

# QARun GUI Testing

---

## Getting Started Guide

Release 4.6



**COMPUWARE®**

Please direct questions about *QARun*  
or comments on this document to:

***QARun* Technical Support**  
Compuware Corporation  
31440 Northwestern Highway  
Farmington Hills, MI 48334-2564

**1-800-538-7822**

Outside the USA and Canada, please contact  
your local Compuware office or agent.

© 1996-1998 Compuware Corporation. All rights reserved. Unpublished - rights reserved under the Copyright Laws of the United States.

#### U.S. GOVERNMENT RIGHTS

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in Compuware Corporation license agreement and as provided in DFARS 227.7202-1(a) and 227.7202-3(a) (1995), DFARS 252.227-7013(c)(1)(ii)(OCT 1988), FAR 12.212(a) (1995), FAR 52.227-19, or FAR 52.227-14 (ALT III), as applicable. Compuware Corporation.

THIS PRODUCT CONTAINS CONFIDENTIAL INFORMATION AND TRADE SECRETS OF COMPUWARE CORPORATION. USE, DISCLOSURE, OR REPRODUCTION IS PROHIBITED WITHOUT THE PRIOR EXPRESS WRITTEN PERMISSION OF COMPUWARE CORPORATION. ACCESS IS LIMITED TO AUTHORIZED USERS. USE OF THIS PRODUCT IS SUBJECT TO THE TERMS AND CONDITIONS OF THE USER'S LICENSE AGREEMENT WITH COMPUWARE CORPORATION.

Compuware, *QARun*, *QACenter*, *QADirector*, and *QALoad* are trademarks or registered trademarks of Compuware Corporation.

Acrobat<sup>®</sup> Reader copyright © 1987-1998 Adobe Systems Incorporated. All rights reserved. Adobe, Acrobat, and Acrobat Reader are trademarks of Adobe Systems Incorporated.

CICS, OS/2, and REXX are trademarks of International Business Machines Corporation.

All other company or product names are trademarks of their respective owners.

Doc. CWQUGSG4D  
April 12, 1999

# Table of Contents

<b>Introduction .....</b>	<b>vii</b>
Who Should Read This Guide .....	viii
Related Publications .....	viii
World Wide Web Information .....	ix
<i>QARun</i> Terminology .....	ix
Getting Help .....	x
<b>Chapter 1. Introducing QADemo .....</b>	<b>1-1</b>
The Company Profile .....	1-2
The Target Application — QADemo .....	1-2
Logging On and Selecting Main Options.....	1-2
QADemo's Main Options .....	1-4
<b>Chapter 2. Building the Driver Script .....</b>	<b>2-1</b>
Exercise 1 — Creating a Driver Script.....	2-2
Starting <i>QARun</i> .....	2-2
Changing Configuration Options .....	2-3
Getting Started .....	2-5
Learning the Script.....	2-5
Stop Learning the Script.....	2-7
The Resulting Script.....	2-7
Understanding the Script.....	2-8
Running the Script.....	2-10
Exercise Summary .....	2-10
<b>Chapter 3. Building Test Scripts.....</b>	<b>3-1</b>
Exercise 2 — Creating List, Window, and Menu Checks.....	3-2
Getting Started .....	3-2
Learning the Script.....	3-3
Inserting Checks.....	3-3
Continuing the Script .....	3-10
The Resulting Script.....	3-11
Running the Script.....	3-11
Analyzing the Results .....	3-14
Exercise Summary .....	3-15

Exercise 3 — Creating Form Checks .....	3-15
Changing Configuration Options .....	3-15
Getting Started .....	3-16
Learning the Script .....	3-17
Continuing the Script .....	3-20
Running the Script.....	3-20
Exercise Summary .....	3-21
Exercise 4 — Creating Clock Checks To Test Performance .....	3-21
Getting Started .....	3-21
Learning the Script.....	3-22
Continuing the Script .....	3-27
The Resulting Script.....	3-28
Modifying the Script .....	3-28
Additional Script Modifications.....	3-30
Running the Script.....	3-32
Analyzing the Results .....	3-33
Exercise Summary .....	3-35
Exercise 5 — Using External TestData Files .....	3-36
Getting Started .....	3-36
Learning the Script.....	3-36
The Resulting Script.....	3-37
Understanding TestData Files .....	3-38
Modifying the Script .....	3-40
The Modified Script .....	3-41
Running the Script.....	3-42
Exercise Summary .....	3-42
Exercise 6 — Creating Clock Checks for External Data Entry .....	3-42
Getting Started .....	3-42
Learning the Script.....	3-43
Modifying the Script .....	3-44
Defining an Event .....	3-45
Additional Script Modifications.....	3-45
The Resulting Script.....	3-46
Running the Script.....	3-47
Analyzing the Results .....	3-47
Exercise Summary .....	3-48
Exercise 7 — Creating Text Checks .....	3-48
Getting Started .....	3-48
Learning the Script.....	3-49
Exercise Summary .....	3-56
<b>Chapter 4. Using Driver and Test Scripts Together .....</b>	<b>4-1</b>
Exercise 8 — Using the Run Command .....	4-2
Getting Started .....	4-2
Modifying the Driver Script.....	4-3

The Resulting Script.....	4-4
Running the Complete Script.....	4-5
Analyzing the Results .....	4-6
Viewing Failed Checks .....	4-6
Failed Menu Checks.....	4-7
Failed Text Checks.....	4-10
Updating Checks .....	4-11
Exercise Summary .....	4-11
Exercise 9 — Inserting Script Dialog Boxes.....	4-11
Getting Started .....	4-12
Creating the Dialog .....	4-12
Adding Controls .....	4-14
The Resulting Script.....	4-20
Modifying the Driver Script.....	4-20
Running the Script.....	4-22
Exercise Summary .....	4-22
<b>Index .....</b>	<b>I-1</b>



# Introduction

This manual focuses on testing graphic user interface (GUI) applications using the demonstration program QADemo as the target application. This guide describes basic testing premises using *QARun* — how it works, what it can do, and what you can use it for — through a comprehensive tutorial designed to introduce you to the concepts involved with building complete test suites.

The exercises presented in this manual are designed to take you from the most basic testing principles through the more advanced testing concepts using *QARun*. This manual is divided into the following chapters:

- **“Chapter 1. Introducing QADemo”**: This chapter establishes a testing scenario and introduces the target application used in the tutorial — QADemo.
- **“Chapter 2. Building the Driver Script”**: This chapter provides an exercise that assists you with creating a driver script that will be used in later exercises. The purpose of this chapter is to help you create the driver script and to explain some of *QARun*’s basic functionality.
- **“Chapter 3. Building Test Scripts”**: This chapter provides individual exercises that introduce various aspects of testing using *QARun*. Each exercise creates a test script that will be used to test specific elements of the target application. Each exercise introduces *QARun* features that can be used to effectively test the target application.
- **“Chapter 4. Using Driver and Test Scripts Together”**: This chapter provides exercises that discuss integrating driver scripts and individual test scripts in order to create a complete test suite. An additional exercise is presented that introduces *QARun*’s Dialog Editor.

The concepts required to test character-based applications are covered in the *QARun Character-Based Testing Getting Started Guide*. You should refer to that document for introductory information related to testing mainframe applications.

---

## Who Should Read This Guide

The *QARun GUI Testing Getting Started Guide* is intended for new *QARun* users who need to learn how to implement testing concepts using *QARun* as their testing tool.

We designed and organized the information in this guide to help novice users gain a basic understanding of the test-building process using *QARun*. This guide does not contain extensive reference or theoretical information. You can find that information in the online help facility, in the *QARun User's Guide*, and in the *QARun Language Reference Manual*.

We assume that you have some familiarity with basic Microsoft Windows navigation. If this isn't the case, familiarize yourself with the documentation accompanying your copy of Microsoft Windows before reading this guide.

---

## Related Publications

In addition to the *QARun GUI Testing Getting Started Guide*, the *QARun* documentation set includes the following other publications as well as a complete online help facility.

- *QACenter Installation and Configuration Guide* provides licensing instructions, step-by-step installation procedures, and database selection information.
- *QARun Character-Based Testing Getting Started Guide* provides a “hands on” tutorial designed to introduce you to the concepts required to test mainframe character-based applications. The tutorial begins by creating a driver script, then creates an individual test script, and then integrates the two to form a complete test system.
- *QARun User's Guide* provides a complete reference to using *QARun*. It provides a basic product and component overview and explains how to configure the system, create scripts, define test conditions (checks), and view the results of a test run.
- *QARun Language Reference Manual* provides a reference to the commands available for use in your *QARun* scripts. It contains detailed information specific to command syntax, variants, operation, and script examples. It is intended for experienced *QARun* users who wish to utilize the scripting language to develop robust, sophisticated test procedures.
- *QARun Online Help* provides help information. If you press the **F1** key while using the software, you can obtain dialog-sensitive online help. If you encounter any problems that cannot be resolved, contact *QARun* Technical Support.
- The online Bookshelf provides access to the complete *QARun* documentation set in Adobe Acrobat PDF format. The Bookshelf is installed automatically with the *QARun* program files. You can access the Bookshelf by selecting **Bookshelf** from the *QARun* program group. The book will open automatically in Acrobat Reader.

If you have any comments or suggestions regarding *QARun*, please fill out the Reader's Response Form located in the back of this guide.

To access more information about Compuware Corporation and its enterprise-wide systems software tools, refer to the sections below.

---

## World Wide Web Information

To access Compuware Corporation's site on the Internet World Wide Web, point your browser at <http://www.compuware.com>. The Compuware site provides a variety of product and support information.

### FrontLine Support Web Site

You can access online technical support for Compuware products via our FrontLine support web site. FrontLine provides you with fast access to critical information about your *QACenter* product. You can read or download documentation, frequently asked questions, and product fixes, or directly e-mail Compuware with questions or comments.

In order to access FrontLine, you must first register and obtain a password. To register, point your browser at <http://frontline.compuware.com>. FrontLine is currently available for customers in the United States and Canada. FrontLine services for other countries will be available in the future.

---

## QARun Terminology

The following terms are used throughout the *QARun GUI Testing Getting Started Guide*.

**Target Application:** The system under test.

**Scripts:** Step-by-step instructions that specify how the target system should be tested.

**Test Site:** A point in the target application where testing begins.

**Driver Script:** A *QARun* script that drives the target application to a test site. A driver performs no testing.

**Test Script:** A *QARun* script that is called by the driver script when a test site is reached. A test script performs some actions and then executes a "check" that compares the target's response to the actions with the expected response.

**Check:** Definition of the correct response of the target application to the given input. A check may be called by single or multiple test scripts.

## Getting Help

At Compuware, we strive to make our products and documentation the best in the industry. Feedback from our customers helps us maintain our quality standards.

If you need support services, please obtain the following information before calling Compuware's 24-hour product support hotline:

- The version of *QARun* you are using, which is displayed when you select the About command from the Help menu or on the coversheet of this document.
- The version of Microsoft Windows you are using, for example: Windows NT service pack 1, 2, or 3; Windows 95 OEM2 or OEM1. This information is obtainable from the system icon in the control panel.
- The location of the problem in the *QARun* software, and the actions taken before the problem occurred.
- The exact *QARun* message, if any.
- The exact workstation, system, or Microsoft Windows error message, if any.



***QARun* Technical Support**  
Compuware Corporation  
31440 Northwestern Highway  
Farmington Hills, MI 48334-2564  
1-800-538-7822

## Chapter 1. Introducing QADemo

The subsequent sections of this getting started guide describe how to use *QARun* to perform practical exercises that are common testing requirements. To complete these exercises, we must first create a testing scenario that includes a fictitious company profile, an application-under-test (the target application), and the driver and test scripts.

The *QARun* installation process loads a demonstration application that we will use as our target application — QADemo. This chapter describes the testing scenario: the company you are working for, its testing requirements, and the target application. Several short exercises are provided to familiarize you with the functionality of QADemo. These exercises include:

- Logging on to the system
- Using the Main Options dialog box
- Adding a car
- Making an inquiry
- Booking a car
- Transferring a car
- Selling a car.

After you gain an understanding of QADemo, Chapter 5, “Tutorial” provides several exercises aimed at building test suites to ensure that the target application meets the company’s requirements and works as specified. It provides a hands-on guide for building test suites to use against the target application.

## The Company Profile

The company software you are testing is for an international car distribution center with outlets in New York, London, and Paris.

The company requires an application to keep track of stock in all three outlet locations, to allow the stock to be transferred, to allow cars to be sold (only if they are in stock), to generate invoices, to keep a record of customer names and addresses, to book stock in, to add new cars, to calculate distributor discounts, and to delete obsolete lines.

The QADemo software will be available to you, for testing purposes, in three phases:

- **Phase One, Version 1** — This version is the basic system containing all the necessary features, but it is not yet functional. This version of the software will allow you to define what should be tested and how to test it.
- **Phase Two, Version 2** — This version of the software has some new modifications to the interface, and it is now functional. This version of the software will allow you to test the interface and enhance your existing scripts in order to test the functionality and response of the system.
- **Phase Three, Version 3** — This version of the software is the complete working version of the system. The tests that you built using the previous versions of the software can now be run against the final version as an acceptance test.

---

## The Target Application — QADemo

The target application is a database that is designed as a stock control and sales ledger system. The program allows cars to be sold and transferred between locations. The user may select which location they wish to work from during the logon sequence. This allows the user to transfer cars between locations and to add or delete cars at any site.

## Logging On and Selecting Main Options

Use the following procedure to start a copy of QADemo and to display the Main Options dialog box:

1. Start QADemo:
  - Windows 95 and NT 4 users:

Although *QARun* can execute an application from anywhere in the system, the safest way to launch the application is to use the **Start** button on the taskbar.

- Click the **Start** button and choose **Run** from the menu. In the Run dialog box, enter the application's directory location. The default directory is:

```
"C:\Program Files\Compuware\QARun\Demos\qademo.exe" /v3
```

- Click **OK**.

- Windows NT 3.51 users:

Although *QARun* can execute an application from anywhere in the system, the safest ways to launch the application is to choose **Run** from Program Manager's File menu.

- From the **File** menu, choose **Run**. Enter the application's directory location in the Run dialog box. The default directory is:

```
C:\Program Files\Compuware\QARun\Demos\qademo.exe /v3
```

- Click **OK**.

2. After a short wait, the Enter Sign on Details dialog box displays. The dialog asks for a user ID, password, and the site location.

The screenshot shows a dialog box titled "Enter Sign on Details". It has a standard Windows-style title bar with a close button (X). The dialog contains three input fields: "User Id:" with a text box, "Password:" with a text box, and "Location" with three radio buttons labeled "London", "New York", and "Paris". The "Paris" radio button is selected. To the right of the input fields are two buttons: "OK" and "Cancel".

3. Enter the following sign on information:

- Enter **CW** in the User ID field
- Enter **PASS** in the Password field.
- Select the London radio button as your site location.

After successfully signing on, the New dialog box displays. This dialog box displays a list of QADemo's main options:

The screenshot shows a dialog box titled "New". It has a standard Windows-style title bar with a close button (X). The dialog contains a list box with four items: "Add Car", "Inquire", "Book Car In", and "Transfer Car". The "Add Car" item is selected and highlighted. To the right of the list box are two buttons: "OK" and "Cancel".

- Each main option is accessed by highlighting it and clicking **OK**.
- Every time you close an option's window (the Add Car window, for example), you are returned to QADemo's main window.

The New dialog box is not automatically redisplayed. From the **File** menu, choose **Main Options** to recall this dialog box.

## QADemo's Main Options

After logging on to QADemo, the New dialog box displays. The New dialog box presents a list of QADemo's main options. The remaining sections of this chapter are designed to introduce you to and familiarize you with accessing and using QADemo's main options.

### Add Car

Cars can only be added to the database using QADemo Version 3. Choosing the **Add Car** option displays the following dialog box:

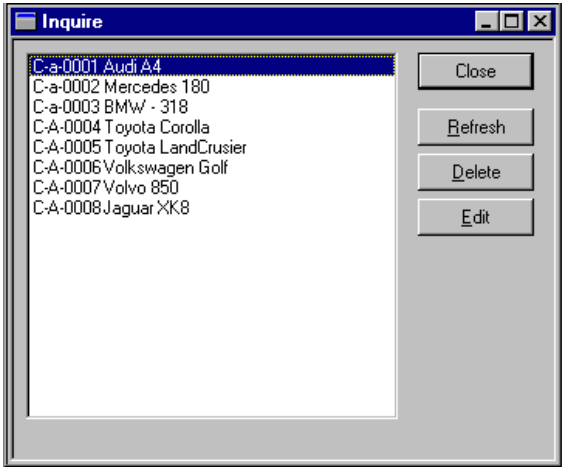
The 'Add Car' dialog box contains the following fields and controls:

- Ref: C-A-0013
- Make: [Empty]
- Buttons: Add, Close
- Engine size: [Empty]
- Year: [Empty]
- Doors: [Empty]
- Color: [Empty]
- Sale Price: 0
- Deal Discount: 0 %
- Quantity: 0
- Dealer Price: 0
- Extras section with checkboxes:
  - Radio
  - Leather
  - 4WD
  - Air Cond.
  - Sun Roof
  - Metallic

### Inquire

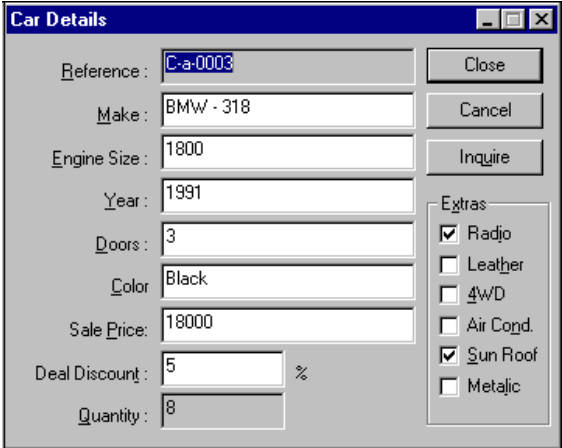
The Inquire option can only be performed using QADemo Version 3.

1. Choose the **Inquire** option to display the Inquire dialog box:



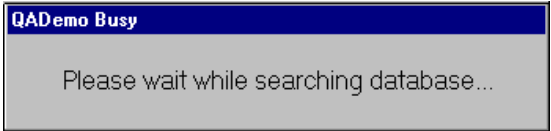
This dialog box contains a list of available cars, including any added cars.

- 2. Highlight the required car and click the **Edit** button. The Car Details dialog box displays:

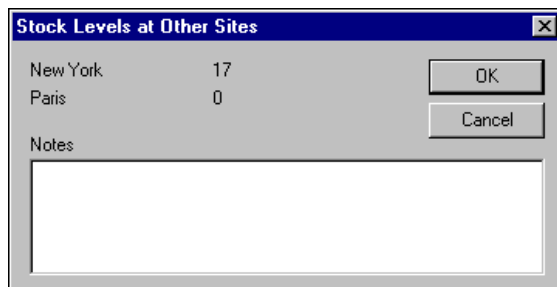


This dialog box displays the car details and stock levels at the current site.

- 3. Click the **Inquire** button. The following message box displays.



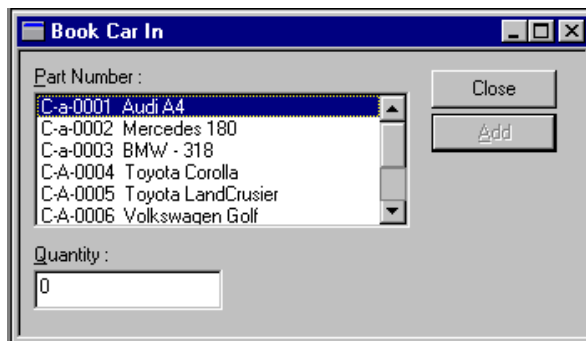
After a short wait, this window disappears. No user action is required. The following dialog box then displays:



## Book Car In

The Book Car In option can only be performed using QADemo Versions 2 and 3. This action is different from the Add Car option because it only allows you to add to the currently available levels of stock for existing cars. You may not add a new model to the stock.

1. Choose the **Book Car In** option to display the following dialog box:



2. Enter a value in the **Quantity** field to activate the **Add** button.
3. Click the **Add** button to increment the total stock level by the number displayed in the Quantity field.

## Transfer Car

The Transfer Car option can only be performed using QADemo Versions 2 and 3. The Transfer Car option allows you to transfer cars to a specific site. You cannot transfer a car to the site you are currently logged on to.

1. Choose the **Transfer Car** option to display the following dialog box:



2. Enter a value greater than zero in the **Quantity** field and click the **Transfer** button. This transfers the car(s) to the site indicated with the **Transfer Stock To** option.

A message box displays, indicating if there is sufficient available stock to successfully enact the transfer, or if there is insufficient stock to complete the transfer request.

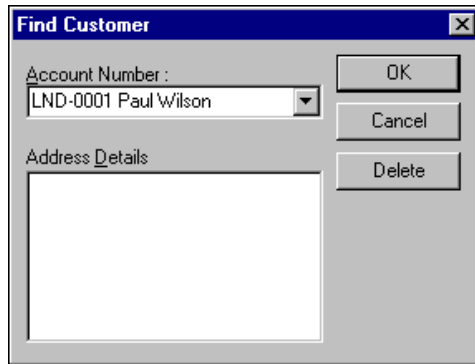
## Customer Invoice

The last option on the New dialog box is Customer Invoice. Using the Customer Invoice option is only possible using QADemo Version 3. This option allows you to sell one or more cars to an existing or new customer and generates an invoice.

