. For more information, see [Running Year 2000 Tests](#) on page 35.



- 2 If you are running the test on an application that was converted to a new date format, select the **Convert to New Date Format** check box.
- 3 If you want to run the test with aging, select the **Apply Aging** check box and do one of the following:
 - To increment all dates, click **Add to Recorded Date** and specify the years, months or days. You can also align dates to a particular day by clicking the **Align to** check box and specifying the day.
 - To change all dates to a specific date, click **Change all dates to** and select a date from the list.
- 4 Click **OK**.

Note: When you run a test, you can override the options you specified in the Year 2000 Run Mode dialog box. For more information, see Chapter 4, [Overriding Date Settings](#).



Running Year 2000 Tests

After you set the Year 2000 run mode, you can run your Year 2000 test script.

To run a Year 2000 test:

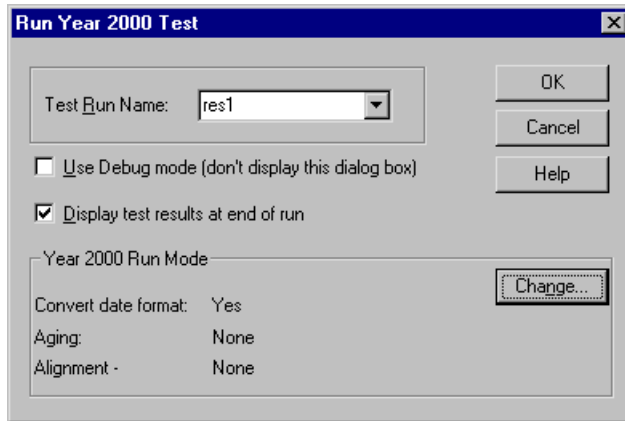
- 1 If the Year 2000 test is not already open, open it.
- 2 Choose a general run mode (Verify, Debug, or Update) from the dropdown list of modes on the Standard toolbar.
- 3 Choose the appropriate **Run** menu command or click one of the **Run** buttons.
For more information on Run commands, refer to the *WinRunner User's Guide*.



Note that in *Update* mode, dates in the script are not aged or translated to a new format. In *Debug* mode the test script immediately starts to run using the Year 2000 run mode settings defined in the Year 2000 Run Mode dialog box.



If you selected *Verify* mode, the Run Year 2000 Test dialog box opens.



- 4 Assign a name to the test run. Use the default name appearing in the **Test Run Name** field, or type in a new name.
- 5 If you would like to change the Year 2000 run mode settings, click **Change** and specify the Year 2000 run mode settings.
- 6 Click **OK** to close the dialog box and run the test.



Changing Year 2000 Run Mode Settings with TSL

You can set conditions for running Year 2000 test using the following TSL functions:

- The **Y2K_disable_format** function disables a date format. It has the following syntax:

```
Y2K_disable_format ( format );
```

- The **Y2K_enable_format** function enables a date format. It has the following syntax:

```
Y2K_enable_format ( format );
```

- The **Y2K_leading_zero** function determines whether to add a zero before single-digit numbers when aging and translating dates. It has the following syntax:

```
Y2K_leading_zero ( mode );
```

- The **Y2K_set_aging** function ages the test script. It has the following syntax:

```
Y2K_set_aging ( format, type, days, months, years );
```



- The **Y2K_set_replay_mode** function sets the Year 2000 run mode. It has the following syntax:

```
Y2K_set_replay_mode ( mode );
```

- The **Y2K_set_year_limits** function sets the minimum and maximum years valid for date verification and aging. It has the following syntax:

```
Y2K_set_year_limits ( min_year, max_year );
```

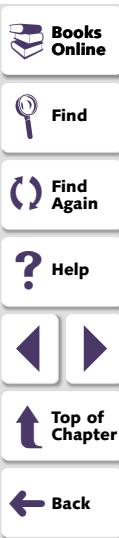
- The **Y2K_set_year_threshold** function sets the year threshold (cut-year point). If the threshold is 60, all years from 60 to 99 are recognized as 20th century dates and all dates from 0 to 59 are recognized as 21st century dates. This function has the following syntax:

```
Y2K_set_year_threshold ( number );
```

- The **Y2K_align_day** function ages dates to a specified day of the week or type of day. It has the following syntax:

```
Y2K_align_day ( align_mode, day_in_week );
```

For more information on Year 2000 TSL functions, refer to the *TSL Online Reference*.