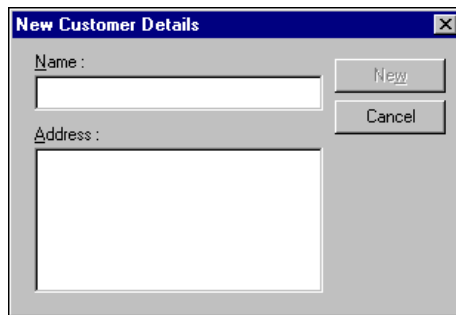
1. Choose the **Customer Invoice** option. The Customer Invoice dialog box displays:



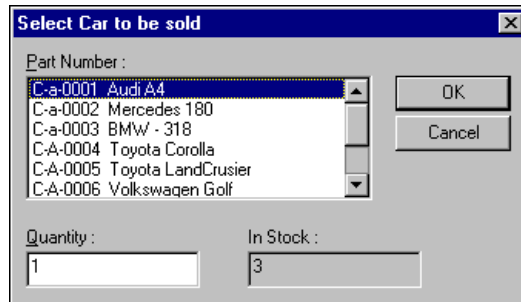
2. Click the **Find** Button to locate an existing customer. The Find Customer dialog box displays:



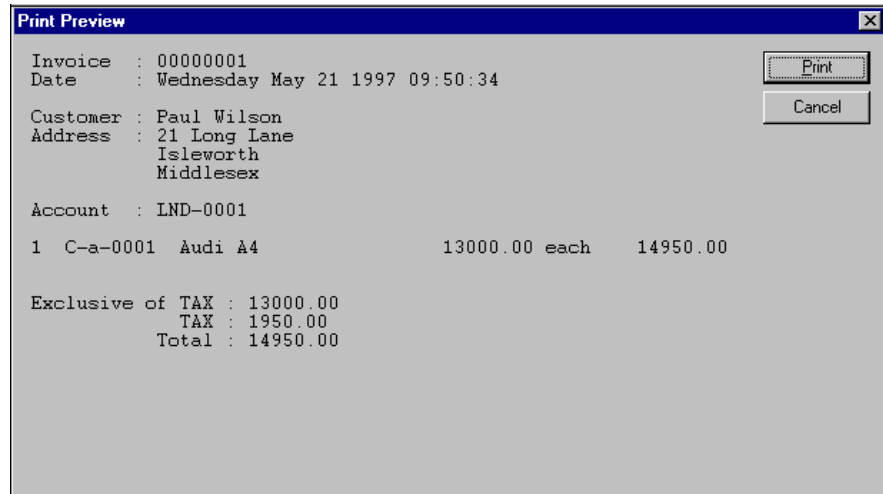
3. Select the **Account Number** drop-down list to display more customer list selections. Highlight a customer and click **OK** to add the customer details to the invoice.
4. Click the **New Customer** button from the Customer Invoice dialog box to display the following dialog box:



5. Enter the new customer details in the **Name** and **Address** entry fields. Then, click the **New** button to update the existing customer database and add the new customer details.
6. Once new customer details are entered into the invoice, the **Sell Car** button on the Customer Invoice dialog box is enabled. Click the **Sell Car** button to display the following dialog box:



7. Highlight a car from the list. The quantity in stock is displayed in the lower-right corner of the dialog box. Enter the quantity to sell in the **Quantity** field. If the quantity to sell is less than the amount in stock, the successful message box displays. Attempting to sell more cars than are in stock causes the error message box to display.
8. Once the car is successfully sold, the **Print** button becomes enabled. Click the **Print** button to display the Print Preview dialog box:



9. QADemo now provides the option to print the invoice or cancel the request (QADemo is simulation software. No printing actually occurs). Click the **Print** button to display a confirmation box.

The Print Preview display is automatically closed, and you are returned to the Customer Invoice dialog box.

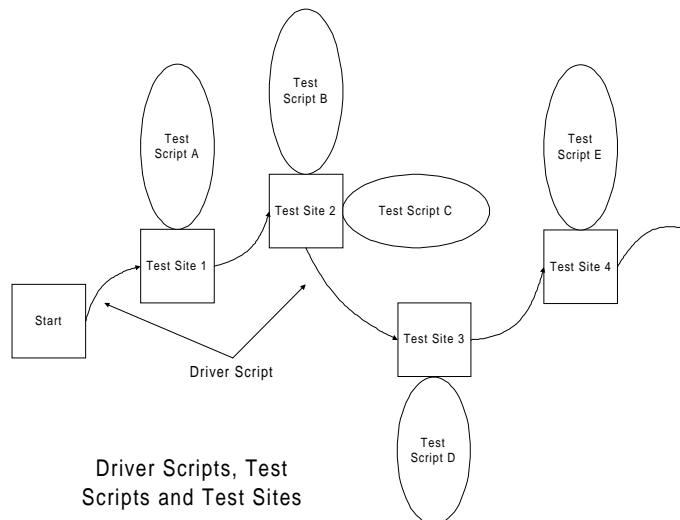
This completes your introduction to the sample target application, QADemo. You should now be familiar enough with the application to begin testing some of its elements using *QARun*. After logging on to QADemo, you are presented with the New dialog box. The New dialog box displays a list of QADemo's main options. The remaining chapters of this guide are designed to introduce you to and familiarize you with accessing and using QADemo's main options.



## Chapter 2. Building the Driver Script

The purpose of the following exercise is to build a driver script to run a copy of the target application, QADemo.

A complete test suite contains many individual tests, executed at different points (test sites) within the target application. The tests are invoked by the controlling script, called a *driver script*. Driver scripts are built the same way as test scripts are, but the driver does not actually test the target application. It merely “drives” the target system to the proper point to begin a test (called a *test site*) and then calls one or more test scripts to begin testing at that test site. When the test finishes, the driver regains control and proceeds to the next test site. Figure 2-1 demonstrates how a driver script interacts with test scripts.



**Figure 2-1.** Driver Script and Test Script Integration

A well-designed test script returns the target application to the original test site before completion. This ensures that the driver scripts can always pick up from where they left off, and that you can add or remove tests at a test site without modifying the driver path.

---

## Exercise 1 — Creating a Driver Script

When complete, the driver script created in this exercise will select each main option, one at a time, from the QADemo New dialog box. After arriving at each option, the driver script will close the resulting dialog box. The driver script will not perform any processing.

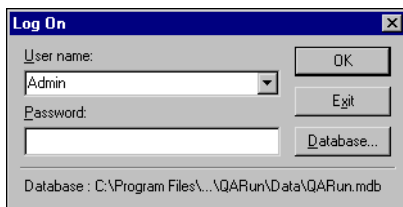
### Starting QARun

Use the following procedure to launch QARun and to log on to the QARun database:



1. Start QARun:
  - If you are starting QARun under Windows 95 or Windows NT 4:
    - Click the taskbar **Start** button and select **Programs\Compuware\QARun** from the Start menu.
    - Select the QARun icon.
  - If you are starting QARun under Windows NT 3.51:
    - Double-click the QARun icon in the QARun program group.

The QARun splash screen displays followed by the QARun Log On dialog box:



2. If your userID has been added to the User Table, select your User name from the drop-down list and enter your password in the Password entry field.

If you are using the software for the first time, you may log on as Admin. The default password for this user is “Admin”. You should change this password once you have logged on to prevent unauthorized access.

3. Click **OK** to log on.

## Changing Configuration Options

Before you begin creating the driver script, you need to change one of *QARun*'s configuration options. Changing this option will make your resulting driver script more closely resemble the examples in the following exercises and allow you to test other versions of *QADemo*. Specifically, you'll need to change the following:

- **Script Editor** — You will change the default script's text and change the way *QARun* learns pauses between your actions. You will also tell *QARun* to Learn image selections as *BitmapSelects*.

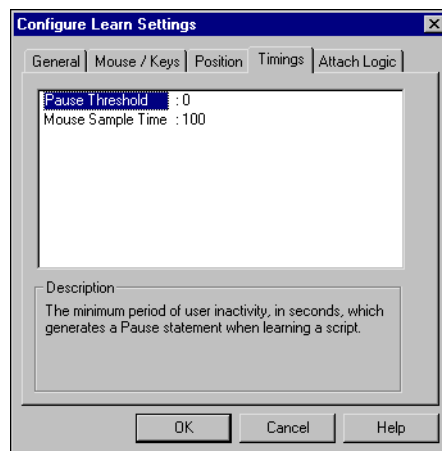
You must change the specific configuration options described below in order to generate scripts that closely resemble the examples provided throughout this guide; however, if you wish, you may also change some of the Script Editor's general configuration options to suit your personal preferences. For example, you may change the fonts and colors used in the Script Editor, and you may also change the way the Script Editor behaves when it is active. If you wish to alter any of these general configuration options, refer to the section on configuring the Script Editor in the *QARun User's Guide*.

### Changing Learn Settings

Use the following procedure to change the Script Editor's Learn Settings options:



1. Click the **Editor** button on the Modules toolbar to open the Script Editor.
2. From the Script Editor's **Options** menu, choose **Learn Settings**. The Configure Learn Settings dialog box displays.
3. Click the **Timings** tab and double-click the **Pause Threshold** option.



4. If necessary, change the **New Value** field to **0** and click **OK**.

The Pause Threshold value is the amount of time that *QARun* will wait before learning pauses between learned actions in the script. Setting the value to 0 tells *QARun* not to Learn your pauses and replays scripts as fast as possible.

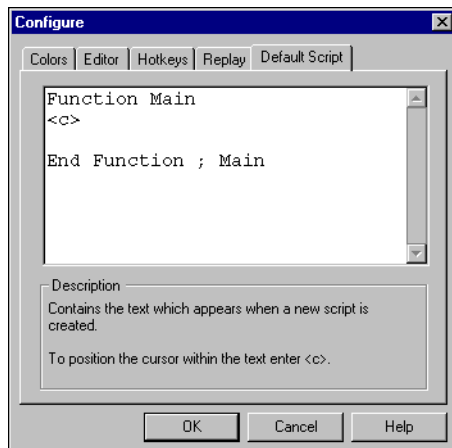
5. Click the **General** tab and select the **BitmapSelects** check box.
6. Ensure that the **TypeToControl** check box is not selected.

## Changing Default Script Information

Use the following procedure to change the Script Editor's default script information:



1. Click the **Editor** button on the Modules toolbar to open the Script Editor.
2. From the Script Editor's **Options** menu, choose **Configure**. The Configure dialog box displays.
3. Click the **Default Script** tab:



*QARun*'s default script allows you to specify information that is automatically placed into each subsequent script that is created. By default, *QARun* provides error handling text in the default script, but you can change the default script to anything.

4. Error handling will not be used for the exercises that follow. Remove all information from the default script except the following code:
 

```
Function Main
<c>
End Function ; Main
```
5. Click **OK**.

## Getting Started

Now that you have changed *QARun*'s configuration options, you are ready to begin the exercise. This exercise is designed to create a simple driver script.

**Exercise Prerequisites:** Before beginning this exercise, you must have completed the following prerequisites:

- Properly installed *QARun*.
- Changed *QARun*'s configuration options (see page 2-3).
- Closed any open *QARun* scripts.

**Testing Requirements:** The following test elements are created during this exercise:

- Script named *Driver1*.

You should allow 10–15 minutes to complete this exercise.

## Learning the Script

Use the following procedure to create and Learn a new script for this exercise:

1. Start *QARun* and log on to the system (see page 2-2).
2. From the **File** menu, choose **New**. The New dialog box displays.
3. Select **Script** from the Create New list and click **OK**.
4. Click the **Learn** button on the Script Editor's toolbar. *QARun* minimizes to allow clear access to the desktop.
5. Start QADemo Version 1:
  - Windows 95 and NT 4 users:



Although *QARun* can execute an application from anywhere in the system, the safest way to launch the application is to use the **Start** button on the taskbar.

— Click the **Start** button and choose **Run** from the menu. In the Run dialog box, enter the application's directory location. The default directory is:

```
"C:\Program Files\Compuware\QARun\Demos\qademo.exe" /v1
```

— Click **OK**.

- Windows NT 3.51 users:

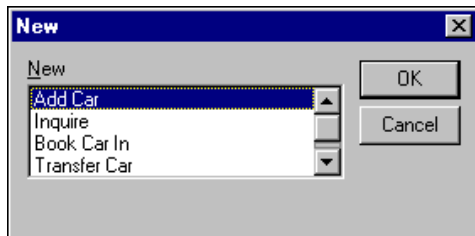
Although *QARun* can execute an application from anywhere in the system, the safest way to launch the application is to choose **Run** from Program Manager's **File** menu.

- From the **File** menu, choose **Run**. Enter the application's directory location in the Run dialog box. The default directory is:

```
C:\Program Files\Compuware\QARun\Demos\qademo.exe /v1
```

- Click **OK**.

6. After a short wait the QADemo main window appears along with the Enter Sign on Details dialog box.
7. Complete the Enter Sign on Details dialog box with the following information:
  - a. Enter **CW** in the User ID field.
  - b. Enter **PASS** in the Password field.
  - c. Select the London radio button as your site location.
8. Click **OK**. The New dialog box displays a list of QADemo's main options:



9. Select **Add Car** from the list and click **OK**. The Add Car dialog box displays.
10. Click the **Close** button to return to QADemo's Add Car dialog box.

Remember, no processing is carried out at this point; you're simply creating a driver script that selects each main option and then returns to the New dialog box. You're not actually testing the application at this time — you will create individual test scripts in future exercises to conduct the actual testing.

11. From the **File** menu, choose **Main Options**. The New dialog box re-displays.
12. Repeat the steps 9–11 to select each option in the New dialog box list. Remember to click the **Close** button after selecting each option.

## Stop Learning the Script

After selecting each option from the New dialog box, use the following procedure to stop learning the driver script.

1. From the File menu, choose **Exit**. QADemo closes.
2. To stop learning, press the **Learn Hotkey**, {Alt {F10}}.

Hotkeys allow you to access *QARun*'s major functions without recording the hotkey keystrokes while Learn is active. The Hotkey is not learned in the script, nor is it seen by any other application.

If the key combination assigned to the hotkey is already assigned to your target application, you can change the hotkey assignment by choosing **Configure** from the Script Editor's Options menu.

## The Resulting Script

After you turn Learn off, *QARun* reappears and displays the contents of your new script. The script should resemble the following example:

```
Function Main
Attach "PopupWindow~1"
    Button "Start", 'Left SingleClick'
    PopupMenuSelect "Run..."
Attach "Run PopupWindow"
    ComboText "&Open:",
    "C:\Program Files\Compuware\QARun\Demos\QADemo.exe" /v1
    Button "OK", 'Left SingleClick'
Attach "Enter Sign on Details PopupWindow"
    EditText "&User Id:", "CW"
    EditText "Pa&ssword:", "Pass"
    RadioButton "London", 'Left SingleClick'
    Button "OK", 'Left SingleClick'
Attach "New PopupWindow"
    ListBox "&New ", "Add Car", 'Left SingleClick'
    Button "OK", 'Left SingleClick'
Attach "Add Car ChildWindow"
    Button "Close", 'Left SingleClick'
Attach "QADemo MainWindow"
    MenuSelect "File~Main Options"
Attach "New PopupWindow"
    ListBox "&New ", "Inquire", 'Left SingleClick'
    Button "OK", 'Left SingleClick'
Attach "Inquire ChildWindow"
    Button "Close", 'Left SingleClick'
```

```
Attach "QADemo MainWindow"
    MenuSelect "File~Main Options"
Attach "New PopupWindow"
    ListBox "&New ", "Book Car In", 'Left SingleClick'
    Button "OK", 'Left SingleClick'
Attach "Book Car In ChildWindow"
    Button "Close", 'Left SingleClick'
Attach "QADemo MainWindow"
    MenuSelect "File~Main Options"
Attach "New PopupWindow"
    ListBox "&New ", "Transfer Car", 'Left SingleClick'
    Button "OK", 'Left SingleClick'
Attach "Transfer Car ChildWindow"
    Button "Close", 'Left SingleClick'
Attach "QADemo MainWindow"
    MenuSelect "File~Main Options"
Attach "New PopupWindow"
    ListBox "&New ", "Customer Invoice", 'Left SingleClick'
    Button "OK", 'Left SingleClick'
Attach "Customer Invoice ChildWindow"
    Button "Close", 'Left SingleClick'
Attach "QADemo MainWindow"
    MenuSelect "File~Exit"
End Function
```



**Note**

---

Your script results may vary slightly from the above example if your QARun system configuration is different than the one used to Learn the example script or if you learned extra mouse clicks.

---

## Understanding the Script

The script you just learned is comprised of five basic sections that merit explanation.

- Attaching to the dialog box.
- Signing on to the application.
- Selecting options from the list.
- Closing the dialog box.
- Closing the application.

This section explains how each action was converted into QARun's script language. The following is a descriptive breakdown of the script.

**Attaching to the Dialog Box:** Attach is probably the most important command in the *QARun* scripting language. By attaching to a particular window, *QARun* ensures that its actions are directed to the correct recipient. There may be many windows on the screen when a script is replayed — perhaps even windows that weren't there when the script was originally recorded. The Attach statement ensures that *QARun* finds the appropriate window and makes it the active window for the commands that follow. Most *QARun* scripts will have several Attach statements.

Because object mapping is turned on, your script's attach statements should resemble the following example:

```
Attach "Name of Window to attach to"
```

**Entering the Sign On Instructions:** In this portion of the script, the script attaches to the Enter Sign on Details dialog box, enters information into the User ID and Password fields, selects the radio button for the London location, and then clicks the **OK** button:

```
Attach "Enter Sign on Details PopupWindow"
  EditText "&User Id:", "CW"
  EditText "Pa&ssword:", "Pass"
  RadioButton "London", 'Left SingleClick'
  Button "OK", 'Left SingleClick'
```

**Selecting an Item from the List:** Each item on the New dialog box's list is selected, in order, from the list box. Each selection should resemble the following example:

```
Attach "New PopupWindow"
  ListBox "&New", "Add Car", 'Left SingleClick'
  Button "OK", 'Left SingleClick'
```

**Closing the Dialog Box:** Each option's dialog box is closed by clicking the **Close** button. Each option should resemble the following example:

```
Attach "Add Car ChildWindow"
  Button "Close", 'Left SingleClick'
```

**Closing the Application:** In this section of the script, the script attaches to the main window and selects **Exit** from the File menu:

```
Attach "QADemo MainWindow"
  MenuSelect "File~Exit"
```

## Running the Script

Before attempting to run the script, ensure that all active versions of QADemo are closed. Use the following procedure to run the driver script:

1. From the **Run** menu, choose **Run Script** or click the **Run** button from the toolbar.
2. The Summary Info dialog box displays the first time the script runs. Enter a file name for the script, such as *Driver1*. Conventional Windows long file names are supported.

A description of what the script does and any useful comments may be added at this point. Click **OK**.

3. The Run Script environment dialog box appears. This dialog box allows you to determine how the script is run, if the script generates a log file, and what type of information gets logged. It also allows you to determine how the script replays and how long the script should wait for windows and controls to appear before timing out.
4. QADemo should run, opening each main option from the New dialog box in the order that it appears in the driver script.

## Exercise Summary

After completing “Exercise 1 — Creating a Driver Script”, you should be able to:

- Understand the purpose of a driver script.
- Start the QARun application.
- Turn the Learn facility on and off while recording a script.
- Understand how basic actions are translated into the QARun’s script language.
- Run a script.

The exercises in the following sections focus on building test scripts that will be called by the driver script at the appropriate places.

## Chapter 3. Building Test Scripts

You will use the following exercises to build several test scripts. These scripts will eventually be called by the driver script. Together, the driver script and the test scripts will make a complete test suite. Test scripts are created the same way as driver scripts; however, you should create the driver and test scripts separately. Maintaining these scripts separately makes it much easier to modify your test system when the target application changes.

The following exercises demonstrate the main check types — window, form (dialog), menu, list and response checks, and text and bitmap checks. Additionally, one exercise includes a stress test using data read in from an external data file. This chapter contains the following exercises to help you build and understand test scripts:

- “Exercise 2 — Creating List, Window, and Menu Checks” creates a test script that is designed to undertake a series of list, window, and menu checks at a specific test site.
- “Exercise 3 — Creating Form Checks” focuses on creating form checks on the target application’s dialog boxes.
- “Exercise 4 — Creating Clock Checks To Test Performance” builds a performance test that makes an inquiry and then checks the target application’s response time. The exercise then uses the Repeat command to loop the script to generate a true performance test.
- “Exercise 5 — Using External TestData Files” creates a script that enters variable data into the target application — to test behavior with differing input or to test behavior under heavy-load conditions. This exercise creates a test script that carries out a volume test by reading a data file and entering the information into QADemo.
- “Exercise 6 — Creating Clock Checks for External Data Entry” builds a test script that reads records from an external file and adds the records to the database.
- “Exercise 7 — Creating Text Checks” creates a test script that uses a text check to highlight areas of the screen (typically fields of data) and analyzes the values displayed to check dates and numeric values and ensure that fields contain alpha or alphanumeric strings.

---

## Exercise 2 — Creating List, Window, and Menu Checks

The following exercise creates a test script that is designed to eventually take control from the driver script after the driver logs onto the system and before the Add Car option is selected. At this point in the target application, the test script will undertake a series of list, window, and menu checks before it restores the application to its original state and relinquishes control to the driver script.

### Getting Started

**Exercise Prerequisites:** Before beginning this exercise you must have completed the following prerequisites:

- Closed all copies of QADemo.
- Closed all copies of QARun.

**Testing Requirements:** The requirements for this test script are that it:

- Checks the contents of the New dialog box.
- Checks the Main Window at startup to ensure that the title bar states “QADemo”.
- Checks Main Window’s menus at startup.
- Selects **Add Car** from the New dialog box’s list and:
  - Checks the Main Window title for QADemo - Add Car.
  - Checks the Main Window menu options to verify that the View menu contains the Add Car window name and that it is checked.
- Selects **Main Options** from the File menu to recall the Main Options list.
- Selects **Inquire** from the list and:
  - Checks the Main Window to ensure that the title shows QADemo ~ Inquire.
  - Checks the Main Window menu options to verify that the View menu contains the Add Car and Inquire window names, with Inquire window name selected.
- Restores the system to its starting point by closing both windows and recalling the main options New dialog box.

The following test elements are created during this exercise:

- Script named *Exercise 2*.
- List check named *Main Options List*.
- Window check named *QADemo Main Window at Startup*.
- Window check named *The Add Car Window*.
- Window check named *Check the Inquire Window*.
- Menu check named *QADemos menus at startup*.
- Menu check named *Checking the View Menu Option*.
- Menu check named *Checking the View Menu Again*.