Overriding Date Settings

You can override how WinRunner identifies and ages selected dates.

This chapter describes:

- **Overriding Aging of Specific Date Formats**
- **Overriding Aging or Date Format of an Object**
- **Overriding Date Formats and Aging with TSL**



About Overriding Date Settings

As you debug your tests, you may want to override how WinRunner identifies or ages specific date fields in your application. You can override the following:

- *Aging of a specific date format.* You can define that a specific date format (for example, MM/DD/YY) will be aged differently than the default aging applied to other date formats.
- *Aging or date format of a specific object.* You can define that a specific object that resembles a date (for example, a catalog number such as 123172) will not be treated as a date object. You can specify that a specific date object (such as a birth date) will not be aged. Or, you can define that a specific object will be assigned a different date format than that of the default.

When WinRunner runs tests, it first examines the general settings defined in the Year 2000 Run Mode window. Then, it examines the aging overrides for specific date formats. Finally, it considers overrides defined for particular objects.

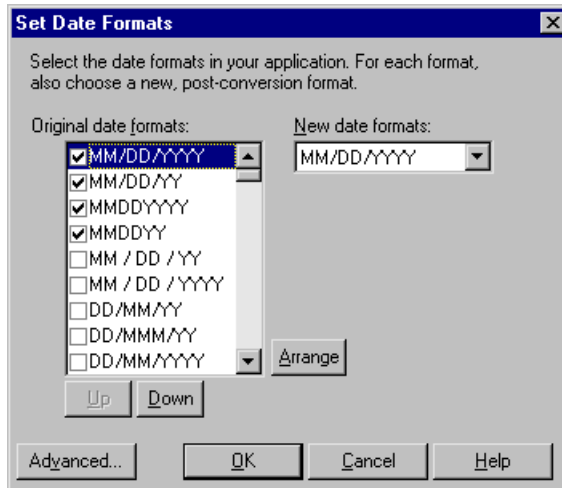


Overriding Aging of Specific Date Formats

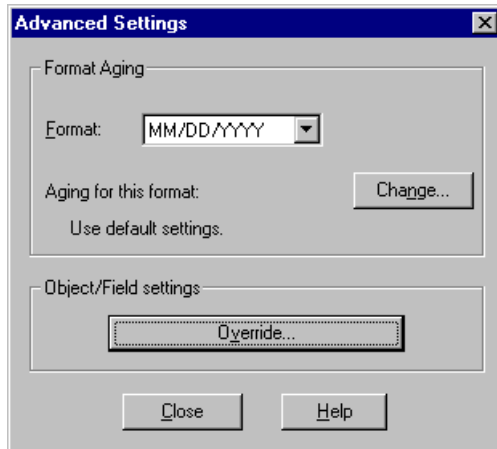
You can override the aging of a specific date format so that it will be aged differently than the default aging setting.

To override the aging of a date format:

- 1 Choose **Year2000 > Set Date Formats**. The Set Date Formats dialog box opens.

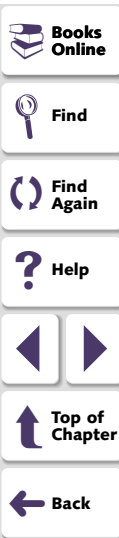


- 2 Click the **Advanced** button. The Advanced Settings dialog box opens.

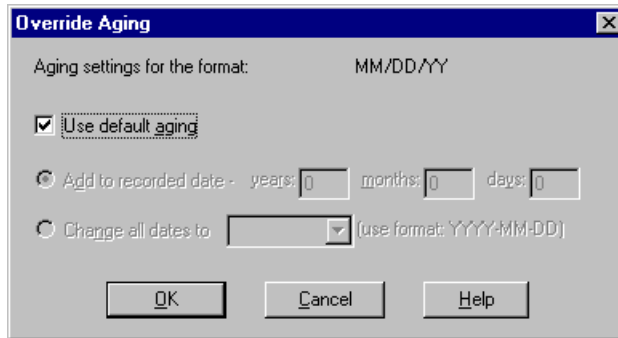


- 3 In the **Format** list, select a date format.

Note that the Format list displays only the date formats that are checked in the Set Date Formats dialog box.



- 4 Click **Change**. The Override Aging dialog box opens.



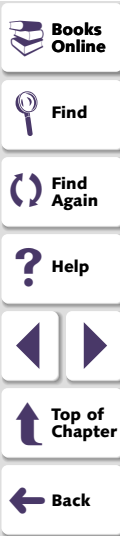
- 5 Clear the **Use default aging** check box and select one of the following:
- To increment the date format by a specific number of years, months, and days, select the **Add to recorded date** option. To specify no aging for the date format, use the default value of 0.
 - To choose a specific date for the selected date format, select **Change all dates to**, and choose a date from the list.
- 6 Click **OK** to close the Override Aging dialog box.



Overriding Aging or Date Format of an Object

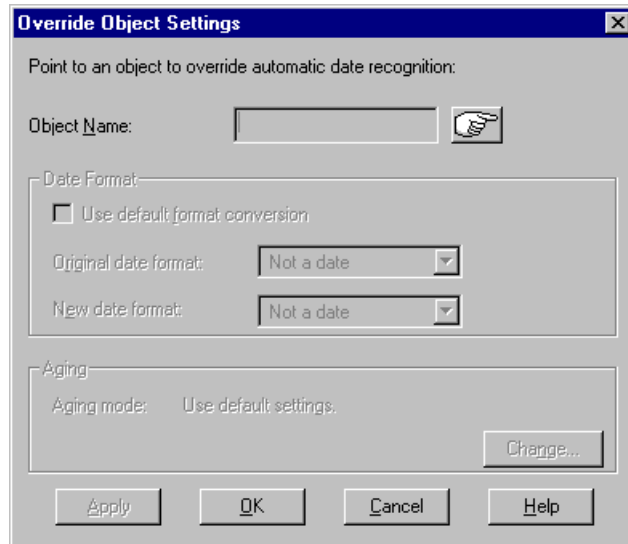
For any specific object, you can override the default settings and specify that:

- the object should not be treated like a date object
- the object should be aged differently
- the object should be converted to a different date format



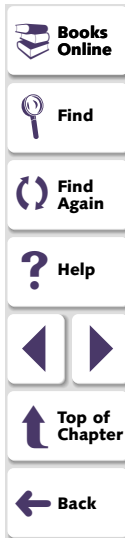
To override settings for an object:

- 1 Choose **Year2000 > Override Object Settings**. The Override Object Settings dialog box opens.

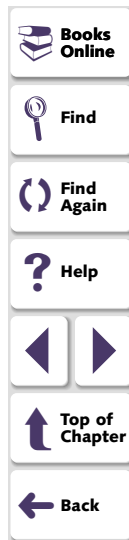
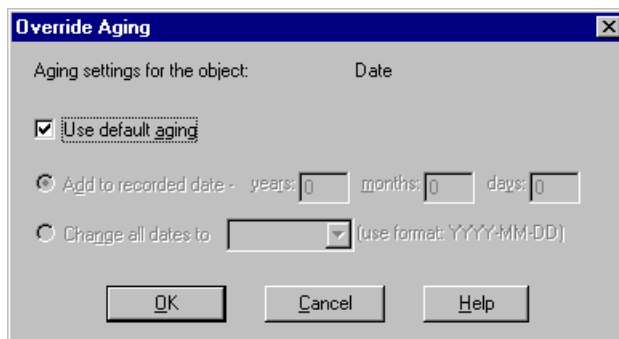


- 2 Click the pointing hand button and then click the date object.