You should allow 10–15 minutes to complete this exercise.

## Learning the Script

Use the following procedure to run the appropriate version of QADemo and to start defining test cases against the target application:

1. Start *QARun* and open a new script.
2. Start QADemo Version 1 in the normal way (see page 2-5 if necessary).
3. Sign on to the system using the following information:
  - a. Enter **CW** in the User ID field.
  - b. Enter **PASS** in the Password field.
  - c. Select the London radio button to choose the site location.
4. Click **OK**.
5. Press the **Learn Hotkey**, {Alt {F10}}, to start learning the script.

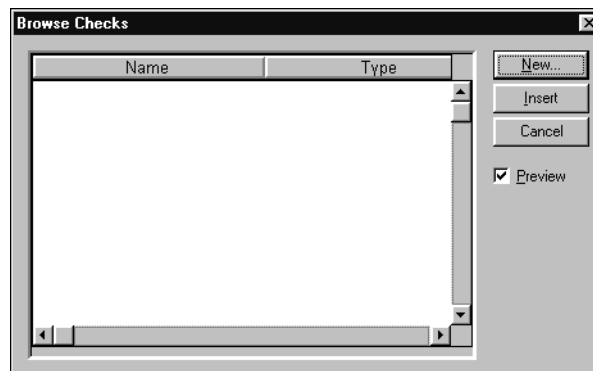
## Inserting Checks

The first goal of this test script is to check the contents of the New dialog box's main options list before carrying out any additional actions.

Remember, the driver script has already signed on to the system. This test script takes control after sign on, performs the tests, and returns control to the driver when the checks are complete. The first test script performs a check on the contents of the Main Options list control.

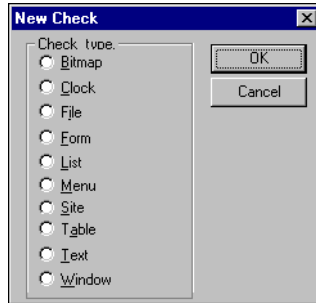
### Creating a List Check

1. Press the **Insert Check Hotkey**, {Alt {F8}}, to insert a check. The Browse Checks dialog box displays:

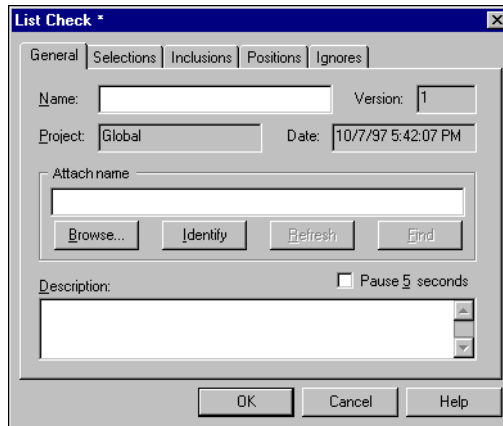


The list is probably empty because you have not defined any checks yet. The checks you create will appear in the list. After you create a check, you can highlight it and click the **Insert** button to insert it into a script.

2. Click the **New** button on the Browse Checks dialog box to define a new check. The following dialog box displays:



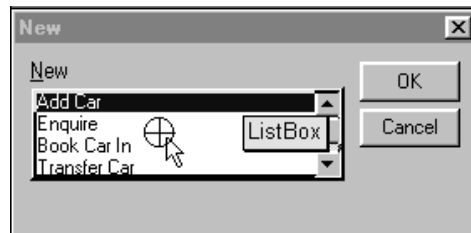
3. Select the List radio button to create a list check and click **OK**. The List Check dialog box displays:



- The Name field is mandatory, and each check must be given a unique name. The check name can be up to 50 characters and may include spaces.
  - The Description field is optional. However, a combination of descriptive names and detailed descriptions makes it easier to identify the check's purpose both in the script and in the check map.
4. Enter a name, such as *Main Options List* into the Name field and click the **Identify** button.

QARun minimizes. If the cursor is not over the list, it changes to the “no drop” pointer (a circle with a line through it). You cannot select the control when the cursor is in this state.

5. Position the pointer over the list of options. The cursor changes to the “target pointer,” and the list is highlighted as shown below:



The control type is displayed in a text panel. It is now possible to select the ListBox by single-clicking.

QARun restores, and the List Check dialog box is completed with the appropriate control information.

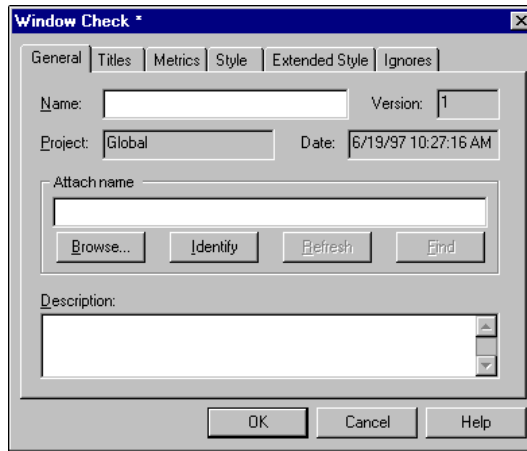
6. Click the List Check dialog box’s other tabs to view the available check options:
  - Selections tab:** Displays the currently selected items in the list. To create multiple selected items, click once on each of the required items in the list.
  - Inclusions tab:** Lists all items included in the check. To ignore an item during a check, click once in the selection box. Ignored items are marked with an “x”.
  - Positions tab:** Shows the position of the items in the list. Items may be inserted or removed, the text may be edited, and the list items can be moved up or down to modify the check to the expected results.
  - Ignores tab:** Shows which aspects of the list are included in the check. The default is to check every aspect, and each option is marked with an “x”.
7. Select **OK** to save the check. The check syntax is automatically added to the script.

## Creating a Window Check

The next test requirement is to check the Main Window to ensure that the title bar states the application name “QADemo.” You will accomplish this using a window check. Window checks check the overall structure of the window. For example: Is it in focus? Does it have a menu bar? Does it have scroll bars? Is the title correct? This window check will be modified to ignore the window’s size and position.

Use the following procedure to create a window check:

1. Press the **Insert Check Hotkey, {Alt {F8}}**.
2. Click the **New** button on the Browse Checks dialog box. The New Check dialog box displays.
3. Select the Window radio button and click **OK**. The Window Check dialog box displays:



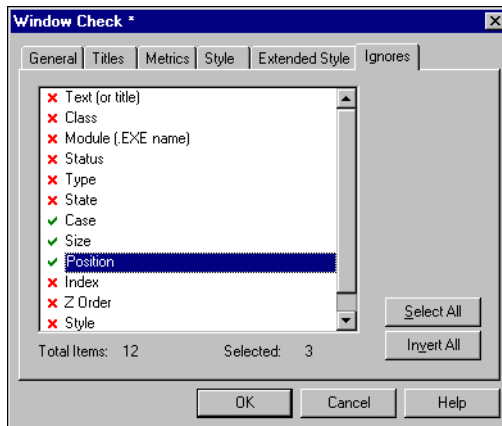
4. Enter a name such as *QADemo Main Window at Startup* and click the **Identify** button. QARun minimizes and the cursor changes to the target pointer.
5. Position the pointer over the QADemo window.

As the pointer is moved around the window, each child window is highlighted and the text panel indicates the control type.

6. Position the pointer on the QADemo title bar — so that the text panel says “MainWindow” — and click once.

QARun restores and the Window Check dialog box is completed with the appropriate window information.

7. Select the Ignores tab from the Window Check dialog box. The following information displays:



This tab defines the window elements and values that are included or ignored when the check is run.

- From the list, click the “x” for both the Size and Position options so that they appear with a checkmark. This indicates that these aspects of the window will be ignored when the check is performed.



#### Note

---

The Case option involves recognition of the attach name when the script is run under Windows NT or Windows 95. Under Windows NT, the attach names are displayed in uppercase.

---

- Select **OK** to save the check. The check syntax is automatically pasted into the script.

### Creating a Menu Check

The next test requirement is to check the menu items on the Main Window at startup. This is accomplished using a menu check. Menu checks check the status of each individual menu item: Is it enabled/disabled? Is it checked? Are there accelerator clashes, quick-key clashes, etc.?

Use the following procedure to create a menu check:

- Press the **Insert Check Hotkey**, {Alt {F8}}.
- Click the **New** button from the Browse checks dialog box. The New Check dialog box displays.
- Select the Menu option and click **OK**. The following dialog box displays:

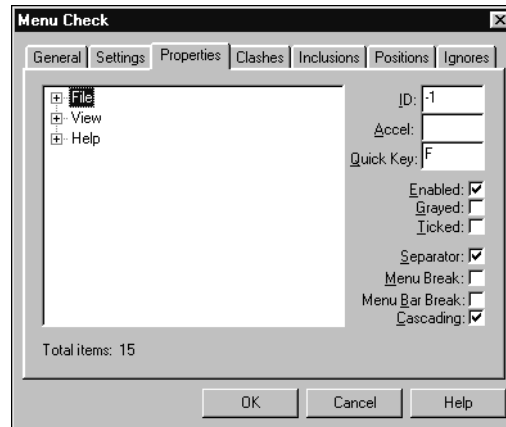
- Enter a check name, such as *QADemos menus at startup* and click the **Identify** button. *QARun* minimizes and the cursor changes to the target pointer.
- Position the pointer over the *QADemo* window.

As with window checks, when the pointer is moved around the window, each child window highlights and the text panel indicates the respective control type.

6. Position the pointer on the QADemo title bar so that the text panel displays “MainWindow” and click once.
7. Ensure that the cursor is pointing to the title bar and press **{Ctrl{Ins}}** as instructed by the text panel.

This captures the menu items from the identified window. QARun restores and the Menu Check dialog box is completed with the appropriate menu information.

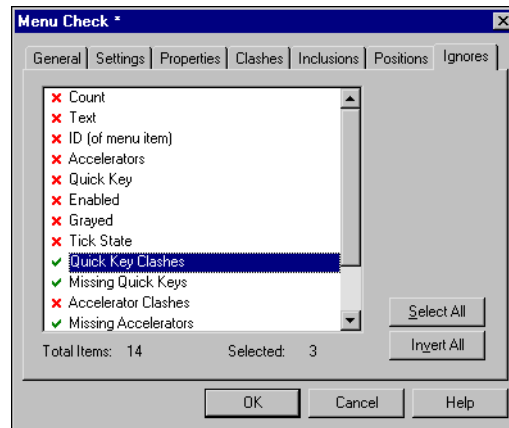
8. Select the Properties tab from the Menu Check dialog box. The following information displays:



**Hint**

If the display area is blank, the cursor was not positioned over the correct window. Select **General** and identify the window again, ensuring that the pointer is placed over the correct window before capturing the menus.

- Clicking on the “+” indicator beside each top-level menu option displays the drop-down menu items in tree format.
  - Highlighting a menu item displays its ID, accelerator, and quick keys, as well as its status. This information may be edited and changed to reflect expected results.
9. Select the Ignores tab from the Menu Check dialog box. The following information displays:



This tab defines the menu elements and values that are included or ignored when the check is run.

10. From the list, click the “x” for the Quick Key Clashes option so that it appears with a checkmark. This tells *QARun* to ignore any quick key clashes on the this menu when the check is run.

When quick key or accelerator key clashes are not ignored, and clashes exist, the menu check will fail, even if the clash exists in both the source and recorded check.

11. Click the Menu Check dialog box’s other tabs to view the available check options:

**Settings tab:** Determines if the menu items are checked according to the text displayed on the menu or by the menu item’s numeric ID. This is particularly useful for multi-language systems where the menu text is sure to change. The numeric ID is usually unique and does not change. This means that menu checks will work, regardless of language version.

**Clashes tab:** Displays any accelerator or quick-key clashes on each menu.

**Inclusions tab:** Determines which menu items are included or excluded from the check. The default is to include all menu items.

**Positions tab:** Furthers the check scope by inserting and deleting menu items and moving items up and down. This is used to reflect expected results not yet available on the current version of the software.

**Ignores tab:** Determines which part of the menu’s status is included or excluded from the check.

12. Select **OK** to save the check.

## Continuing the Script

So far, the test script will not carry out any tasks other than performing checks. In this section of the exercise, you will add functionality to drive the application and check the changes in its state before restoring the application to its previous state and relinquishing control to the driver script.

The next test requirement is to select the Add Car option and perform another window and menu check. Remember, *QARun* is still in Learn mode. Use the following procedure to perform another window and menu check:

1. Select the Add Car option in the list and click **OK**. Although the Add Car option is already highlighted, clicking on it ensures that *QARun* learns the selection. Therefore, if the option ever changes position, the script will still select it.
2. Create another window check for the Add Car dialog box by pressing the **Insert Check Hotkey**, **{Alt {F8}}**. Remember to ignore the window size and position (see “Creating a Window Check” on page 3-5 if necessary).

Assign the check a name, such as *The Add Car Window*.

3. Create another menu check on the Main Window’s menu *with* the Add Car dialog box displayed by pressing the **Insert Check Hotkey {Alt {F8}}** once the Add Car dialog box displays. Remember to ignore the quick key clashes (see “Creating a Menu Check” on page 3-7 if necessary).

Assign the check a name, such as *Checking the View Menu Option* and click **OK**. Do not close the Add Car dialog box.

4. From the **File** menu, choose **Main Options**. The New dialog box displays.
5. Select the Inquire option from the list and click **OK**.
6. Create a window check for the Inquire dialog box, remembering to ignore size and position (“Creating a Window Check” on page 3-5 if necessary). Assign the check a name, such as *Check the Inquire Window*.

Then, create a menu check, remembering to ignore the quick key clashes (see “Creating a Menu Check” on page 3-7 if necessary). Assign the check a name, such as *Checking the View Menu Again* and click **OK**.

7. Close the Inquire and the Add Car dialog boxes.
8. From the **File** menu, choose **Main Options**. QADemo is restored to its original state.
9. Press the **Learn Hotkey**, **{Alt {F10}}**, to turn Learn off.

## The Resulting Script

The script should resemble the following example. The check syntax appears in bold typeface.

```
Function Main
; A list check on the main options New list in QADemo version 1
Check "Main Options List"

; Window Check on QADemo Version 1
Check "QADemo Main Window at Startup"

; Checking Menus at startup on QADemo Version 1
Check "QADemos menus at startup"
Attach "New PopupWindow"
    ListBox "&New ", "Add Car", 'Left SingleClick'
    Button "OK", 'Left SingleClick'
; Checking that the title bar changes to Add Car
Check "The Add Car Window"

; Check that the Add Car window name is added to the View Menu
Check "Checking the View Menu Option"
Attach "QADemo - Add Car MainWindow"
    MenuSelect "File~Main Options"
Attach "New PopupWindow"
    ListBox "&New ", "Inquire", 'Left SingleClick'
    Button "OK", 'Left SingleClick'
; Check that the Title Bar changes to Inquire
Check "Check the Inquire Window"

; Check that the View Menu now contains the Inquire window name
Check "Checking the View Menu Again"
Attach "Inquire ChildWindow_0001"
    Button "Close", 'Left SingleClick'
Attach "Add Car ChildWindow"
    Button "Close", 'Left SingleClick'
Attach "QADemo MainWindow"
    MenuSelect "File~Main Options"
End Function
```

The check definition is stored in the database so the check definition can be used in multiple test scripts and can be modified at any time. Saving any changes to the modified check automatically updates all scripts that reference the check.

## Running the Script

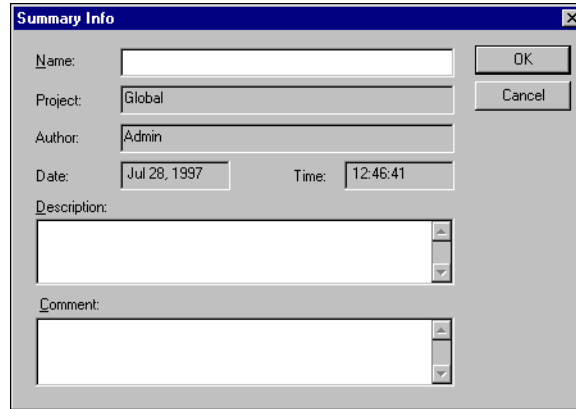
Although the driver script is intended to call the specified test scripts when it is complete, it is still possible to run the test scripts independent of the driver script (as long as the

target application is advanced to the proper test site). This is useful for verifying that the test-script components work properly.

Use the following procedure to run your test script without using the driver script:



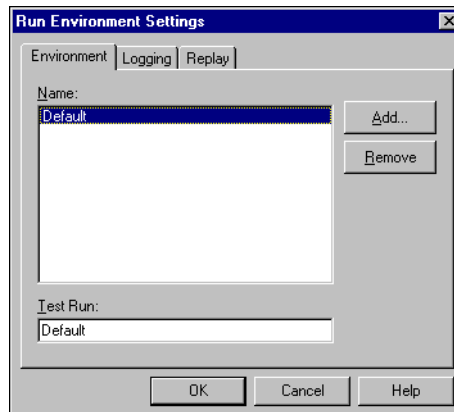
1. Click the **Run** button on *QARun*'s toolbar. The Summary Info dialog box displays the first time the script is run:



2. Enter a file name for the script such as *Exercise 2* (conventional Windows long file names are supported) and click **OK**. The Run Script dialog box displays.

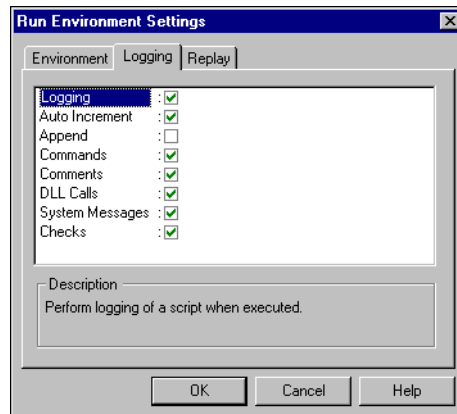
This dialog box allows you to determine how the script is run, if the script generates a log file, and what type of information gets logged. It also allows you to determine script replay and determine how long the script should wait for windows and controls to appear before timing out.

3. Click the **Edit** button from the Run Script dialog box. The Run Environment Settings dialog box displays:



The Environment tab allows you to add new environments to the list. Each environment may have a different group of settings and may be used by different scripts.

4. Highlight the Default environment from the list.
5. Click the Logging tab from the Run Environment Setting dialog box:
  - a. Select the Logging option by placing a check in the box.
  - b. Clear the Append option box by removing the check. The dialog box should now resemble the following example:

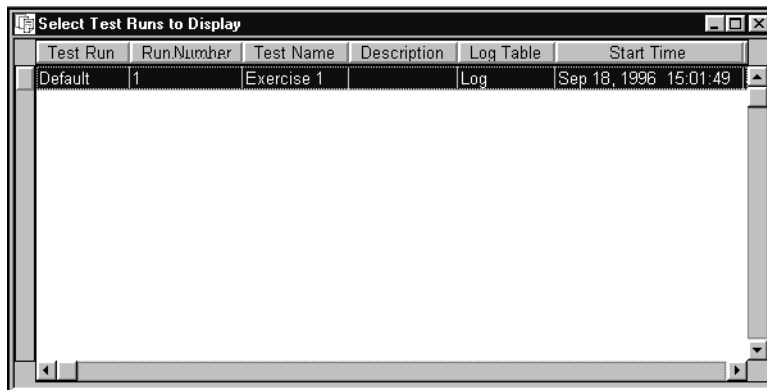


- c. Click **OK**. The Run Script dialog box reappears.
6. Click **OK** to begin running the script.

## Analyzing the Results

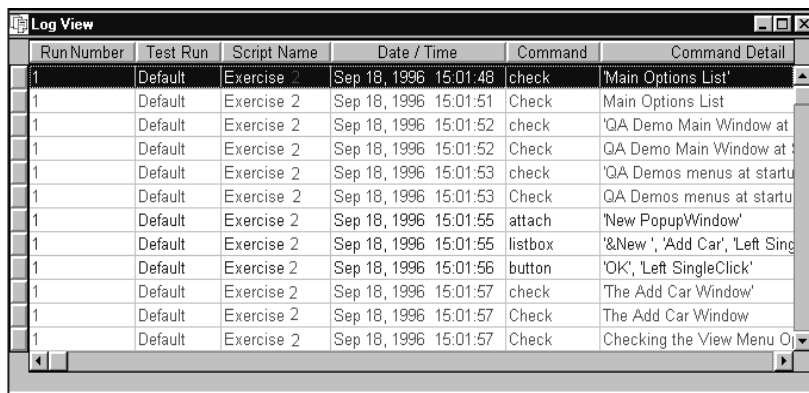
After you run the script, it becomes possible to view the log file. Use the following procedure to view the log file:

1. From the **File** menu, choose **Browse**.
2. Select **Log View** from the list and click **OK**. The following window displays:



Every time the test script runs, the script Run Number increments. This means that a complete log history is kept for reference.

3. Double-click the log file name. The window display changes to the following:



The result of each check is logged to the log file. If you successfully complete the exercises, your checks should have passed and now appear in green (the default color for checks that pass). Failed checks display in red.

No further action is necessary at this point. Later, when the entire test suite is run against QADemo Version 3, many of the checks will fail because the application has been modified. It will then be possible to analyze the differences between the expected and recorded results.

## Exercise Summary

After completing this exercise, you should be able to successfully:

- Create an individual test script.
- Insert list, window, and menu checks into the test script.
- Understand check syntax.
- Run a test script independent from the driver script.
- Use the log file to analyze your check results.

---

## Exercise 3 — Creating Form Checks

This exercise focuses on creating form checks using the Inquire option from the New dialog box list. This exercise requires QADemo Version 2 because QADemo Version 1 does not contain a database to perform an inquiry against.

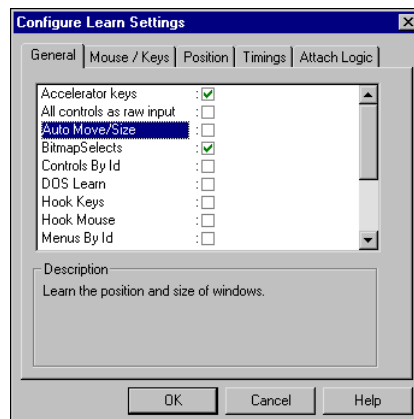
### Changing Configuration Options

Before you begin creating this test script, you need to change one of *QARun*'s configuration options and one of *QARun*'s replay options. Changing these options will make ensure that your test script replays properly.