WinRunner displays the name of the selected date object in the **Object Name** box.



- 3 To override date format settings or to specify that the object is not a date object, clear the **Use default format conversion** check box and do one of the following:
 - To specify that the object should not be treated like a date object, select **Not a Date** in the **Original date format** field and in the **New date format** field.
 - To override the date format assigned to the object, select the object's original date format and its new date format in the respective fields.
- 4 To override the aging applied to the object, click **Change**. The Override Aging dialog box opens.



- 5 Clear the **Use Default Aging** check box and do one of the following:
 - To increment the date format by a specific number of years, months, and days, select the **Add to recorded date** option. To specify no aging for the date format, use the default value of 0.
 - To choose a specific date for the selected date format, select **Change all dates to**, and choose a date from the list.
- 6 Click **OK** to close the Override Aging dialog box.
- 7 In the Override Object Settings dialog box, click **Apply** to override additional date objects, or click **OK** to close the dialog box.



Overriding Date Formats and Aging with TSL

You can override dates in a Year 2000 test script using the following TSL functions:

- The **Y2K_age_string** function ages a date string. It has the following syntax:

```
Y2K_age_string ( date, years, month, days, output );
```

- The **Y2K_align_day** function ages dates to a specified day of the week or type of day. It has the following syntax:

```
Y2K_align_day ( align_mode, day_in_week );
```

- The **Y2K_change_original_new_formats** function overrides the date format for a date object. It has the following syntax:

```
Y2K_change_original_new_formats ( field name, original format, new format  
[, TRUE/FALSE ] );
```

- The **Y2K_change_field_aging** function overrides the aging applied to the specified date object. It has the following syntax:

```
Y2K_change_field_aging ( field name, aging type, days, months, years );
```

- The **Y2K_set_aging** function ages the test script. It has the following syntax:

```
Y2K_set_aging ( format, type, days, months, years );
```



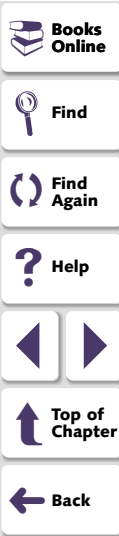
- The **Y2K_set_system_date** function sets the system date and time.

```
Y2K_set_system_date ( year, month, day [, day, minute, second] );
```

- The **Y2K_type_mode** function disables overriding of automatic date recognition for all date objects in a GUI application.

```
Y2K_type_mode ( mode );
```

For more information on Year 2000 TSL functions, refer to the *TSL Online Reference*.

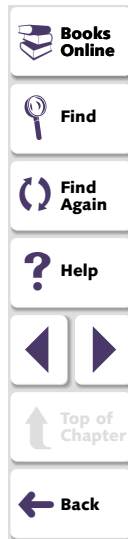


Viewing Year 2000 Test Results

After you run a Year 2000 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:

- **Viewing the Results of a Date Checkpoint**
- **Viewing the Results of a GUI Checkpoint**



About Viewing Year 2000 Test Results

When a test run is completed, you can view detailed test results in the WinRunner Test Results window. The report contains a description of the major events that occurred during the test run, such as date checkpoints and GUI checkpoints. It also includes tables and pictures to help you quickly analyze the test results.

For more information on the WinRunner Test Results window, refer to the *WinRunner User's Guide*.



Viewing the Results of a Date Checkpoint

Date checkpoints capture information about date fields in a screen of a terminal emulator application. When you run a Year 2000 test, WinRunner 2000 compares the expected dates with the actual dates in the application. After you run a test, you can view the date checkpoint results in a table.



To view the results of a date checkpoint:



- 1 Choose **Tools > Test Results** or click the **Test Results** button in the main WinRunner window. In the test log, look for “check date” entries in the **Event** column. Passed date checkpoints appear in green; failed date checkpoints appear in red.

The screenshot shows the WinRunner Test Results window for a test named 'y2k_test'. The window title is 'WinRunner Test Results - [E:\Program Files\Mercury Interactive\WinRunner\y2k_test]'. The main area displays a tree view with a red 'X' icon and the text 'Test Result: fail'. Below this, there are two green checkmarks indicating that the total number of bitmap and GUI checkpoints are both 0. A yellow warning icon is present next to the 'General Information' section.

Line	Event	Details	Result	Time
1	start run	y2k_test	run	00:00:00
2	check date	y2k1	mismatch	00:00:00
3	stop run	y2k_test	fail	00:00:03



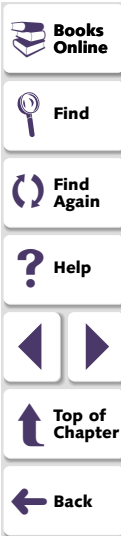


- 2 Double-click a “check date” entry in the test log. Alternatively, highlight the entry and choose **Options > Display** or click the **Display** button. The Y2K Date Verify Viewer opens:

	result	field name	expected - captured	expected - translated	actual
1	✗	Date	100297	100202	100297
2	✓	Approved	10/26/85	10/26/90	10/26/90

For each date, the following information is displayed:

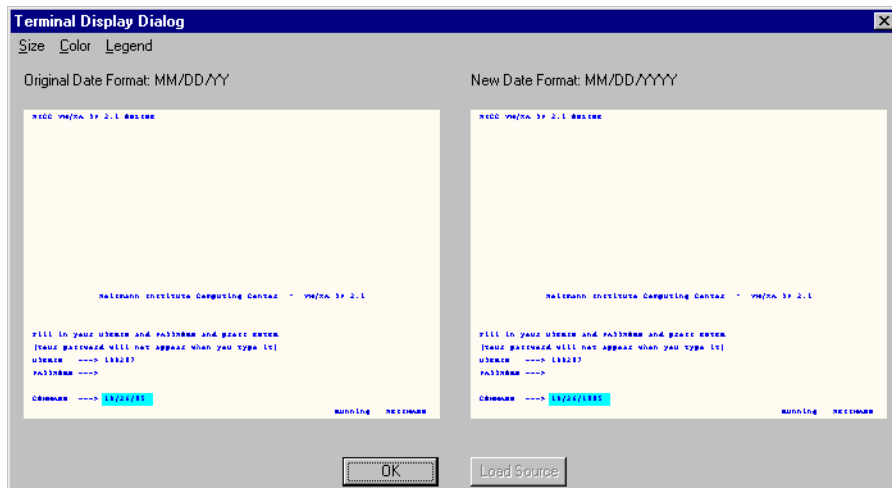
- **result:** Result of the check on the field (pass or fail).
- **field name:** Name of the date field checked.
- **expected - captured:** Original date captured. Note that you can edit this field by clicking the right mouse button.



- **expected - translated:** Date after a new date format and/or aging was applied.
- **actual:** Actual date appearing in the application.



- 3 To view the screen and field containing the captured date, double-click a check. The Terminal Display dialog box opens:



The **Original Date Format** screen highlights the date in its original format. The **New Date Format** screen highlights the date in its new format.

- To view the Terminal Display dialog box in another size, choose an option on the **Size** menu.
- To view the text in the Terminal Display dialog box in another color, choose an option from the **Color** menu.

Click **OK** to close the dialog box.

- 4 Choose **File > Exit** to close the Y2K Date Verify Viewer.



Viewing the Results of a GUI Checkpoint

GUI checkpoints enable you to check GUI objects in your application. When you run a Year 2000 test, WinRunner compares the expected date with the actual date in the application.