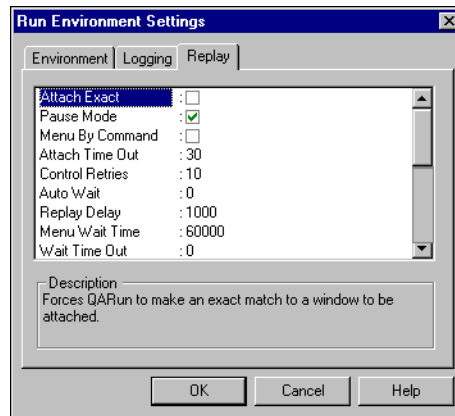
Use the following procedure to change these configuration options:



1. Click the **Editor** button on the Modules toolbar to open the Script Editor.
2. From the Script Editor's **Options** menu, choose **Learn Settings**. The Configure Learn Settings dialog box displays.
3. Click the **General** tab and clear the **Auto Move/Size** check box.



- From the **Options** menu, choose **Edit Run Environments**. The Run Environments Settings dialog box displays:



- Click the Replays tab and double-click the Attach Time Out option. The Attach Timeout dialog box displays.
- Enter a value of **30** in the New Value field and click **OK**.
- Click **OK** on the Run Environment Settings dialog box to close the dialog and return to the Script Editor.

## Getting Started

**Exercise Prerequisites:** Before beginning this exercise you must have completed the following prerequisites:

- Closed any open *QARun* scripts.
- Closed any open copies of *QADemo*.
- Changed *QARun*'s configuration options (see page 3-15).

**Testing Requirements:** The requirements for this test script are that it:

- Checks the contents of the Inquire dialog box list.
- Makes an inquiry.
- Checks the resulting dialog box.
- Makes an inquiry to see how many cars are in stock.
- Checks the resulting dialog box.
- Restores the system back to normal.

The following test elements are created during this exercise:

- Script named *Exercise 3*.
- List check named *Inquire List Items*.
- Form check named *Car Details Dialog*.
- Form check named *Stock Levels Form*.

## Learning the Script

Using the following procedures to move the target application to the point where you can start defining the test cases and begin learning the script:

1. Start *QARun* and open a new script.
2. Run QADemo Version 2:
  - Windows 95 or Windows NT 4 users:
    - Click the **Start** button from the taskbar and select **Run** from the menu. Enter the directory location for QADemo Version 2 in the Run dialog box. The default path is:  
`"C:\Program Files\Compuware\QARun\Demos\qademo.exe" /v2`
    - Click **OK**.
  - Windows NT 3.51 users:
    - From Program Manager's **File** menu, choose **Run**. Enter the directory location for QADemo Version 2 in the Run dialog box. The default path is:  
`C:\Program Files\Compuware\QARun\Demos\qademo.exe /v2`
    - Click **OK**.
3. Sign on to the system using the following information:
  - a. Enter **CW** in the User ID field.
  - b. Enter **PASS** in the Password field.
  - c. Select the London radio button as the site location.
4. Click **OK**.
5. Select Inquire from the New dialog box's main options list and click **OK**.
6. Press the **Learn Hotkey**, {**Alt {F10}**}, to begin learning the script.
7. Click the **Refresh** button on the Inquire dialog box to display the contents of the list field.
8. Create a list check on the contents of the list box field on this dialog box. To accomplish this:
  - a. Press the **Insert Check Hotkey**, {**Alt {F8}**}, and click **New** from the Browse Checks dialog box.
  - b. Select the List radio button from the New check dialog box to create a new list check and click **OK**.
  - c. Enter a name for the check such as *Inquire List Items* and click **Identify**.

- d. Position the pointer over the list box on the Inquire dialog box so that it changes to the target pointer and click once.

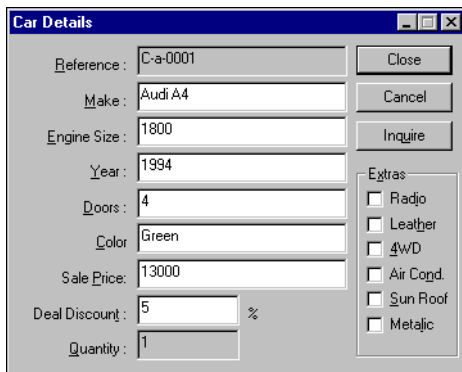
The List Check dialog box is restored and completed with the applicable control information.

- e. Click **OK** to save the check.

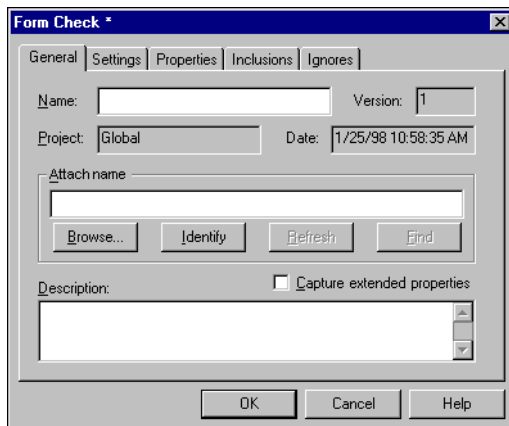
### Creating a Form Check

The next test requirement is to inquire about a car from the list and perform a check on the resulting dialog box. Use the following procedure to create a form check:

1. With Learn still on, double-click the first list item “Audi A4”. The Car Details dialog box displays:



2. Press the **Insert Check Hotkey, {Alt {F8}}**, and click **New** from the Browse Checks dialog box. The New check dialog box displays.
3. Select the Form radio button to create a new form check and click **OK**. The Form Check dialog box displays:



4. Enter a name such as *Car Details Dialog* in the Name field and click the **Identify** button.
5. Position the pointer over the Car Details dialog box title bar so that the text panel shows PopupWindow as the control type and click once.

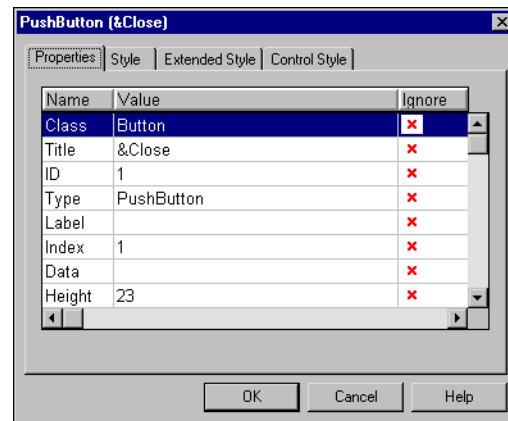
The Form Check dialog box reappears and is completed with the appropriate control details.

6. Click the Form Check dialog box's other tabs to view the available check options:
 

**Settings tab:** Determines how each control on the dialog box is checked. The options are as follows: Title and Class; Identifier and Class; and Title, Identifier, and Class.

**Properties tab:** Shows the contents of the dialog box in a tree view. Items can be added and deleted from the view and moved up and down.

Double-clicking on an item in the tree displays a details dialog box. For example, editing the **Close** button item displays the following dialog box:



Here, the control's properties are displayed and can be modified to reflect anticipated results.

**Inclusions tab:** Indicates which controls are included or excluded from the check. The default is to include everything.

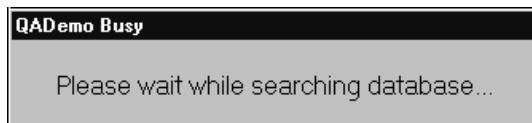
**Ignores tab:** Shows which aspects of the dialog box controls are ignored in the check. Aspects such as Count and Class can be used to scope the check to reflect expected results.

7. Click **OK** to save the check.

## Continuing the Script

The next test requirement is to perform an inquiry on stock levels of this car. Use the following procedure to perform an inquiry on the car (remember, Learn is still on):

1. Click the **Inquire** button from QADemo's Car Details dialog box. The QADemo Busy message box displays.



2. Wait for this dialog to disappear and then single-click on the Stock Levels at Other Sites dialog box when it appears.

This ensures that *QARun* attaches to this window before performing the next check.

3. Press the **Insert Check Hotkey**, {Alt {F8}}, and click the **New** button from the Browse Checks dialog box. The New check dialog box displays.
4. Select the Form radio button to create a new form check and click **OK**.
5. Create a form check that checks the Stock Levels at Other Sites dialog box and give the check a name such as *Stock Levels Form*.
6. Click the **OK** button on the Stock Levels at Other Sites dialog box after you have defined and saved the form check.
7. Click the **Close** button on the Car Details dialog box. This restores QADemo to the Inquire dialog box.
8. Press the **Learn Hotkey**, {Alt {F10}}, to stop learning.

## Running the Script

In a later exercise, you will add these test scripts to the driver script you created in “Exercise 1 — Creating a Driver Script”; however, you may run the script independently if QADemo is at the correct test site. For example, the beginning test site for this exercise is the QADemo Inquire dialog box. Use the following procedure to run the test script:

1. From the Run menu, choose **Run Script**.
2. The Summary Info dialog box displays the first time the script runs. Enter a file name for the script, such as *Exercise 3*. Conventional Windows long file names are supported.
3. Click **OK**.

## Exercise Summary

After completing this exercise, you should be able to successfully:

- Create a list check.
- Create a form check.
- Understand the form check tab options.
- Attach to message boxes to ensure proper script flow.
- Restore the target application to its pre-test state.

---

## Exercise 4 — Creating Clock Checks To Test Performance

This exercise again focuses on the QADemo Inquire option and requires QADemo Version 2.

In this exercise, you will build a performance test that makes an inquiry and then checks the response time for the Stock Levels at Other Sites dialog box to appear. You will then use the Repeat command to loop the script, thereby generating a true performance test.

### Getting Started

**Exercise Prerequisites:** Before beginning this exercise you must have completed the following prerequisites:

- Closed any open *QARun* scripts.
- Closed any open copies of QADemo.

**Testing Requirements:** The requirements for this test script are that it:

- Calls the Car Details dialog box by double-clicking on an entry in the list.
- Makes an inquiry to see how many cars are in stock.
- Performs a clock check to record how long it took the system to find the stock levels. The acceptable time is less than two seconds.
- Restores the system back to normal.

The following test elements are created during this exercise:

- Script named *Performance Test*.
- Window check named *Stock Levels at Other Sites*.
- Clock check named *Response Time*.
- Log Filter named *Performance Testing*.

## Learning the Script

Use the following procedure to run QADemo Version 2 and to start defining the test case against the target application:

1. Start *QARun* and open a new script.
2. Start QADemo Version 2 in the normal manner (see page 3-17 if necessary).
3. Sign on to the system using the following information:
  - a. Enter **CW** in the User ID field.
  - b. Enter **PASS** in the Password field.
  - c. Select the London radio button as the site location.
4. Click **OK**.
5. Select Inquire from the New dialog box's main options list and click **OK**.
6. Press the **Learn Hotkey**, {**Alt {F10}**}, to begin learning the script.
7. Click the **Refresh** button on the Inquire dialog box to display the contents of the list field.
8. Make an inquiry on a car in the list. To do this:
  - a. Double-click on the first list item, "Audi A4."
  - b. Click the **Inquire** button from the Car Details dialog box to make a stock inquiry at other sites.

## Defining an Event

To successfully create a performance test, you must tell the script what response from the target application actually signifies the end of the "Inquiry" process (in this case, it is the appearance of the Stock Levels at Other Sites dialog box). You must also instruct *QARun* on how to determine when that expected response has occurred.

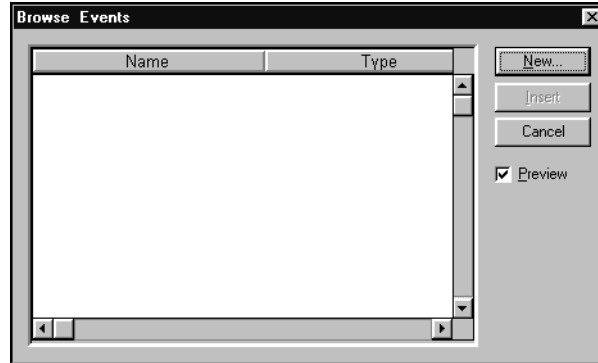
*QARun* uses *Events* to monitor these actions. *QARun* has an event-driven programming language that allows you to define user actions or system responses. Events are *QARun*'s way of monitoring system processing. *QARun* can recognize the following types of events:

- Text appearing on the screen.
- Typing on the keyboard.
- Making menu selections.
- Windows appearing, being moved, sized, etc.
- Mouse actions.
- Dates and times.
- Graphics appearing on the screen.

QARun can be instructed to wait for an event to occur before continuing with the script (using the Wait command), or it can be instructed to react whenever an event occurs (using the Whenever command).

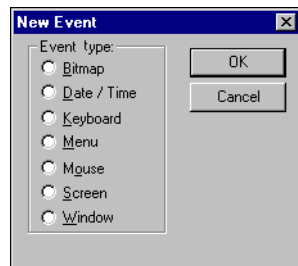
Use the following procedure to define an event for your test script:

1. Press the **Insert Event Hotkey**, {Alt {F7}}. The Browse Events dialog box displays:

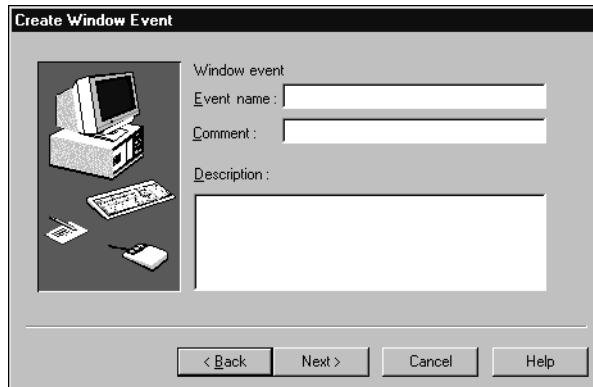


If you haven't defined any events yet, the list will be empty. Events appear in this list as they are created.

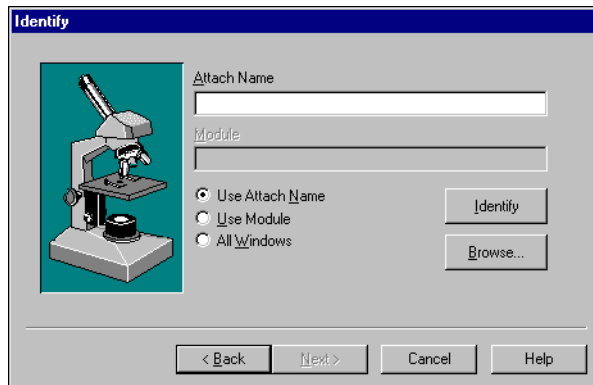
2. Click the **New** button on the Browse Events dialog box to define a new event. The New Event dialog box displays:



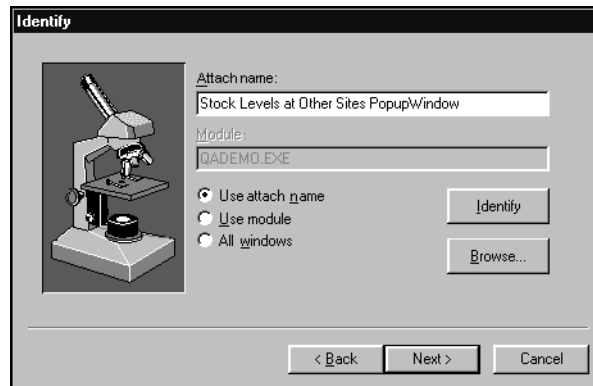
3. Select the **Window** radio button and click **OK**. The Create Window Event dialog box displays:



4. Enter a name, such as *Stock Levels at Other Sites* into the Event Name field.
  - The Event name field is mandatory, and each event must be given a unique name. The event name can be up to 50 characters and can include spaces.
  - The Description field is optional; however, a combination of descriptive event names and detailed descriptions makes it easier to identify the event's purpose in the script and in the event map.
5. Click **Next**. The Identify dialog box displays:

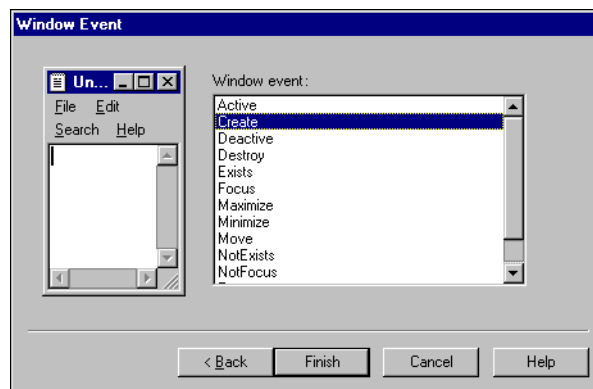


6. Click the **Identify** button. QARun minimizes and the cursor changes to a target pointer.
7. Position the pointer over the *Stock Levels at Other Sites* dialog box's title bar and click once. The Identify dialog box reappears and contains the window details:



- When defining a window event, it is possible to specify the window or program in which the event will trigger. It can be set to trigger in a specific application, module, or only within a particular window in the application.
- Use the Attach name option to indicate that the event will only trigger in this window. The Use module option specifies that the trigger will work wherever you are within the application.

8. Click **Next**. The Window Event dialog box displays:



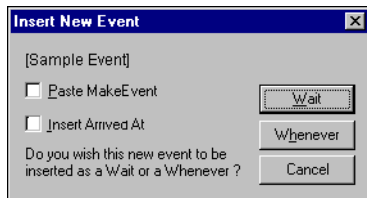
This dialog box determines the window actions that will trigger the event. The list contains all possible window actions.

You can set an event to trigger whenever a window is activated, minimized, maximized, restored, destroyed (closed), moved, sized, created (invoked or run), or deactivated (out of focus), etc.

- From the Window event list, select Exists.

This ensures that QARun Waits until the dialog box exists before continuing the script. Exists is different than Created because a window can be created long before it appears on the screen. The Exists option satisfies the event only when the window is physically on the screen, enabled, or disabled.

- Click the **Finish** button on the Window Event dialog box. The following dialog box displays:

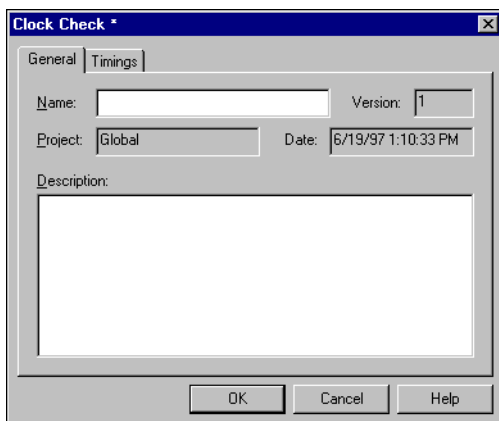


- Ensure that the Paste MakeEvent and Insert Arrived At check boxes are not selected and click **Wait**. The event is pasted into the script as a Wait statement.

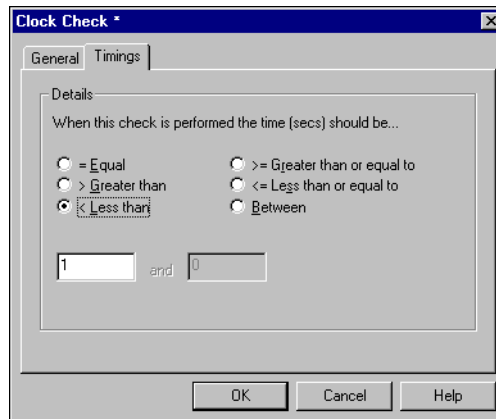
## Creating a Clock Check

The next test requirement is to define a clock check that will hold the expected response time of the system. The response time will be checked against the actual time the check is called. Use the following procedure to insert a clock check into your test script:

- Press the **Insert Check Hotkey, {Alt {F8}}**.
- Click the **New** button from the Browse Checks dialog box. The New Check dialog box displays.
- Select the **Clock** radio button and click **OK**. The Clock Check dialog box displays:



4. Enter a name such as *Response Time* and click the Timings tab. The display changes to the following:



5. Ensure that the Less than radio button is selected.
6. Enter the value 2 in the Value field.

Defining the parameters for the clock simply sets the check value. It is also necessary to instruct *QARun* when to start, stop, and reset the clock.

7. Click **OK** to save the check definition.

## Continuing the Script

The last test requirement is to restore *QADemo* to its normal state. Use the following procedure to restore *QADemo* to its original state:

1. Click the **OK** button on the Stock Levels at Other Sites dialog box.
2. Click the **Close** button on the Car Details dialog box.
3. Press the **Learn Hotkey**, **{Alt {F10}}** to turn Learn off.

## The Resulting Script

After completing the above steps, your test script should look very similar to the following example. Some Attach syntax may vary slightly, depending upon your Learn options. In this example, the new event and check syntax appears in bold typeface for identification purposes.

```
Function Main
Attach "Inquire ChildWindow~1_0003"
    Button "&Refresh", 'Left SingleClick'
    ListBox "@Inquire ListBox~1", c"C-a-0001\tAudi A4",
        'Left SingleClick'
    ListBox "@Inquire ListBox~1", c"C-a-0001\tAudi A4",
        'Left DoubleClick'
Attach "Car Details PopupWindow_0000"
    Button "Inq&uire", 'Left SingleClick'
If Wait(30, "", "Stock Levels at Other Sites") = 1
    ;
    ; Event Passed
    ;
Else
    ;
    ; Timeout of 30 seconds has been exceeded
    ;
EndIf
Check "Response Time"
Attach "Stock Levels at Other Sites PopupWindow"
    Button "OK", 'Left SingleClick'
Attach "Car Details PopupWindow_0000"
    Button "Close", 'Left SingleClick'
End Function ; Main
```

## Modifying the Script

The default Wait syntax for the event is the standard If...Else...Endif logic. The Wait command, followed by the value 30, forces *QARun* to pause the script for 30 seconds. Therefore, the script syntax causes the script to wait 30 seconds for the event to occur before timing out and processing the Else argument. The timeout value is set in the Script Editor's Configure dialog box and can be set to any value.

You must make some modifications to the event logic to force the script to wait indefinitely for the event to occur. The If...Else...Endif construct is not necessary and can be removed. Changing the default wait time to "0" causes the script to wait indefinitely for the event to occur. Change the syntax to read as follows:

```
Wait ( 0, "For", "Stock Levels at Other Sites")
```

The word "For" in the Wait Statement is a “noise word” and is included to make the statement more understandable.

There are three commands associated with clock checks that make the clock behave in the same way as a stopwatch. The associated *QARun* commands are:

```
ClockReset <Clock Name>
ClockStart <Clock Name>
ClockStop <Clock Name>
```

You should insert these commands into your test script at the appropriate points. The clock should be reset at the beginning of the script, then started when the **Inquire** button is clicked, then stopped when the Wait statement is satisfied but before the check is carried out.

Script modifications appear in bold typeface to assist you in identifying the differences. Change your test script to the following:

```
Function Main
;Reset the clock to zero
ClockReset "Response Time"
Attach "Inquire ChildWindow~1_0003"
    Button "&Refresh", 'Left SingleClick'
ListBox "@Inquire ListBox~1", c"C-a-0001\tAudi A4",
    'Left SingleClick'
ListBox "@Inquire ListBox~1", c"C-a-0001\tAudi A4",
    'Left DoubleClick'
;Start the inquiry
Attach "Car Details PopupWindow_0000"
    Button "Inquire", 'Left SingleClick'
    ;Start the clock
    ClockStart "Response Time"
    ;The Event Logic
    Wait( 0, "For", "Stock Levels at Other Sites")
    ;When Event happens stop the clock
    ClockStop "Response Time"
    ;Compare values
    Check "Response Time"
Attach "Stock Levels at Other Sites PopupWindow"
    Button "OK", 'Left SingleClick'
Attach "Car Details PopupWindow_0000"
    Button "Close", 'Left SingleClick'
End Function ; Main
```

## Additional Script Modifications

This script is almost ready to run. In fact, you can test run it now. However, the script will only perform the Inquire operation once. As stated at the beginning of this exercise, the purpose of the script is to repeat the Inquire process several times. In order to do this, some extra code must be inserted into the script. The *QARun* scripting language contains special commands that cause scripts to repeat lines of code until a predefined event occurs.

The Repeat...Until syntax is as follows:

```
Repeat
    <instructions>
Until <event>
```

The event may be internal or external. Internal events are usually based on a numeric variable equaling a given number. External events are usually defined as keyboard, mouse, window, or screen events.

Your test script will use an internal event — a numeric variable — that stops the script from looping.

We recommend reducing *QARun*'s logging activity while performance testing because each time *QARun* writes to the log file, it affects the recorded results of the clock check.

Although it is possible to filter logging before running the script, these filters affect the entire script. In this situation, it is better to prevent logging at specific points in the script. This is achieved using the LogOff and LogOn commands. These commands can be used at any point in the script. The syntax is as follows:

```
LogOff( "FunctionType" )
```

Where "FunctionType" is the name of the function to be disabled, such as Attach, Type, etc. An asterisk, "\*", can be used to disable logging for all functions.

Modify the script to contain the following syntax. The added syntax appears in bold typeface:

```

Function Main
;Set up a numeric variable called "count"
Count = 0
;Set up the Repeat Loop Logic
Repeat
    ;Reset the clock to zero
    ClockReset "Response Time"
Attach "Inquire ChildWindow~1_0003"
    Button "&Refresh", 'Left SingleClick'
        ListBox "@Inquire ListBox~1", c"C-a-0001\tAudi A4",
            'Left SingleClick'
        ListBox "@Inquire ListBox~1", c"C-a-0001\Audi A4",
            'Left DoubleClick'

;Start the inquiry
Attach "Car Details PopupWindow_0000"
    Button "Inq&uire", 'Left SingleClick'
        ;Turn OFF logging at this stage
        LogOff("***)

;Start the clock
ClockStart "Response Time"

;The Event Logic
Wait( 0, "For", "Stock Levels at Other Sites")

;When Event happens stop the clock
ClockStop "Response Time"

        ;Turn all Logging back ON
        LogOn("***)

;Compare values
Check "Response Time"
Attach "Stock Levels at Other Sites PopupWindow"
    Button "OK", 'Left SingleClick'
Attach "Car Details PopupWindow_0000"
    Button "Close", 'Left SingleClick'
        ;Increment the variable by one
        Count = Count + 1
;Repeat the script until the variable equals 5
Until Count = 5
End Function ; Main

```

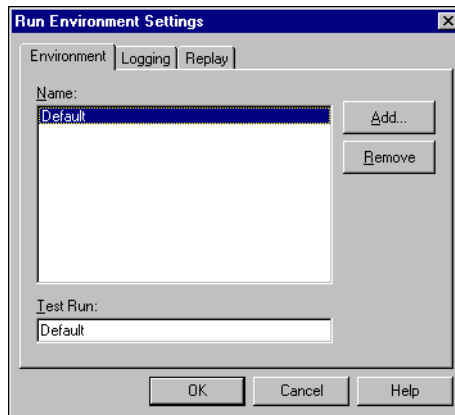
## Running the Script

After completing the above modifications, you are ready to run your test script. Use the following procedure to run the test script:

1. From the **Run** menu, choose **Run Script**.

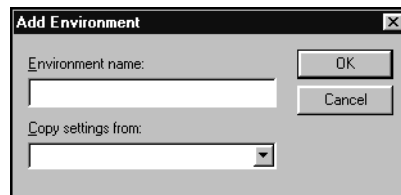
The Summary Info dialog box is displayed because this is the first time the script has been run.

2. Enter a file name for the script, such as *Performance Test* and click the **OK** button. The Run Script dialog box displays.
3. Click the **Edit** button on the Run Script dialog box. The Run Environment Settings dialog box displays:



The Environment tab allows you to add new environments to the list. A run environment is a collection of runtime settings. Each environment may have different settings that can be used by different scripts at runtime.

4. Click **Add** to create a new run environment. The Add Environment dialog box displays:

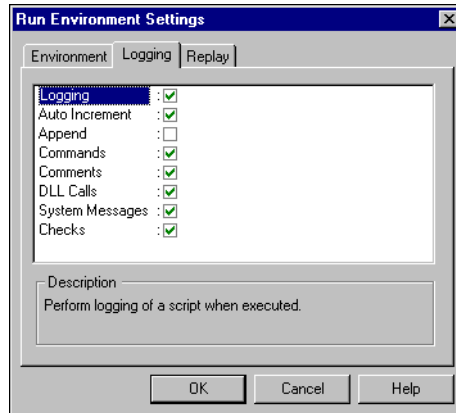


5. Enter a name for the new environment, such as *Performance Testing*.

The Copy settings from drop-down list allows you to select the settings from any existing run environment.

6. Click **OK**. The Run Environment Settings dialog box re-displays.

7. Select Logging tab.
  - a. If necessary, click the Logging check box to select it.
  - b. If necessary, click the Log Append option to clear this option. The dialog box should now appear as follows:



Although all other options are set to be logged, this script actually turns their logging off internally. Therefore, these options can be ignored.

8. Click **OK** and the Run Script dialog box reappears.
9. Click the **OK** button on the Run Script dialog box. The script should run and generate a log file.

## Analyzing the Results

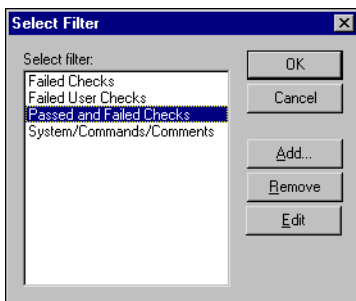
After you run the script, it is possible to view the resulting log file. Use the following procedure to view the log file:

1. From the **File** menu, choose **Browse**
2. Select **Log View** from the list.
3. Double-click on the Performance Test log file name in the Browse Logs dialog box. The log file opens:

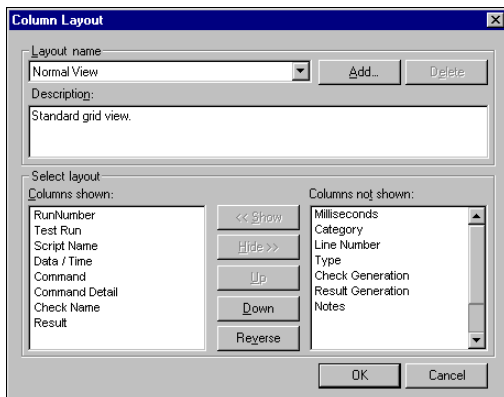
RunNumber	Test Run	Script Name	Date / Time	Command	Command Detail
1	Default	Performance Test	Oct 21, 1996	Start	Performance Test
1	Default	Performance Test	Oct 21, 1996	clockreset	'Response Time'
1	Default	Performance Test	Oct 21, 1996	attach	'Enquire ChildWindow~1'
1	Default	Performance Test	Oct 21, 1996	button	'&Refresh', 'Left SingleClick'
1	Default	Performance Test	Oct 21, 1996	attach	'Enquire ChildWindow~1'
1	Default	Performance Test	Oct 21, 1996	listbox	'@Enquire ListBox~1', 'Click'
1	Default	Performance Test	Oct 21, 1996	listbox	'@Enquire ListBox~1', 'Click'
1	Default	Performance Test	Oct 21, 1996	attach	'Car Details PopupWindow'
1	Default	Performance Test	Oct 21, 1996	button	'Inquire', 'Left SingleClick'
1	Default	Performance Test	Oct 21, 1996	logoff	''
1	Default	Performance Test	Oct 21, 1996	Check	Response Time
1	Default	Performance Test	Oct 21, 1996	attach	'Stock Levels at Other Sites'
1	Default	Performance Test	Oct 21, 1996	button	'OK', 'Left SingleClick'
1	Default	Performance Test	Oct 21, 1996	attach	'Car Details PopupWindow'
1	Default	Performance Test	Oct 21, 1996	button	'Cancel', 'Left SingleClick'

You can apply various filters to the log, including the ability to filter the log details to display only checks.

- From the **View** menu, choose **Select Filter**. The Select Filter dialog box displays:

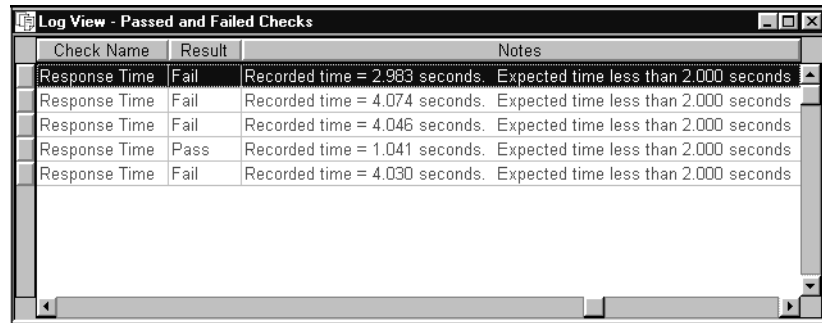


- Select **Passed and Failed Checks** from the Select filter list and click **OK**.
- Click the **Grid Column Layout** button on the Log View toolbar. The Column Layout dialog box displays:



7. Ensure that the Notes option appears in the Columns shown list. This allows you to view the expected and actual times for the clock check.
8. Click **OK**.

When the Log View display returns, scroll the horizontal bar until you can see the Notes column. The Log View changes to resemble the following example:



Check Name	Result	Notes
Response Time	Fail	Recorded time = 2.983 seconds. Expected time less than 2.000 seconds
Response Time	Fail	Recorded time = 4.074 seconds. Expected time less than 2.000 seconds
Response Time	Fail	Recorded time = 4.046 seconds. Expected time less than 2.000 seconds
Response Time	Pass	Recorded time = 1.041 seconds. Expected time less than 2.000 seconds
Response Time	Fail	Recorded time = 4.030 seconds. Expected time less than 2.000 seconds

From the Notes column, you can view the recorded and the expected performance of the timings included in the clock check.

## Exercise Summary

After completing this exercise, you should be able to successfully:

- Define an window event.
- Create a clock check.
- Insert Wait commands into your test scripts.
- Use Repeat logic in your scripts.
- Use loops to create performance testing.
- Turn logging on and off.
- View your script log.

---

## Exercise 5 — Using External TestData Files

An important testing capability is the ability to enter variable data into the target application — to test correct behavior with differing input or to test behavior under heavy-load conditions. QARun is able to enter data into the target application using structured data input.

The purpose of this exercise is to build a test script that carries out a volume test by reading a data file and entering the information into QADemo.

When complete, this test script should be able to add a customer record and update the QADemo database. The test script will use the Repeat command and some special system functions to Loop several times, entering a new customer record each time it loops.

### Getting Started

The driver script commands the target application to select the Customer Invoice option from the New dialog box main options list and then invokes the Invoice Screen. The Invoice screen has the Add New Customers option.

**Exercise Prerequisites:** Before beginning this exercise you must have completed the following prerequisites:

- Closed any open QARun scripts
- Closed any open copies of QADemo.

**Testing Requirements:** The requirements for this test script are that it:

- Invoke the Add Customer Entry Screen and add a customer to the database using an external testdata file.
- Make a performance check to determine how long it takes the database to update with the new record. The acceptable time is less than two seconds.

The following test elements are created during this exercise:

- Script named *Adding New Customers*.

### Learning the Script

You are now ready to define the test case against the target application. Use the following procedure to advance QADemo to the proper test site and to create the test script:

1. Run QARun and open a new script.
2. Start QADemo Version 2 in the normal manner (see page 3-17 if necessary).

3. Sign on to the system using the following information:
  - a. Enter **CW** in the User ID field.
  - b. Enter **PASS** in the Password field.
  - c. Select the London radio button as the site location.
4. Click **OK**.
5. Select Customer Invoice from the New dialog box's main options list and click **OK**.
6. Press the **Learn Hotkey**, {Alt {F10}}, to start learning the script.
7. Click the **New Customer** button on the Customer Invoice display. The New Customer Details dialog box displays.
8. Complete the Name and Address fields and click **New** to add the record to the database. A QADemo confirmation box displays.
9. Click **OK** to confirm the addition.
10. Press the **Learn Hotkey**, {Alt {F10}}, to turn Learn off.

## The Resulting Script

After completing the above steps, the test script should look very similar to the following example:

```
Function Main
Attach "Customer Invoice ChildWindow"
  Button "&New Customer", 'Left SingleClick'
Attach "New Customer Details PopupWindow"
  EditText "&Name :", "Sam Smith"
Attach "~N~QADEMO.EXE~Edit~&Address : "
  Type "31440 Northwestern Highway{Return}"
  Type "Farmington Hills{Return}"
  Type "Michigan 48334-2564{Return}"
Attach "New Customer Details PopupWindow"
  Button "Ne&w", 'Left SingleClick'
Attach "QADemo PopupWindow"
  Button "OK", 'Left SingleClick'
End Function ; Main
```

The remaining sections of this exercise discuss how to modify the script to add more than one record into the database and execute a performance check each time a new record is added. You will add the new customer records using an external testdata file. The following section explains how *QARun* works with testdata files.

## Understanding TestData Files

TestData files provide an efficient way for scripts to access external data. Using testdata files allows you to separate your script logic from its data. For example, inputting 500 new database entries requires only a single script entry. The script can read the 500 sets of input data from an external testdata file at runtime.

A testdata file is a comma separated variable (CSV) file where each line constitutes a record. Each record contains fields that are separated by commas. Consider the following data file. Here, there are three records, each with six fields exported from a database application as a CSV file:

```
J. Smith,The Mall,London,TW7 4DS,UK,8471666
J. Cotez,Ave St Maria,Madrid,987621,Spain,98762301
A. Gilbert,Rue Albert,Issigeac,24679,France,53234512
```

There should not be any spaces between the fields and their comma separators. Fields containing commas may be included within the testdata file if they are enclosed in double quotes. For example:

```
Compuware,31440 Northwestern Highway,"Farmington Hills, MI"
Compuware Ltd,"163, Bath Road","Slough, SL1 4AA"
```

QARun can read the contents of this data file. To specify the data file from within the script, you set the function TestData() to the name of the file, for example:

```
TestData( "CUSTOMER.CSV" ) ;Use default QARun Directory
```

or:

```
TestData( "C:\TESTDATA\CUSTOMER.CSV" );Specify directory
```

QARun does not use the original CSV file, but instead generates a special indexed file when the script is run — customer.INX. This special index file is automatically generated if it does not exist or if the date does not match the CSV data file date. The index file looks like the following:

```
0 0      Field 1      Field 2      Field 3      .....
Record 1 J. Smith,  The Mall,   London,     .....
Record 2 J. Cotez,   Ave St Maria, Madrid,     .....
Record 3 A. Gilbert, Rue Albert,  Issigeac,  .....
```

QARun's record pointer is automatically positioned just before the first field at position 0 0. The first record in the file is record 1, and the first field of each record is field 1.

Every time you change the testdata file name by setting TestData(), the current field and record markers are "set" to 0, even if you use the same data file name.

This indexed file is referenced by the special Type command that is used to access each record and field in turn. The Type syntax is:

```
Type "{Record.Field}"
```

For example:

```
Type "{1.1}{Return}"           ; First Record - First Field
Type "{1.2}{Return}"           ; First Record - Second Field
```

The results of the above would be:

```
J. Smith
The Mall
```

The following wildcard characters may be used instead of explicit record/field references:

```
=      Same record or field
+      Next record or field
-      Previous record or field
*      Random record or field
```

Therefore:

```
Type "{=.6}"                   ; Selects 6th field of the current record
Type "{+.4}"                   ; Selects 4th field of the next record
Type "{=.+}"                   ; Selects next field of the current record
Type "{*.5}"                   ; Selects 5th field of any record
Type "{*.*}"                   ; Selects any field from any record
```

The example script on page 3-41 reads the data file, Customer.csv, creates an index file, customer.INX, and then enters the data into the QADemo database. The program loops around and continues entering data until it reaches the end of the data file.

## Looping

The *QARun* commands `TestDataFieldCount()` and `TestDataRecordCount()` tell you the number of records and fields per record that are contained in the current data file.

The `TestDataCurField()` and `TestDataCurRecord()` commands tell you the record and field that is currently selected.

Together, these four commands allow you to build loops that can access each record and field in the testdata file. The script modifications described in the next section allow the script to access the testdata file Customer.csv and enter each item from the testdata file into the script.

## Modifying the Script

1. The first modification that you will make is to add an instruction to the script that tells QARun which testdata file to use. Add the following code after the Function Main command and before the main body of the script:

```
TestData("customer.csv")
Repeat           ; Also start the repeat until loop
```

2. Next, you will build the record entry logic that extracts records from the testdata file and enters them into QADemo. Because the first entry field is an edit control, this is accomplished using the TestDataTransform() function. This function is used to extract values from a testdata file, either to use in controls that cannot be “typed” in (such as Edit controls) or to enable processing of the value before passing it on to the target application.

The section of the script that reads:

```
Attach "New Customer Details PopupWindow"
    EditText "&Name :", "Sam Smith"
```

Must change to read the first field of the first record and then enter that data into the edit control. Replace the above Attach statement with the following syntax:

```
Attach "New Customer Details PopupWindow"
CustomerName = TestDataTransform("{+.1}")
    EditText "&Name :", CustomerName
```

Where CustomerName is a variable that contains the result of the TestDataTransform() function. The "{+.1}" parameter is the instruction to read the first/next record and the first field from the testdata file.

3. The value you entered into the edit control during Learn must now be replaced with the variable CustomerName. The next script change is the section that reads:

```
Attach "~N~QADEMO.EXE~Edit~&Address :"
    Type "31440 Northwestern Highway{Return}"
    Type "Farmington Hills{Return}"
    Type "MI 48334-2564{Return}"
```

Here, the script attaches to the multi-line edit control. Because this control allows the Type command to be used, it is not necessary to use the TestDataTransform() function to read the information from the testdata file. Change the script so that it reads as follows:

```
Attach "~N~QADEMO.EXE~Edit~&Address :"
    Repeat           ;Start a second repeat loop
        Type "{=.+}"
        Type "{Return}"
    Until TestDataCurField = TestDataFieldCount
```

The script does an Attach to the multi-line edit control. A second Repeat loop is started that reads each field from the current record until the end of the record is reached.

The Repeat loop is terminated when both the TestDataCurField() and TestDataFieldCount() functions match.

4. The final modification is to “close” the first Repeat loop that was started at the beginning of the script. At the very end of the script, before the End Function; Main command, add the following line of code:

```
Until TestDataCurRecord = TestDataRecordCount ; end of
"record"
```

The first Repeat loop continues until the end of the data file is encountered. The second loop continues until the last field in each record is reached.

## The Modified Script

The finished script should now look similar to the following example:

```
Function Main
TestData("Customer.csv")
Repeat
  Attach "Customer Invoice ChildWindow~1_0002"
  Button "&New Customer", 'Left SingleClick'
  Attach "New Customer Details PopupWindow"
  CustomerName = TestDataTransform("{+.1}")
  EditText "&Name :", CustomerName
  Attach "~N~QADEMO.EXE~Edit~&Address :"
    Repeat
      Type "{=.+}"
      Type "{Return}"
    Until TestDataCurField = TestDataFieldCount
  Attach "New Customer Details PopupWindow"
  Button "Ne&w", 'Left SingleClick'
  Attach "QADemo PopupWindow"
  Button "OK", 'Left SingleClick'
Until TestDataCurRecord = TestDataRecordCount ; end of record loop
End Function ; Main
```

## Running the Script

When you run this script, *QARun* will extract information from an external testdata file and automatically enter it into the New Customer Details dialog box. Run the script using the following procedure:

1. Save the script as *Adding New Customers*.
2. Once the script has finished running, from the **File** menu, choose **Reset Database** to reset the QADemo database.

## Exercise Summary

After completing this exercise, you should be able to successfully:

- Understand how testdata files are read.
- Read testdata record information into your test script.
- Use testdata functions to build loops into your test script.