To view the results of a GUI checkpoint:

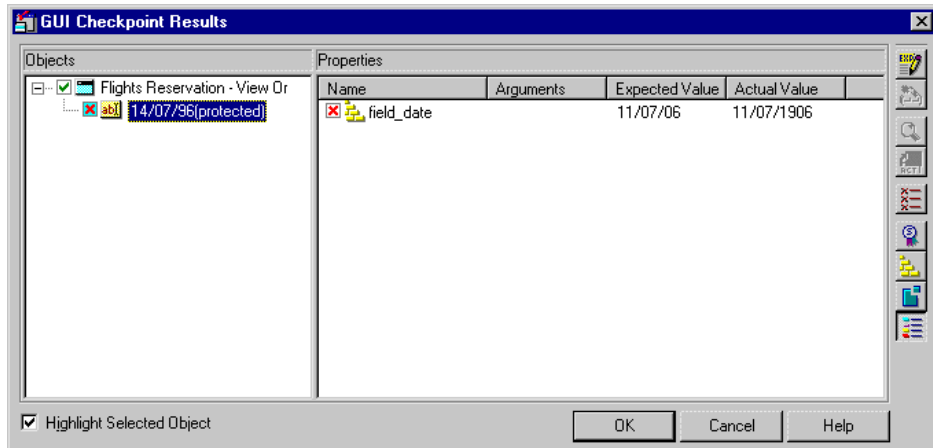


- 1 Choose **Tools > Test Results**, or click the **Test Results** button in the main WinRunner window. In the test log, look for “end GUI checkpoint” entries in the **Event** column. Passed date checkpoints appear in green; failed date checkpoints appear in red.

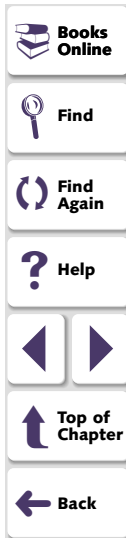




- 2 Double-click an “end GUI checkpoint” entry in the test log. Alternatively, highlight the entry and choose **Options > Display** or click the **Display** button. The GUI Checkpoint Results dialog box opens, listing the results of the selected GUI checkpoint.

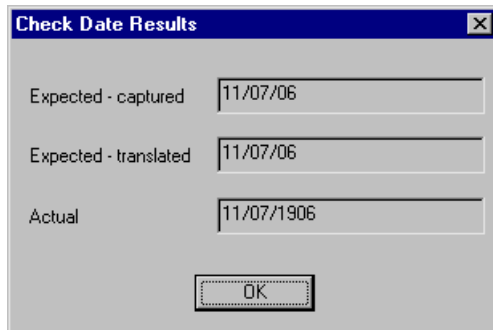


The **Object** column lists the objects that were checked. Each check is marked as either passed or failed. The expected and the actual results are displayed in the right columns.





- 3 To view detailed information about a check on a date, double-click the check or click the **Compare Expected and Actual Values** button. The Check Date Results box dialog box opens.



The Check Date Results box lists the original expected date, the expected date after aging and translation, and the actual date appearing in the object.

Click **OK** to close the dialog box.

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



Index

A

- Advanced Settings dialog box 42
- aging
 - defining 34
 - overriding 39–49
- alignment, setting 34
- Assessment testing
 - defined 9
 - workflow 10

C

- Check Date Results dialog box 59
- Check GUI dialog box 25
- checking dates
 - in edit boxes 23
 - in static text fields 23
 - in table contents 23
- creating tests 13–29
- cut-year points 6, 38

D

- date checkpoints
 - for a specific area 21
 - for all dates in a screen 20
 - viewing results 52

- date field expansion 6
- date formats
 - overriding 41
 - setting 17
 - Year 2000 run mode 31
- Debug mode 32