---

## Exercise 6 — Creating Clock Checks for External Data Entry

This exercise builds a test script that reads records from an external data file and then adds the records to the QADemo database. This exercise addresses the second test requirement mentioned in the previous exercise — making a performance check to determine how long it takes the database to update with the new record.

## Getting Started

**Exercise Prerequisites:** Before beginning this exercise you must have completed the following prerequisites:

- Closed any open *QARun* scripts.
- Closed any open copies of QADemo.
- Access to the script named *Adding New Customers* (created in “Exercise 5 — Using External TestData Files”).
- Access to the clock check *Response Time* (“Exercise 4 — Creating Clock Checks To Test Performance”).

**Testing Requirements:** The requirements for this test script are that it:

- Makes a performance check to determine how long it takes the database to update with the new record. The acceptable time is less than two seconds.

The following test elements are created during this exercise:

- Script named *Adding New Customers*.
- Window event named *Adding Customer Record*.

## Learning the Script

Use the following procedure to start building your test script:

1. Start *QARun* and open the *Adding New Customers* script that you created during “Exercise 5 — Using External TestData Files”.
2. Start *QADemo Version 2* in the normal manner (see page 3-17 if necessary).
3. Sign on to the system using the following information:
  - a. Enter **CW** in the User ID field.
  - b. Enter **PASS** in the Password field.
  - c. Select the London radio button as the site location.
4. Click **OK**.
5. Select Customer Invoice from the New dialog box’s list. The Customer Invoice dialog box displays.
6. Click the **New Customer** button on the Customer Invoice dialog box. The New Customer Details dialog box displays.
7. Add a fictitious record and click the **New** button. A confirmation dialog box displays. Do **not** click this dialog box.



---

Do not click the **OK** button on the New Customer confirmation dialog box. This dialog box must be active to complete future script modifications.

If you inadvertently clicked **OK**, you may add another fictitious record, and the confirmation dialog box will reappear.

---

8. With *QADemo* still open and the confirmation dialog box still open, toggle to the *QARun* script. You will now begin making the script modifications that are necessary to successfully complete this exercise.

## Modifying the Script

Similar to “Exercise 4 — Creating Clock Checks To Test Performance”, the necessary clock handling and checking must now be added to this script to run a performance test.

We recommend reducing *QARun*’s logging activity while performance testing. Although it is possible to filter logging before running the script, these filters affect the entire script. In this situation, it is better to prevent logging at specific points. This is achieved by the use of the `LogOff` and `LogOn` commands. These commands can be used at any point in the script.

Use the following procedure to modify your test script:

1. Insert a new line above the `TestData( )` command.
2. Add the following syntax to the top of the script:

```
LogOff(" *") ; Turn all logging off
```

3. Add the following syntax on a new line following the `Repeat` statement:

```
ClockReset "Response Time" ; Reuse the clock check earlier
```

This statement resets the clock to zero — just like resetting a stopwatch. Because the expected response time in this case is the same as the one created in “Exercise 4 — Creating Clock Checks To Test Performance”, the same check can be used again. It is not necessary to define another clock check.

4. Insert a performance check where the script adds the customer record, selects the **New** button, and the confirmation dialog box displays. This modification affects the following area of the script:

```
Attach "New Customer Details PopupWindow"
    Button "Ne&w", 'Left SingleClick'
Attach "QADemo PopupWindow"
    Button "OK", 'Left SingleClick'
```

The clock must be started after the **New** button has been pressed and stopped after the dialog box displays, but before the **OK** button is pressed. This means a window event must be defined so that the script knows when to stop the clock.

5. Position the cursor between the logic for processing the **New** button and the `Attach` statement for the `PopupWindow`.
6. Add the following to the script:

```
ClockStart "Response Time"
```

7. Position the cursor on a blank line following the `Attach "QADemo PopupWindow"` and add the following line of code to the script:

```
ClockStop "Response Time"
```

## Defining an Event

The confirmation dialog box displayed by the application is considered to be a window. *QARun* must wait for this window to appear before processing the rest of the script.

Because *QARun* is not in Learn mode, it is not possible to press the **Insert Event Hotkey**. So, you'll use another method to insert events from the Script Editor:

1. Position the cursor on a blank line following the `ClockStart "Response Time"` statement.
2. From *QARun*'s **Insert** menu, choose **Event\Window**. The Browse Window Events dialog box displays.
3. Click the **New** button on the Browse Window Events dialog box. The Create Window Event dialog box displays.
4. Enter a name for the event, such as *Adding Customer Record*, and click **Next**. The Identify dialog box displays.
5. Click **Identify**. *QARun* minimizes and the cursor changes to the target pointer.
6. Position the pointer over the confirmation dialog box's title bar and click once. The Identify dialog box reappears and now contains the window details.
7. Click the **Next** button.
8. Select **Exists** from the Window Event scroll list.

This ensures that *QARun* waits until the dialog box exists before continuing the script. The **Exists** option is different from the **Created** option because a window can be created before it displays on the screen. The **Exists** option satisfies the event only when the window is physically on the screen, enabled, or disabled.

9. Click **Finish**. The Insert New Event dialog box displays.
10. Click the **Wait** button. The event will be pasted into your script.

## Additional Script Modifications

To force the script to wait indefinitely for the event to occur, some modifications must be made to the event logic.

If present, the `If...Else...EndIf` construct is not needed and can be removed. Also, if necessary, changing the default wait time to "0" will cause the script to wait indefinitely for the event to occur. Change the syntax so that it reads as follows:

```
Wait ( 0, "For", "Adding Customer Record" )
```

The word `"FOR"` in the `Wait` statement is a "noise word" and is only included to make it more understandable.

Next, add the following under the ClockStop command:

```

LogOn( "*" ) ; turn logging back on
Check "Response Time" ; Add already defined clock check
LogOff( "*" ) ; Turn logging off again

```

## The Resulting Script

The script should now look similar to the following example:

```

Function Main
LogOff("*) ; Turn logging off
TestData("customer.csv")
Repeat
Clockreset "Response Time" ; Reset the clock
  Attach "Customer Invoice ChildWindow~1_0002"
  Button "&New Customer", 'Left SingleClick'
  Attach "New Customer Details PopupWindow"
  CustomerName = TestDataTransform("{+.1}")
  EditText "&Name :", CustomerName
  Attach "~N~QADEMO.EXE~Edit~&Address :"
  Repeat
    Type "{=.+}"
    Type "{Return}"
  Until TestDataCurField = TestDataFieldCount
  Attach "New Customer Details PopupWindow"
  Button "Ne&w", 'Left SingleClick'
  ClockStart "Response Time" ; Start the clock
  Wait( 0, "For", "Adding Customer Record") ; Window Event
  Attach "QADemo PopupWindow"
  ClockStop "Response Time" ; Stop the clock
  LogOn("*) ; Turn logging back on
  Check "Response Time" ; Run the check
  LogOff("*) ; Turn logging off again
  Button "OK", 'Left SingleClick'
Until TestDataCurRecord = TestDataRecordCount; end of record loop
End Function ; Main

```

## Running the Script

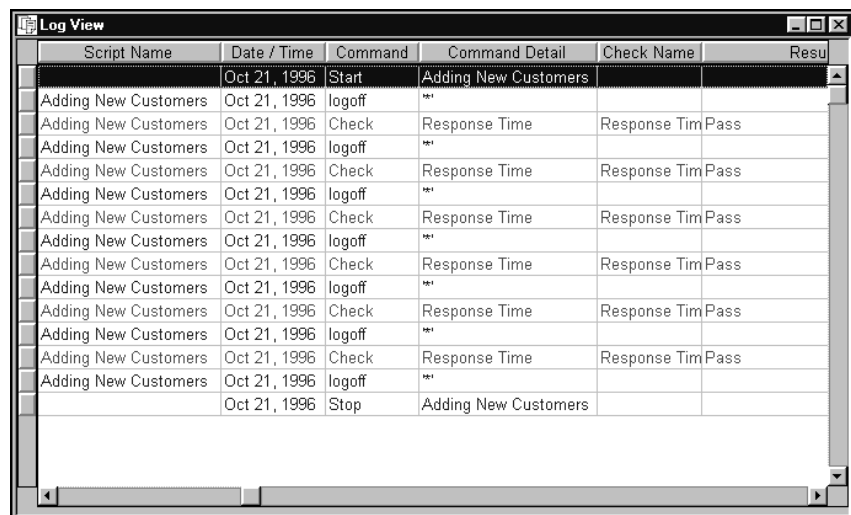
This script was already saved and run before you began editing in this exercise. Therefore, when you run it this time, you will not be asked to supply a name and the Summary Info dialog box will not appear.

1. Before running the script, close the QADemo New Customer confirmation box.
2. Select a predefined run environment from the Run dialog box. This script bypasses any previously defined replay options because it handles the switching off and on of logged statements internally.
3. Click **OK** to run the script.
4. Once the script has finished running, from QADemo's **File** menu, choose **Reset Database** to reset the QADemo database.

## Analyzing the Results

Once the script has run, it is possible to view the resulting log file. To view the log file:

1. From the **File** menu, choose **Browse**.
2. Select Log View from the list and click **OK**.
3. Double-click the *Adding New Customers* log file name in the Browse Logs window. The log file opens and should look similar to the following example:



Script Name	Date / Time	Command	Command Detail	Check Name	Resu
	Oct 21, 1996	Start	Adding New Customers		
Adding New Customers	Oct 21, 1996	logoff	**		
Adding New Customers	Oct 21, 1996	Check	Response Time	Response Tim	Pass
Adding New Customers	Oct 21, 1996	logoff	**		
Adding New Customers	Oct 21, 1996	Check	Response Time	Response Tim	Pass
Adding New Customers	Oct 21, 1996	logoff	**		
Adding New Customers	Oct 21, 1996	Check	Response Time	Response Tim	Pass
Adding New Customers	Oct 21, 1996	logoff	**		
Adding New Customers	Oct 21, 1996	Check	Response Time	Response Tim	Pass
Adding New Customers	Oct 21, 1996	logoff	**		
Adding New Customers	Oct 21, 1996	Check	Response Time	Response Tim	Pass
Adding New Customers	Oct 21, 1996	logoff	**		
Adding New Customers	Oct 21, 1996	Check	Response Time	Response Tim	Pass
Adding New Customers	Oct 21, 1996	logoff	**		
	Oct 21, 1996	Stop	Adding New Customers		

Your results may differ depending on how well the target application performed.

## Exercise Summary

After completing this exercise, you should be able to successfully:

- Create a performance check for database modifications.
- Insert an event without being in Learn mode.

---

## Exercise 7 — Creating Text Checks

The following exercise creates a test script that sells a car to an existing customer and generates an invoice. When inserted into the driver script, the script will take over from the testdata entry and performance scripts created in previous exercises and conduct a text check.

By using a text check, it is possible to highlight areas of the screen (typically fields of data) and analyze the values displayed. You can check dates and numeric values and ensure that fields contain alpha or alphanumeric strings. Multiple fields within a window may be defined in a single check.

## Getting Started

**Exercise Prerequisites:** Before beginning this exercise you must have completed the following prerequisites:

- Closed any open copies of *QARun*.
- Closed any open copies of *QADemo*.

**Testing Requirements:** The requirements for this test script are that it:

- Sells a car to an existing customer.
- Generates and prints an invoice.
- Checks that the Invoice Date is a valid date.
- Checks the Customer Name.
- Checks the Account Number.
- Checks the Invoice Total.

The following test elements are created during this exercise:

- Script named *Invoice Customer*.
- Form check named *Car Sold*.
- Text check named *Printed Invoice Text Check*.

## Learning the Script

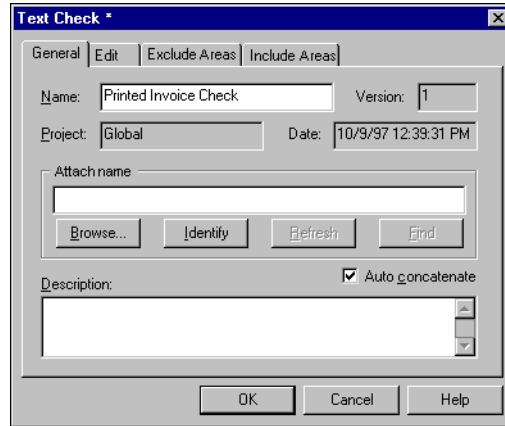
As you Learn this test script, you will insert a text check that will eventually include a number of different types of include areas. These areas will be described as you reach the related portion of this exercise. Use the following procedure to create your test script:

1. Start *QARun* and open a new script.
2. Start *QADemo Version 2* in the normal manner (see page 3-17 if necessary).
3. Sign on to the system using the following information:
  - a. Enter **CW** in the User ID field.
  - b. Enter **PASS** in the Password field.
  - c. Select the London radio button as the site location.
4. Click **OK**. The New dialog box displays.
5. Select Customer Invoice from the list and click **OK**. The Customer Invoice dialog box displays.
6. Press the **Learn Hotkey**, **{Alt {F10}}**, to begin learning the script.
7. Click the **Find** button from the Customer Invoice dialog box to locate an existing customer.
8. Click **OK** on the Find Customer dialog box to select the customer.
9. Click the **Sell Car** button from the Customer Invoice dialog box.
10. Select a car from the Select Car to be Sold dialog box and click **OK**.
11. When the Car Sold confirmation box displays, press the **Insert Check Hotkey**, **{Alt {F8}}**. The Browse Checks dialog box displays.
12. Click the **New** button. The New Check dialog box displays.
13. Select the Form check radio button and click **OK**.
14. Define a form check that identifies the confirmation dialog box and name it *Car Sold*. Click **OK** to save the check and return to *QADemo*.
15. Click **OK** on the confirmation dialog box to return to the Customer Invoice dialog box.
16. Click the **Print** button on the Customer Invoice dialog box. The Print Preview window displays.
17. Press the **Insert Check Hotkey**, **{Alt {F8}}**, to create a text check for this window.

## Creating a Text Check

You are now at the point where you will begin defining the text check. Use the following procedure to create a text check:

1. Click the **New** button from the Browse Checks dialog box. The New Check dialog box displays.
2. Select the Text radio button and click **OK**. The Text Check dialog box displays:



3. Enter a name, such as *Printed Invoice Text Check*, and click the **Identify** button.
4. Position the pointer over the Print Preview window's title bar and click once.  
The text is captured and displayed in a Text Captured window. You can size and position this window however you wish.
5. When the Text Check dialog box reappears, click the Include Areas tab.

## Including and Excluding Areas

After you have identified the window to be checked, it is possible to mask out (exclude) areas of the screen (so that they are not included in the check) as well as to select (include) specific areas in order to perform checks on data contained in the included areas.

- **Include:** Marks out rectangular areas of the screen to perform specific checks on. The type of checks available are Date, Number, Time, and ASCII. When this option is used, all text outside of the rectangle is omitted from the check.
- **Exclude:** Marks out rectangular areas of the screen that will be excluded from the check. This is a way of ignoring values that are not significant or which may acceptably change — such as free disk space, the date, etc. When used, all text outside the rectangle is checked, all text within the rectangle is omitted from the check.

You may define multiple include or exclude areas on any window; however, you may not define include and exclude areas in the same check.

If you define multiple include or exclude areas, the Area Number spin control allows you to move around each of the areas and re-displays the contents of the rectangle. If the defined areas are includes, then you may edit the text within the Text In Area section of the dialog box. If the defined areas are excludes, you may not edit the text.

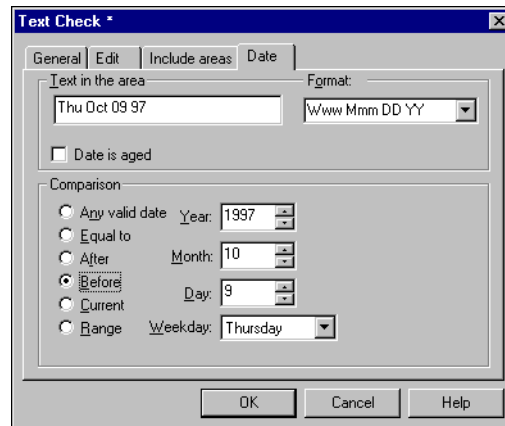
The text captured by Identify is displayed in the Text Captured window.

1. From the Text Check dialog box's Include Areas tab, click the **New** button and position the cursor over the Text Captured window. The cursor changes to a fine cross.
2. Move the mouse cursor to the top-left corner of the date field value. Click-and-hold the left mouse button and drag the cursor to the bottom-right corner of the date (ensure not to include the time information in the include area).
3. Release the mouse button. The area you just selected should appear surrounded by a green box and the Text Check dialog box regains focus.

## Date Checking

Use the following procedure to create a date check:

1. Click the **Date** button on the Text Check dialog box's Include Areas tab. The following information displays:



You may define your own date format by entering it in the Format area on the dialog box or you may use supplied values. Valid date combinations can be made using the following:

<b>DD</b>	Numeric day of the week (e.g. 01 - 31).
<b>Www</b>	Three-character day of week (e.g. Mon, Tue, Wed).
<b>Weekday</b>	Full character name of day of week (Monday - Sunday).
<b>MM</b>	Numeric month value (01 -12).
<b>Mmm</b>	Three-character value for month (Jan - Dec).
<b>Month</b>	Full character name of the month (January - December).
<b>YY</b>	Two-figure numeric value for the year (94, 95, 96, etc.).
<b>YYYY</b>	Four-figure numeric value of year (1994, 1995, 1996, etc.).

You may also define your own separator characters (such as “/”, “-”, or spaces).

After a date format is established, you can verify the date displayed in the included area as: equal to the date defined in the check, the current day’s date, before or after the date defined in the check, between that date and a second date, or simply any valid date.

2. Enter the following format into the list:

Www Mmm DD YY

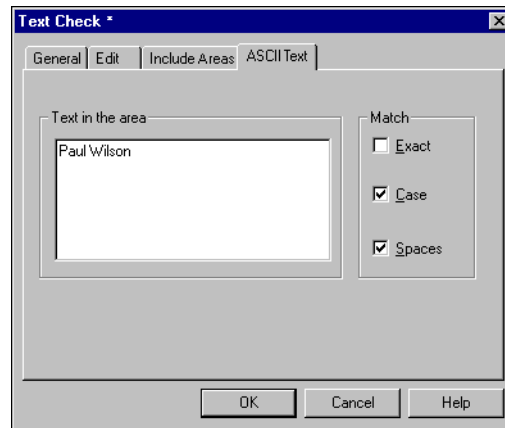
3. Select the Before radio button. This indicates that we are looking for a date before the current date, and, consequently, the check will fail.

You have now successfully defined the area and format for the text check’s date area. Next, you’ll create a text check on an ASCII include area of the same text check.

## ASCII Checking

Use the following procedure to define an ASCII include area for the same text check:

1. Click the Include Areas tab and click the **New** button. This allows you to define another include area for the same text check.
2. On the Text Captured window, highlight the customer name “Paul Wilson.”
3. Return to the Text Check dialog box, and click the **ASCII** button. The following information displays on the ASCII Text tab:



The Match options allow you to determine how the check will handle the text. For example, if the check definition (expected response) was "a b" (a followed by 2 spaces, followed by b), then:

- **Exact:** Forces an exact match of the string. If not selected, an actual response of "123a bXYZ" would pass because the required string is found. If selected, "123a bXYZ" would fail because "a b" is not the same as "123a bXYZ".
- **Space:** Checks exact spacing between characters. If not selected, an actual response of "a b" (a followed by four spaces followed by b) would pass. If selected, a response of "a b" would fail.
- **Case:** Checks for upper and lower case characters. If not selected, an actual response of "A B" would pass. If selected, "A B" would fail.

#### 4. Leave the settings at the default for now.

You have now successfully completed the ASCII check. Next, you'll create a pattern text check for a new include area for the same text check.

## Pattern Checking

This type of check focuses on the alphanumeric format of a field of text (such as the Zip/Post Code field in any database). Pattern checks make it possible to ignore the actual value of the field and check only the composition of the text displayed. For example, the Account Number in QADemo has three characters and a hyphen followed by four numbers. If the script was configured to create invoices for all customers on the database, the pattern check would still pass regardless of the account number shown.

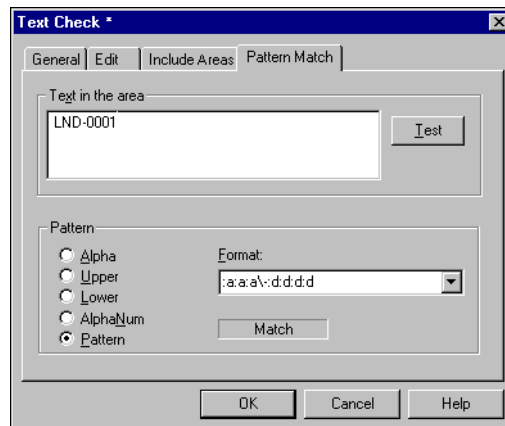
You can use the check to verify that test characters are all upper or lower case, are alphanumeric, or conform to a particular pattern. Pattern checking can ignore the actual value of each character within the string, but still check that it belongs to a class of characters. The classes can be numeric, alphanumeric, or just alpha.

Pattern checks have many valid operators. For a complete list of possible operators, press the **F1** key from the Pattern Match dialog box to receive the online help. The operators that you'll be using for this pattern match are:

- :a** Any alpha character; a-z, A-Z. Starting the pattern definition with a colon “:”, such as :a, indicates that the string starts with any alphabetic character. Omitting the colon indicates that the string starts with the character “a.” Therefore, the pattern :a:a:a, any three alpha characters, is not the same as a:a:a, which is the character "a" followed by any two alpha characters. Note that the check passes if a matching pattern is found anywhere within the included area. For example, a check for three alpha characters (:a:a:a) passes if the included area contains 12abc3. It fails if the area contains 12ab34.
- :d** Any digit or numeric value; 0-9. The check passes if a matching pattern is found anywhere within the included area.
- :** Matches any spaces, tabs and other control characters. For example, a check for :a::a will pass on abc, ab c and ab<Tab>c.
- \** The backslash is used to denote characters, such as \$, =, - etc. that are otherwise used as part of the pattern definition.

Use the following procedure to define a pattern check:

1. Click the Include Areas tab and click the **New** button. This allows you to define another include area.
2. From the Text Captured window, highlight the Account Number field's value, “LND-0001”.
3. Return to the Text Check dialog box and click the **Pattern** button. The following information displays on the Pattern Match tab:



4. Select the Pattern radio button in the Pattern area and enter the following text into the Format field:

```
:a:a:a\ - :d:d:d:d
```

The backslash allows for the hyphen to be physically included in the check.

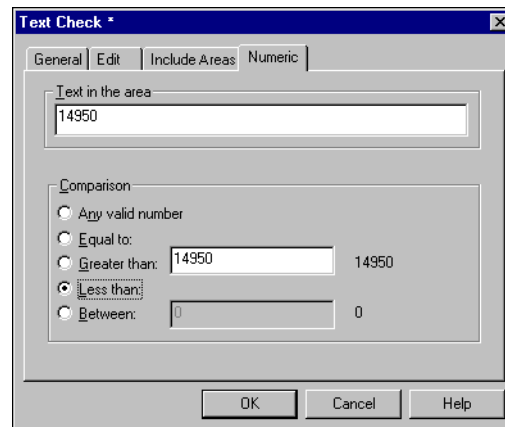
5. Click **Test** button to test and set the pattern.

You have now successfully created the pattern match check. Next, you'll create a numeric text check for a new include area for the same text check.

## Numeric Checking

Use the following procedure to create a number check:

1. Click the Include Areas tab on the Text Check dialog box and click the **New** button. This allows you to define another include area on the same text check.
2. From the Text Captured window, highlight the Total field's value, "14950.00".
3. Return to the Text Check dialog box, and click the **Numeric** button. The following information displays on the Numeric tab:



You can define the check as an exact match, greater or less than the defined value, between the defined value and a second value, or simply any numeric value.

4. Select the Less than radio button. Again, this will make the check fail.
5. Click **OK** to save the completed text check.
6. When QADemo reappears, click **Cancel** to return to the Customer Invoice screen.
7. Press the **Learn Hotkey**, {Alt {F10}}, to stop learning.
8. From the File menu, choose **Save** and save the script as *Invoice Customer*.

## Exercise Summary

After completing this exercise, you should be able to successfully:

- Select areas to be included or excluded in a text check.
- Insert a date text check into your script.
- Insert an ASCII text check into your script.
- Insert a pattern text check into your script.
- Understand the valid pattern check operators.
- Insert a numeric text check into your script.

After completing this exercise, you are ready to begin to learn how to use the test scripts and the driver script together — to complete your test suite. The exercises contained in “Using Driver and Test Scripts Together” describe these concepts.

## Chapter 4. Using Driver and Test Scripts Together

This section shows you how to use the test scripts and the driver script together — to complete your test suite. You will also modify the driver script so that it runs QADemo Version 3. You can then use the log table to analyze any differences between versions.

If you remember, during “Exercise 1 — Creating a Driver Script”, you created a driver script to run the target application, QADemo. This script invoked every item in the New dialog box’s main options list. In subsequent exercises, you created test scripts that performed specific checks on the system and performed some volume data entry. This exercise combines the functionality of the driver and test scripts to build a complete test system.

This chapter contains the following exercises:

- “Exercise 8 — Using the Run Command” introduces the Run command. The Run command allows a script to be called by another script as a separate process, and it’s the key to integrating individual test scripts into a driver script.
- “Exercise 9 — Inserting Script Dialog Boxes” explains how to build a dialog that asks for an ID and password at runtime so that it is not necessary to hard-code that information into the script. It will also ask which version of QADemo to run. The information entered into the dialog will be transferred to user-defined variables within the script and replace the hard-coded information.

---

## Exercise 8 — Using the Run Command

This exercise introduces the Run command. The Run command allows a script to be called by another script as a separate process. When this command is encountered, the current script (the calling script) is suspended, and the called script (“scriptname”) is run. When it finishes executing, it is unloaded from memory and control returns to the calling script. Execution of the calling script resumes on the line following the Run command.

The Run command’s syntax is as follows:

```
Run "Scriptname"
```

The Run command allows you to divide large systems into smaller, more manageable units. A driver script controls the overall flow of the process by calling other scripts in sequence.

### Getting Started

**Exercise Prerequisites:** Before beginning this exercise, you must have completed the following prerequisites:

- Closed any open copies of *QARun*.
- Closed any open copies of *QADemo*.
- Access to the script named *Driver1* created in “Exercise 1 — Creating a Driver Script”.
- Access to the script named *Exercise 2* created in “Exercise 2 — Creating List, Window, and Menu Checks”.
- Access to the script named *Exercise 3* created in “Exercise 3 — Creating Form Checks”.
- Access to the script named *Performance Test* created in “Exercise 4 — Creating Clock Checks To Test Performance”.
- Access to the script named *Adding New Customers* created in “Exercise 6 — Creating Clock Checks for External Data Entry”.
- Access to the script named *Invoice Customer* created in “Exercise 7 — Creating Text Checks”.

**Testing Requirements:** The requirements for this test script are that it:

- Modify the driver script to include individual test scripts.

The following test elements are created during this exercise:

- Modified script named *Driver1*.

## Modifying the Driver Script

Use the following procedure to modify the driver script to include individual test scripts:

1. Start *QARun* and load the *Driver1* script you created in “Exercise 1 — Creating a Driver Script” into the Script Editor.

Each of the test scripts that you created starts at a particular point within the target application. The first test script, *Exercise 2*, tests the initial screen when the Address Book is started.

When the driver script invokes QADemo from the **Start** button, the following code was generated:

```
Attach "PopupWindow"
    Button "Start", 'Left SingleClick'
    PopupMenuSelect "Run..."
Attach "Run PopupWindow"
    ComboText "&Open:",
        "C:\Program Files\Compuware\QARun\Demos\QADemo.exe" /v1
    Button "OK", 'Left SingleClick'
```

2. In order to run QADemo Version 3, modify the line:

```
ComboText "&Open:",
    "C:\Program Files\Compuware\QARun\Demos\QADemo.exe" /v1
```

So that it reads:

```
ComboText "&Open:",
    "C:\Program Files\Compuware\QARun\Demos\QADemo.exe" /v3
```

Where the Driver Script attaches to QADemo and logs on, the following code was generated:

```
Attach "Enter Sign on Details PopupWindow"
    EditText "&User Id:", "CW"
    EditText "Pa&ssword:", "Pass"
    RadioButton "London", 'Left SingleClick'
    Button "OK", 'Left SingleClick'
```

3. Following this block of code, add:

```
Run "Exercise 2"
```

This syntax relinquishes control to the first test script, which performs the window, list, and menu checks defined during the exercise.

Next, locate the section in the script where the driver script selects the Inquire option. The code should appear as follows:

```
Attach "New PopupWindow"
    ListBox "&New ", "Inquire", 'Left SingleClick'
    Button "OK", 'Left SingleClick'
```

4. Following this block of code, add:

```
Run "Exercise 3"  
Run "Performance Test"
```

Now, locate the code where the driver script invokes the Invoice Screen. It should appear as follows:

```
Attach "New PopupWindow"  
    ListBox "&New ", "Customer Invoice", 'Left SingleClick'  
    Button "OK", 'Left SingleClick'
```

5. Following this block of code, add:

```
Run "Adding New Customers"  
Run "Invoice Customer"
```

## The Resulting Script

The following example shows how the modified driver script should appear. The modified and added code appears in bold typeface.

```
Function Main  
Attach "PopupWindow"  
    Button "Start", 'Left SingleClick'  
    PopupMenuSelect "Run..."  
Attach "Run PopupWindow"  
    ComboBox "&Open:",  
        "C:\Program Files\Compuware\QARun\Demos\QADemo.exe" /v3  
    Button "OK", 'Left SingleClick'  
Attach "Enter Sign on Details PopupWindow"  
    EditText "&User Id:", "DTL"  
    EditText "Pa&ssword:", "Pass"  
    RadioButton "London", 'Left SingleClick'  
    Button "OK", 'Left SingleClick'  
Run "Exercise 2"  
Attach "New PopupWindow"  
    ListBox "&New ", "Add Car", 'Left SingleClick'  
    Button "OK", 'Left SingleClick'  
Attach "Add Car ChildWindow~1"  
    Button "Close", 'Left SingleClick'  
Attach "QADemo MainWindow"  
    MenuSelect "File~Main Options"  
Attach "New PopupWindow"  
    ListBox "&New ", "Inquire", 'Left SingleClick'  
    Button "OK", 'Left SingleClick'  
Run "Exercise 3"  
Run "Performance Test"
```

```

Attach "Inquire ChildWindow~1"
    Button "Close", 'Left SingleClick'
Attach "QADemo MainWindow"
    MenuSelect "File~Main Options"
Attach "New PopupWindow"
    ListBox "&New ", "Book Car In", 'Left SingleClick'
    Button "OK", 'Left SingleClick'
Attach "Book Car In ChildWindow~1"
    Button "Close", 'Left SingleClick'
Attach "QADemo MainWindow"
    MenuSelect "File~Main Options"
Attach "New PopupWindow"
    ListBox "&New ", "Transfer Car", 'Left SingleClick'
    Button "OK", 'Left SingleClick'
Attach "Transfer Car ChildWindow"
    Button "Close", 'Left SingleClick'
Attach "QADemo MainWindow"
    MenuSelect "File~Main Options"
Attach "New PopupWindow"
    ListBox "&New ", "Customer Invoice", 'Left SingleClick'
    Button "OK", 'Left SingleClick'
Run "Adding New Customers"
Run "Invoice Customer"
Attach "Customer Invoice ChildWindow"
    Button "Close", 'Left SingleClick'
Attach "QADemo MainWindow"
    MenuSelect "File~Exit"
End Function

```

## Running the Complete Script

In the previous exercises you ran test scripts against the target application. This script will now run QADemo Version 3 and report any problems it finds.

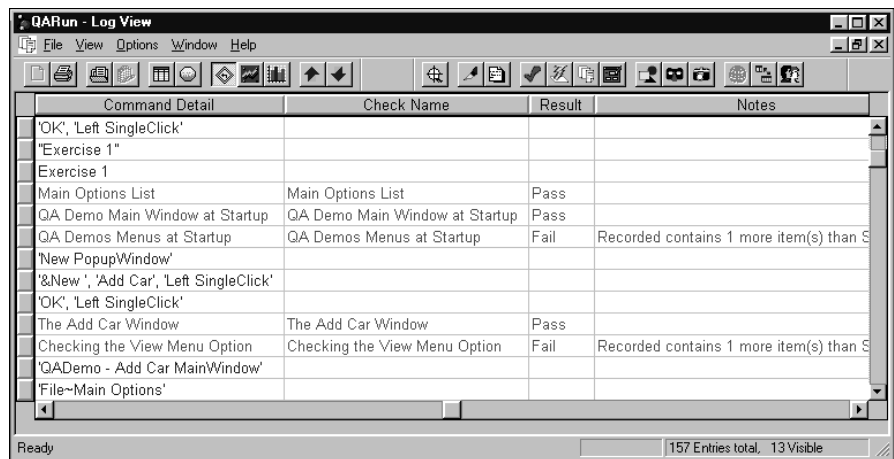
Use the following procedure to run the script:

1. Ensure that any active versions of QADemo are closed and that the test script is accessible.
2. From *QARun*'s **Run** menu, choose **Run Script**.
3. Select the default run environment and click **OK**.

## Analyzing the Results

Once the script runs, it is possible to view the resulting log file. Use the following procedure to view the log file:

1. From the **File** menu, choose **Browse**.
2. Select **Log View** from the Browse dialog box list and click **OK**.
3. Double-click on the *Driver1* file name (the one with the highest run number) in the Browse Logs display. The log file opens and should look like the following example:



Command Detail	Check Name	Result	Notes
'OK', 'Left SingleClick'			
"Exercise 1"			
Exercise 1			
Main Options List	Main Options List	Pass	
QA Demo Main Window at Startup	QA Demo Main Window at Startup	Pass	
QA Demos Menus at Startup	QA Demos Menus at Startup	Fail	Recorded contains 1 more item(s) than S
'New PopupWindow'			
'&New ', 'Add Car', 'Left SingleClick'			
'OK', 'Left SingleClick'			
The Add Car Window	The Add Car Window	Pass	
Checking the View Menu Option	Checking the View Menu Option	Fail	Recorded contains 1 more item(s) than S
'QADemo - Add Car MainWindow'			
'File~Main Options'			



**Note**

You may have to use the scroll bars to display the specific check results.

## Viewing Failed Checks

Clock checks record the *actual* and *expected* time results in the log file and can be viewed by displaying the Notes column.



**Hint**

If the Notes column is not displayed, from the **View** menu, choose **Column Layout>Grid** and add the Notes column to the display.

All other failed check details can be viewed by double-clicking the check name in the Check Name column.

It is easier to analyze failed checks by applying a filter to the log file. To do this:

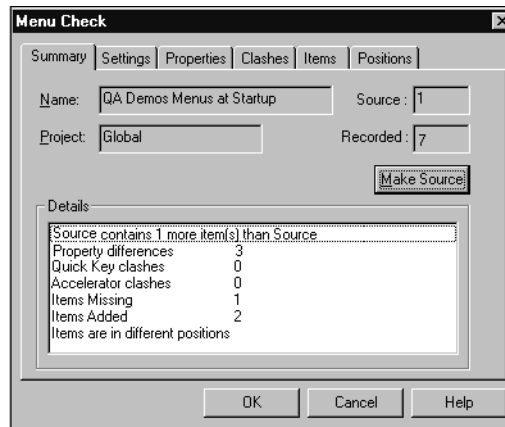
1. From the **View** menu, choose **Select Filter**.
2. Select Passed and Failed Checks from the list and click **OK**.

Certain checks failed against QADemo Version 3 (mostly performance, menus and text checks). All result dialogs display information on the failed check in a similar format. Here is a sample of these dialogs for the checks that failed during this exercise.

## Failed Menu Checks

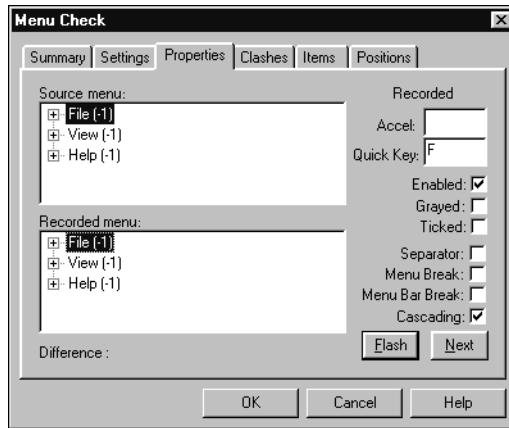
Use the following procedure to view information regarding the script's failed menu checks:

1. Double-click the first failed menu check *QADemo's menus at Startup*. The Menu Check dialog box displays:

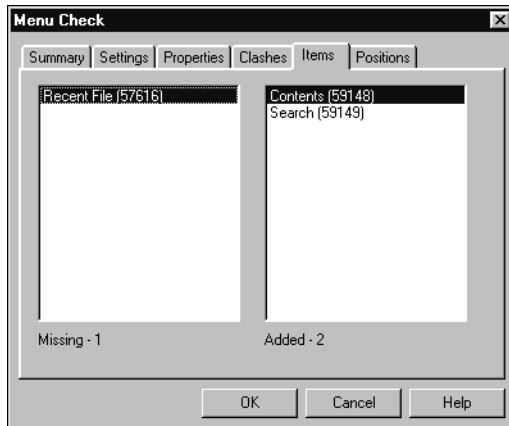


The Details area provides an overall description of why the check failed. In this instance, it failed because there are three property differences: 1 item is missing, 2 items have been added, and some menu items are in different positions.

2. Click the Properties tab. The dialog box now appears as follows:

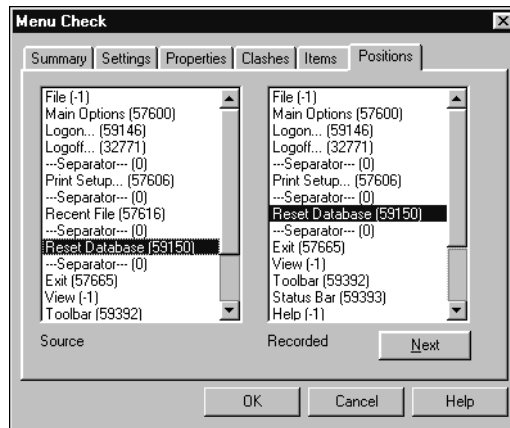


3. Click **Next**. Both the Source and the Recorded tree views will expand, and the right side of the dialog box flashes between the Source data and the Recorded data, showing the differences between the two highlighted menu options.
4. Click **Next** again to proceed to the next difference. After you have viewed all differences, select the Items tab. The dialog box changes to the following:



This dialog box shows which menu items have been added and removed from the previous version.

5. Select the Positions tab. The dialog box changes to the following:

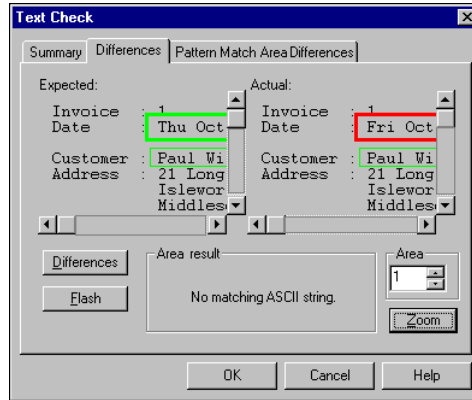


6. Click **Next** to highlight the menu options that have been moved.
7. Click **OK** when you are done and to return to the log file.

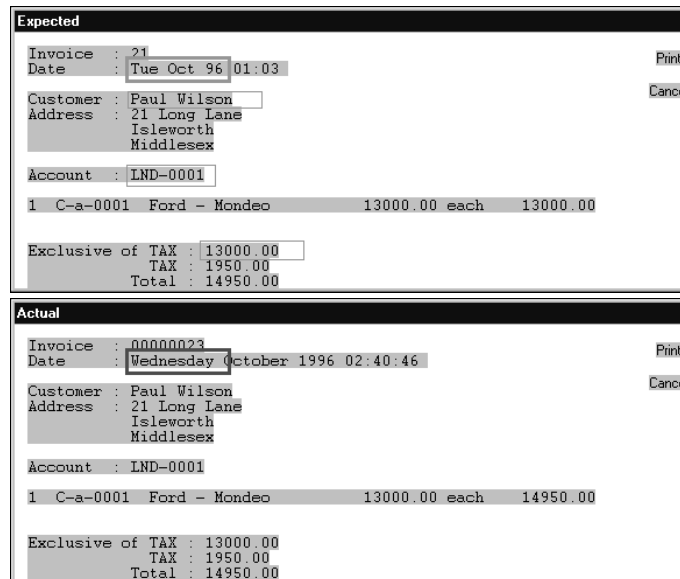
## Failed Text Checks

The check on the printed invoice screen has also failed (don't panic, it was designed to).

1. Double-click the *Printed Invoice Check* Check Name column.
2. Click the Differences tab. The following dialog box appears:



3. Click **Zoom**. Two windows appear on the screen: one for the Expected results and one for the Actual results.



- The areas that passed are displayed in green and the ones that failed are in red.
- Double-clicking on a failed area populates the dialog box with the statistics. The numeric field failed because it checked for a value less than it actually is.

4. Click **OK** on the Text Check dialog box to close the differences windows and return to Log View.

## Updating Checks

You can *update* a check to make the actual information that is encountered during replay the expected version the next time the check is run. This allows you to update your checks without having to manually redefine all the information. All failed checks, with the exception of clock checks, can be updated directly from the log file. Use the following procedure to update the failed checks:

1. From the log file, right-click the failed check name in the Check Name column.
2. Select **Make Source** from the pop-up menu.

## Exercise Summary

After completing this exercise, you should be able to successfully:

- Use the driver script to call individual test scripts.
- Use the Run command from within scripts.
- Use the driver script to run a different version of the target application.
- View and update failed checks.

---

## Exercise 9 — Inserting Script Dialog Boxes

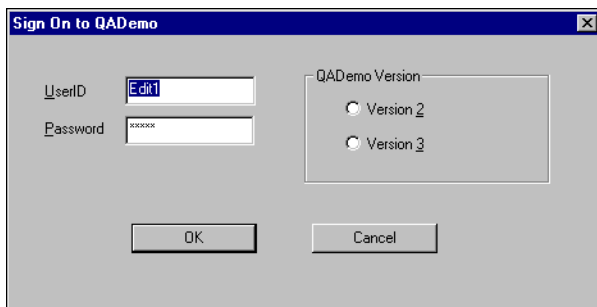
*QARun* contains a Dialog Editor that can be accessed from the Script Editor. The purpose of the Dialog Editor is to allow you to design dialogs that will be displayed when you run your scripts. Dialogs allow you to enter variable or confidential data into *QARun* as it is running, rather than hard-coding it into a script.

Dialogs are created graphically using the tools provided by the Dialog Editor. These graphical representations can then be inserted into the Script Editor as a function. The Dialog function can be called by the script at any time and can be used by any number of scripts. If you need to change the dialog's definition, you can load the image into the Dialog Editor to make the modifications. After you make the changes, all scripts that call the dialog will automatically call the modified version. There is no need to change the function definition within the script (unless you must add logic to process new controls).

Information entered into the dialogs — from pushing a button to entering passwords — can then be processed by the script at runtime. The Dialog Editor offers all of the standard controls for designing Windows-compliant dialogs.

## Getting Started

The driver script built during “Exercise 1 — Creating a Driver Script” logs on to QADemo and drives the application. The UserID and Password are hard-coded into this script. The following exercise explains how to build a dialog that asks for an ID and password at runtime. It will also ask which version of QADemo to run. The information entered into the dialog will be transferred to user-defined variables within the script and replace the hard-coded information. After completing this exercise, the resulting dialog should resemble the example below:



**Figure 4-1.** User-Defined Script Dialog Box

**Exercise Prerequisites:** Before beginning this exercise, you must have completed the following prerequisites:

- Access to the modified script named *Driver1* created in “Exercise 8 — Using the Run Command”).
- Closed QADemo.
- Closed QARun.