E

- edit_date property check 26

G

- GUI Checkpoint Results dialog box 58
- GUI checkpoints, creating 23
 - for multiple objects 23
 - specifying properties to check 24
 - using the default check 23
- GUI checkpoints, viewing results 57

I

- incremental aging 31
- introduction 4–12

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



O

- obj_check_gui function [24](#), [26](#)
- Override Aging dialog box [43](#), [46](#)
- Override Object Settings dialog box [45](#)
- overriding date formats [41](#)
- overriding date objects [44](#)
- overriding date settings [39–49](#)

R

- recording tests [19](#)
- results, viewing [50–59](#)
 - date checkpoints [52](#)
 - GUI checkpoints [57](#)
- Run Year 2000 Test dialog box [36](#)
- Running Year 2000 tests [30–38](#)

S

- Set Date Formats dialog box [17](#)
- setting date formats [17](#)
- setting Year 2000 run mode [33](#)
- static aging [31](#)

T

- Terminal Display dialog box [55](#)
- threshold [6](#), [38](#)

U

- Update mode [32](#)

V

- Verify mode [32](#)
- viewing results [50–59](#)
 - date checkpoints [52](#)
 - GUI checkpoints [57](#)

W

- windowing [6](#)
- workflow
 - Assessment testing [10](#)
 - Year 2000 compliance testing [12](#)

Y

- Y2K Date Verify Viewer [54](#)
- Y2K Verification property check [26](#)
- Y2K_age_string function [48](#)
- Y2K_align_day function [48](#)
- Y2K_calc_days_in_field function [27](#)
- Y2K_calc_days_in_string function [27](#)
- Y2K_change_field_aging function [48](#)
- Y2K_change_original_new_formats function [48](#)

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



- Y2K_check_date function [20](#), [22](#)
- Y2K_disable_format function [37](#)
- Y2K_enable_format function [37](#)
- Y2K_field_to_Julian function [27](#)
- Y2K_is_date_field function [27](#)
- Y2K_is_date_string function [28](#)
- Y2K_is_leap_year function [28](#)
- Y2K_leading_zero function [37](#)
- Y2K_month_language function [28](#)
- Y2K_set_aging function [37](#), [48](#)
- Y2K_set_attr function [28](#)
- Y2K_set_auto_date_verify function [21](#)
- Y2K_set_capture_mode function [28](#)
- Y2K_set_replay_mode function [38](#)
- Y2K_set_system_date function [49](#)
- Y2K_set_year_limits function [38](#)
- Y2K_set_year_threshold function [38](#)
- Y2K_string_to_Julian function [29](#)
- Y2K_type_mode function [49](#)
- Year 2000 compliance testing
 - defined [9](#)
 - workflow for date field expansion [11](#)
 - workflow for windowing [12](#)
- Year 2000 run mode
 - date format [31](#)
 - setting [33](#)
- Year 2000 Run Mode dialog box [33](#)

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 for Year 2000, Version 7.0

© Copyright 1997 - 2001 by Mercury Interactive Corporation

All rights reserved. All text and figures included in this publication are the exclusive property of Mercury Interactive Corporation, and may not be copied, reproduced, or used in any way without the express permission in writing of Mercury Interactive. Information in this document is subject to change without notice and does not represent a commitment on the part of Mercury Interactive.

Mercury Interactive may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents except as expressly provided in any written license agreement from Mercury Interactive.

WinRunner, XRunner, and LoadRunner are registered trademarks of Mercury Interactive Corporation. Astra, Astra SiteManager, Astra SiteTest, RapidTest, QuickTest, Visual Testing, Action Tracker, Link Doctor, Change Viewer, Dynamic Scan, Fast Scan, and Visual Web Display are trademarks of Mercury Interactive Corporation in the United States and/or other countries.

This document also contains registered trademarks, trademarks and service marks that are owned by their respective companies or organizations. Mercury Interactive Corporation disclaims any responsibility for specifying which marks are owned by which companies or organizations.

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

WRY2KUG7.0/01