**Testing Requirements:** The following test elements are created during this exercise:

- Dialog named *Start Script Dialog*.

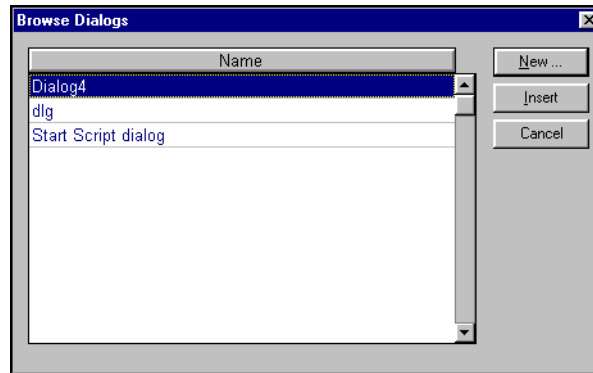
You should allow 20–30 minutes to complete this exercise.

## Creating the Dialog

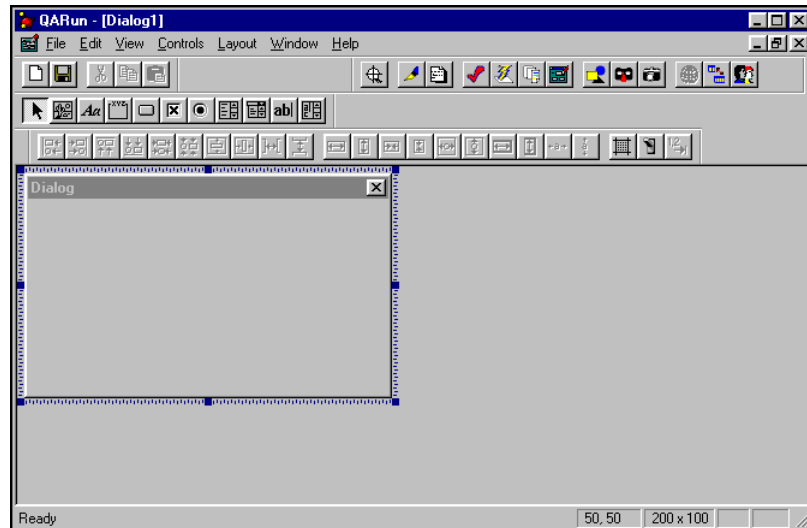
Use the following procedure to create a new dialog from the Dialog Editor. For this exercise:

1. Start QARun and load the *Driver1* script.
2. Position the cursor on a blank line under the `Function Main` command.
3. Click the **Dialogs** Button from the Script Editor’s toolbar to access the Browse Dialogs table. The Browse Dialogs dialog box displays.



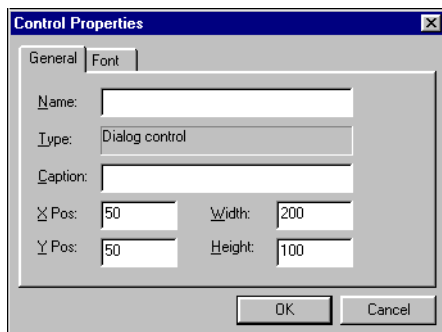


4. Click the **New** button to define a new dialog. The Dialog Editor displays with a blank dialog definition as shown below:



The Dialog Editor window contains two additional toolbars to assist you with creating the dialog boxes. The window also displays a blank dialog box with active control handles.

5. Double-click on the blank dialog to define its properties. The Control Properties dialog box displays:



6. Change the dialog name to *Sign on dialog*. This name will be used within the script by the function definition.
7. Next change the Caption to *Sign On to QADemo*. This text will appear in the title bar of the dialog.
8. Click **OK**.
9. Using Figure 4-1 on page 4-12 as a visual guide, resize the dialog by clicking on the right-bottom corner handle and sizing the dialog.

## Adding Controls

Adding controls to the dialog definition is performed on a point-and-click principal. There are two extra tool bars in the Dialog Editor: the Controls toolbar and the Layout toolbar. Use the Controls toolbar (Figure 4-2) to select the type of control needed.



Figure 4-2. Dialog Editor Controls Toolbar

### Adding Static Text Controls

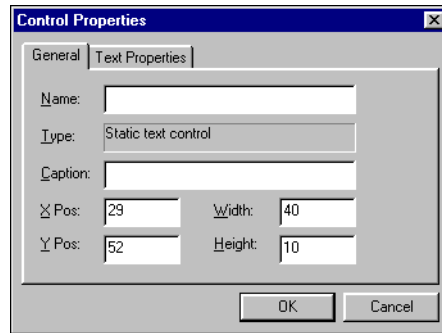
Static text controls are used to provide labels for edit controls, list boxes, and combo boxes which, unlike radio buttons and check boxes, do not have a text caption of their own. Use the following procedure to add static text controls:



1. Click once on the **Text Control** button to select it.
2. Move the cursor to the dialog. The cursor changes to the cross hairline cursor.
3. Click once on the left side of the dialog to drop the control.

The control can be moved around the dialog to the preferred location. It can also be resized using the grab handles.

4. Double-click on the text control to invoke the Control Properties dialog box:



The Name field contains the variable name used to process the contents of the field and can be changed to a more meaningful name to make referencing it easier.

5. Change the Caption field so that it says *&User ID:*. The ampersand (&) before the U in User denotes this as an underlined character.
6. Click the Text Properties tab.
7. Ensure that the Border and No Prefix options are not selected.

The Border option puts a border around the text, and the No Prefix option switches off the underlined character option.

8. Click **OK**.
9. Repeat the above steps to create another text control with the caption *&Password:*.

## Aligning the Controls

The Layout toolbar allows you to format and align controls once they have been placed on the dialog. Use the Layout toolbar (Figure 4-3) to select the type of control needed.



**Figure 4-3.** Dialog Editor Layout Toolbar

Use the following procedure to align two text controls:

1. Highlight both text controls by clicking once on the first control, holding down the **Shift** key, and clicking on the second text control.

The second control will have the bolder of the two highlights. This means that this control is primary, and the alignment modification will be made according to this control's position.

2. Click the **Align Left** button to align the controls to the extreme left of the master control.



## Adding Edit Controls

Edit controls are used to accept user input which can then be processed by the script at runtime. Edit controls are useful for inputting User IDs, passwords, and other variable information

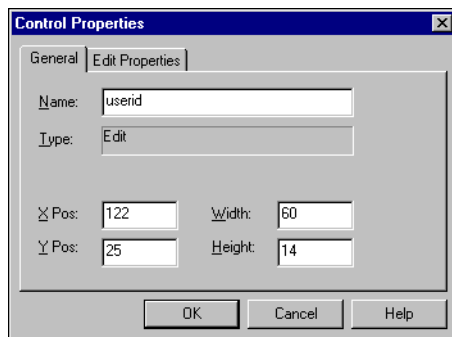
Use the following procedure to insert an edit control.



1. Click once on the **Edit Control** button to select it.
2. Move the cursor over to the dialog. The cursor changes to the cross hairline cursor.
3. Click once on the right side of the User ID static text control to drop the control.

The control can be moved around the dialog to the preferred location. It can also be resized using the grab handles.

4. Double-click on the edit control to display the Control Properties dialog box:



Unlike the static text control, the Name of this control will be used by the script at runtime and should be changed to something more memorable.

5. Change the Name field to *userid*. Remember, variable names are case-sensitive.
6. Click the Edit Properties tab. Ensure that the Visible and the Tabstop options are selected. The Tabstop option allows the user to move to this control using the **TAB** key instead of the mouse.
7. Click **OK**.
8. Repeat steps 1–7 above to create a second edit control for the Password option.
  - a. Change the Name field to *password*.
  - b. Click the Edit Properties tab and ensure that the Visible, Tabstop, and Password check boxes are all selected.

The Password option displays the contents of the control as a series of asterisks rather than visible text.

- c. Click **OK**.

- Align the controls in the same way as the static text controls (see “Aligning the Controls” on page 4-15).

## Adding a Group Box

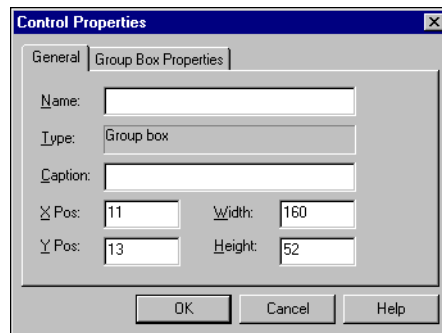
Group Boxes are used to provide a visual grouping of related controls. Use the following procedure to add a group box to the dialog:



- Click the **Add Group Box** button on the toolbar and click on the right side of the dialog definition.

A rectangle with the words “Group Box” displays on the dialog. You can move the box by placing the mouse pointer inside it, holding down the right mouse button, and dragging. In each corner of the box is a size handle. Clicking and dragging these handles will change the size of the box.

- Size the group box so that it can contain two radio buttons and their labels. Don't worry if you get the size of the box wrong, it can always be resized later.
- Double-click on the group box control to invoke the Control Properties dialog box:



As with Static Text controls, the Name field can be ignored as it will not be used to return a value at runtime.




---

Ensure that you've double-clicked the Group Box and the Group Box Control Properties dialog box displays. It's easy to accidentally select the dialog box's Control Properties.

---

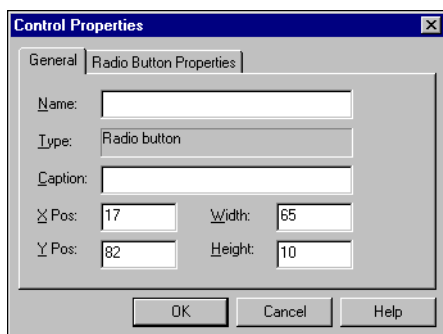
- Change the Caption field to *QADemo Version*. This changes the caption at the top of the rectangle.
- Click **OK**.

## Adding Radio Buttons

A radio button is a user option that is exclusive, i.e., only one option per group may be selected. Use the following procedure to add radio buttons to the dialog:



1. Click the **Radio Button Control** button on the toolbar.
2. Move the mouse pointer into the group box and click again to drop the radio button.  
A radio button permits only one of a group of related items to be selected. Selecting any one will de-select all the others in the group.
3. Double-click on the radio button to invoke the Control Properties dialog box:



The Name of this control will be used by the script at runtime and should be changed to something more memorable.

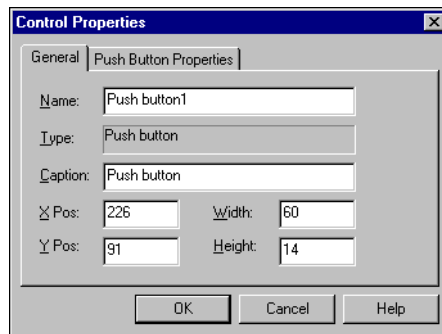
4. Change the Name field to *version2*. Remember, variable names are case-sensitive.
5. Change the Caption field to read *Version &2*. The ampersand indicates which character has the underscore.
6. Click the Radio Button Properties tab and ensure that Visible, Tabstop, and Auto options are selected. The Auto option ensures that the radio buttons display a black dot when selected.
7. Add another radio button. Assign it the name *version3* and assign it the caption *Version &3*.
8. Click **OK**.
9. Align the controls in the same way as the static text controls (see “Aligning the Controls” on page 4-15).

## Adding Push Buttons

A push button, also known as a command button, is generally used by the user to dismiss the dialog, e.g., **OK**, **Cancel**, etc. There are exceptions, such as **Help**. Use the following procedure to add two push buttons to the dialog.



1. Select the **Push Button** button from the toolbar.
2. Click the mouse pointer to drop the push button under the other controls, towards the bottom of the dialog.
3. A button with the word **Push Button** should appear on the dialog. This control can be sized using the size handles.
4. Double-click on the control to invoke the Control Properties dialog box:



Again, the name of this control will be used by the script at runtime and should be changed to something more memorable. Change the value here to *okbutton*. Remember, variable names are case-sensitive.

5. Change the Caption to *OK*.
6. Click the Push Button Properties tab and ensure that the Visible, Tabstop, and Default options are selected. The Default option indicates that this button will be automatically selected if the **Enter** key is pressed.
7. Add another push button and assign the name *cancelbutton* and a caption of *Cancel*.
8. Click **OK**.
9. Align the controls.
10. Save the dialog, giving it the name *Start Script Dialog*.
11. From the **File** menu, choose **Close** to close the Dialog Editor.

The Dialog function definition will be inserted into the active script at the cursor location.

## The Resulting Script

The following is the Dialog function definition that is pasted into the Script Editor in script format:

```
var StartScriptDialog_Controls[]
// StartScriptDialog_Controls[ "Static1" ]
// StartScriptDialog_Controls[ "Static2" ]
// StartScriptDialog_Controls[ "userid" ]
// StartScriptDialog_Controls[ "password" ]
// StartScriptDialog_Controls[ "Group box1" ]
// StartScriptDialog_Controls[ "version2" ]
// StartScriptDialog_Controls[ "version3" ]
// StartScriptDialog_Controls[ "okbutton" ]
// StartScriptDialog_Controls[ "cancelbutton" ]
dialog "Start Script Dialog", StartScriptDialog_Controls
```

Where:

`var StartScriptDialog_Controls[ ]` is a local array.

An array is a collection of related data values referred to by a single variable name, in this case `StartScriptDialog_Controls`. This array references all the controls on the dialog, known as elements, by name. The Dialog Editor inserts all of the array's elements as a series of comment lines for easy identification. The value of any control can be processed by referencing it within the `[ ]` square brackets. For example, to get the value entered in the User ID field:

```
name = StartScriptDialog_Controls["userid"]
```

Where:

`name` is the variable where the value contained in the element `userid` is placed.

The second part of the example,

```
dialog "Start Script Dialog", StartScriptDialog_Controls
```

executes the dialog at runtime. Double-clicking on the word `dialog` produces the graphical image of the dialog.

## Modifying the Driver Script

The section of the driver script that you need to modify is the part directly under the new dialog definition syntax where the script runs QADemo and logs on to the system:

```
Function Main
var StartScriptDialog_Controls[]
dialog "Start Script Dialog", StartScriptDialog_Controls
Attach "PopupWindow"
    Button "Start", 'Left SingleClick'
```

```

    PopupMenuSelect "Run..."
Attach "Run PopupWindow"
    ComboBox "&Open:", "C:\Program
    Files\Compuware\QARun\Demos\QADemo.exe" /v3
    Button "OK", 'Left SingleClick'
Attach "Enter Sign on Details PopupWindow"
    EditText "&User Id:", "CW"
    EditText "Pa&ssword:", "Pass"
    RadioButton "London", 'Left SingleClick'
    Button "OK", 'Left SingleClick'

```

You must modify this section to take advantage of the information entered into the dialog box. The necessary changes appear in bold typeface:

```

Function Main
var StartScriptdialog_Controls[]
dialog "Start Script dialog", StartScriptdialog_Controls
;Get the values of the userid and password fields
name = StartScriptDialog_Controls["userid"]
pswd = StartScriptDialog_Controls["password"]
;If user pushes Cancel kill the script
If cancelbutton = 1
    Stop
Endif
Attach "PopupWindow"
    Button "Start", 'Left SingleClick'
    PopupMenuSelect "Run..."
Attach "Run PopupWindow"
;Process the values of the Radio Buttons
;and run which ever version was selected
if StartScriptDialog_Controls["version2"] = 1
    ComboBox "&Open:", "C:\Program
    Files\Compuware\QARun\Demos\QADemo.exe" /v2
else
    ComboBox "&Open:", "C:\Program
    Files\Compuware\QARun\Demos\QADemo.exe" /v3
endif
    Button "OK", 'Left SingleClick'
;Enter the values of the userid and password fields
Attach "Enter Sign on Details PopupWindow"
    EditText "&User Id:", name
    EditText "Pa&ssword:", pswd
    RadioButton "London", 'Left SingleClick'
    Button "OK", 'Left SingleClick'

```

## Running the Script

In the previous exercises, the test scripts ran without requiring any user interaction. In this exercise, the script will prompt you to enter a user ID and password and then asks you to select a version of QADemo. The script will then run against the target application and report any problems it finds.

Use the following procedure to run the script:

1. Ensure that any active versions of QADemo are closed and that the test script is accessible.
2. From the *QARun* **Run** menu, choose **Run Script**.
3. Select the default run environment and click **OK**.

The script begins to run and then displays the user-defined Log On to System dialog box.

4. Enter your User ID and Password and select the Version 3 radio button. Click **OK** to continue the script execution.

After the scripts finishes running, you may analyze the results. The results should be the same as those described in “Analyzing the Results” on page 4-6.

## Exercise Summary

After completing this exercise, you should be able to successfully:

- Create a user-defined dialog.
- Add various controls to the dialog.
- Align dialog controls.
- Insert a completed dialog box definition into the script.

# Index

## A

ASCII pattern checks, 3-52  
attach statements, 2-9

## C

checks, ix, 3-3  
     clock checks, 3-26  
     filtering results, 3-34  
     form checks, 3-18  
     list check, 3-3  
     menu checks, 3-7  
     text checks, 3-50  
     updating, 4-11  
     viewing failed, 4-6  
         menu checks, 4-7  
         text checks, 4-10  
     window checks, 3-5  
 clock checks  
     as stopwatch, 3-29  
     creating, 3-26  
     viewing results of, 3-35  
 colors, changing, 2-3  
 configuration options  
     changing, 2-3  
 configuring  
     bitmapselects, 2-4  
     script editor, 2-3  
 controls  
     adding, 4-14  
     aligning, 4-15  
     edit controls, 4-16  
     static controls, 4-14  
 customer support, x

## D

data  
     using external, *see testdata files*  
 date checks, 3-51  
 default scripts  
     changing, 2-4  
 dialog editor  
     adding controls, 4-14  
     aligning controls, 4-15  
     creating dialogs, 4-12  
     group boxes, 4-17  
     in scripts, 4-20  
     push buttons, 4-19  
     radio buttons, 4-18  
 dialogs, *see dialog editor*  
 driver scripts, ix, 2-1  
     adding dialogs to, 4-20  
     creating, 2-2  
     overview, 2-1  
     using with test scripts, 2-1, 4-1

## E

edit controls  
     adding to dialogs, 4-16  
 events  
     defining, 3-22  
     waits, 3-28  
     window events, 3-45  
 exclude areas, 3-50  
 external data, *see testdata files*

## F

filters

- using with checks, 3-34

- fonts, changing, 2-3

- form checks

  - creating, 3-18

- FrontLine web site, ix

## G

- group boxes

  - adding to dialogs, 4-17

## I

- include areas, 3-50

- introduction, vii

## L

- learn, stopping, 2-7

- list checks

  - creating, 3-3

- logging on, 2-2

- logs

  - column layouts, 3-34

  - filters, 3-34

  - run numbers, 3-14

  - setting, 3-33

  - viewing, 3-14

## M

- menu checks

  - creating, 3-7

  - failed, 4-7

## N

- numeric checks, 3-55

## P

- passwords

  - logging on as admin, 2-2

- pattern checks, 3-53

- performance tests, *see clock checks*

- push buttons

  - adding to dialogs, 4-19

## Q

- QADemo

  - overview, 1-2

  - signing on to, 2-6

## R

- radio buttons

  - adding to dialog boxes, 4-18

- related publications, viii

- replaying scripts, 2-10

- run command, 4-2

- run environments

  - adding, 3-32

  - selecting, 3-12

- run numbers, 3-14

- running scripts, 4-2

## S

- script editor

  - configuring, 2-3

- scripts, ix

  - calling, 4-2

  - default script, 2-4

  - replaying, 2-10

  - reviewing contents of, 2-8

  - run environments, 3-12

  - stopping learn, 2-7

  - summary info, 3-12

- signing on, 2-2, 2-6

- static controls

  - adding to dialogs, 4-14

- summary info, 3-12

- support web site, ix

## T

- target applications, ix

  - adding external data into, 3-36

  - QADemo overview, 1-1

- terminology, ix

- test scripts, ix

  - building, 3-1

  - using with driver scripts, 4-1

- test sites, ix

  - overview, 2-1

- testdata files, 3-38

  - fields, 3-38

  - looping, 3-39

  - overview, 3-38

  - records, 3-38

  - wildcard characters, 3-39

## text checks

- ASCII, 3-52
- creating, 3-50
- dates, 3-51
- failed, 4-10
- including and excluding areas, 3-50
- numeric, 3-55
- pattern, 3-53

TypeToControl, 2-4

**U**

updating checks, 4-11

user-defined dialogs, *see dialog editor*

**W**

wait command, 3-28

web site for support, ix

## wildcards

- in testdata files, 3-39

## window checks

- creating, 3-5
- ignoring values in, 3-7

## window events

- creating, 3-45

world wide web, ix



# Tell Us What You Think

Compuware needs your feedback to help us meet our commitment to quality software and documentation. Please complete the following questions and return this Reader's Response Form to:

**Compuware Corporation**  
**SSD-P Quality Assurance Manager**  
**P.O. Box 9080**  
**Farmington Hills, MI 48333-9080**  
**Fax: 1-248-737-7339**

Outside the USA and Canada, please contact your local Compuware office or agent to return this form.

## Tell us about the *QARun* GUI Testing Getting Started Guide

Yes	No	Is this document easy to read and understand?
Yes	No	Is the organization clear and helpful?
Yes	No	Is the technical content accurate?

## Tell us about the *QARun* Documentation Set

Yes	No	Does <i>QARun</i> documentation and online Help contain all the information you need?
Hardcopy	Softcopy	What type of <i>QARun</i> documentation do you use?

## Tell us about *QARun*

Yes	No	Does the product work as documented?
Yes	No	Does the software contain all the functions you need?

If you answered No to any question, please provide an explanation or your suggestions in the space provided. If you need more space, use the back of this form.

---

---

---

## Please Tell Us About Yourself

Name

---

Company

---

Address

---

Phone

---

Operating System & Release

---

CWQUGSG4D

