QAD 5.1.0

Table Of Contents

Welcome to QADirector

*QA*Director is a test management solution designed to help testers, developers, and managers deliver thoroughly tested applications on time and on budget.

Testing is a major undertaking that requires the coordination of many steps, many testers, large suites of test scripts, and multiple versions of different applications. *QA*Director supplies a framework for managing the entire testing process—from design, to execution, to analysis.

Plan and execute tests, analyze tests and track defects -- all from a single point of control.

The *QA*Director online documentation has been designed to help you learn and use every aspect of the product.

This online help system has been organized as follows:

Administrating and Configuring	Look here for information on assigning user passwords and permissions, and maintaining the *QA*Director database.
Creating and Managing Projects	Look here for information on creating and editing projects, assigning users to specific projects, and specifying start dates and end dates for projects.
Managing Test Scripts	Look here for information on accessing test scripts, creating categories for the scripts, and copying them to the Test Library.
Designing and Organizing Tests	Look here for information on creating and managing tests and suites, as well as using the Test Library.
Executing Tests in the Suite	Look here for information on viewing scheduled jobs (tests), jobs in progress, and performing further test execution tasks.
Analyzing Tests	Look here for information on viewing job results, and analyzing and comparing them.
Tracking Defects	Look here for information on submitting defects and using *QA*Director in conjunction with TrackRecord to track failed scripts in a test application.

Getting Started

Starting QADirector

The *QA*Director administrator should configure a database, team environment, and user roles, permissions, and passwords before a user can open *QA*Director.

To start *QA*Director:

1. Open *QA*Director from the Windows **Start** menu. Click **Start>Program Files>Compuware>*QA*Director**.
2. When prompted, enter user name and password. User names and passwords are assigned by the *QA*Director administrator in the System Administration Center. Consult with the *QA*Director administrator to obtain an ID and password.

3. If necessary, select a data Source or server from the **Data Source or Server** list. If no data Source or server appear in the list follow these steps:

    a. Click the **Configure** button. The **Select a Database** dialog box appears. The information in this dialog box must be entered correctly.

    b. Select either the **SQL Server/MSDE** option or the **Oracle** option to select a data Source type.

    c. Select a data Source name from the **Data Source Name** list.

    d. Type the database name in the **Database Name** field. If using SQL Server, this should be Default.

    e. Type a schema or owner name in the **Schema/Owner Name** field. If using SQL Server, this should be Default.

    f. Type a valid user name in the **User Name** field.

    g. Type a valid password in the **Password** field.

    h. Click **Test** to ensure the connection works.

    i. Click **Ok** to save the connection information and return to the Login screen.

If a message appears stating that the Test Management Server is not running:

1. Continue logging in to *QA*Director.
2. Before running tests, start the Test Management Server.

Starting QADirector

The *QA*Director administrator should configure a database, team environment, and user roles, permissions, and passwords before a user can open *QA*Director.

To start *QA*Director:

1. Open *QA*Director from the Windows **Start** menu. Click **Start>Program Files>Compuware>*QA*Director**.
2. When prompted, enter user name and password. User names and passwords are assigned by the *QA*Director administrator in the System Administration Center. Consult with the *QA*Director administrator to obtain an ID and password.

3. If necessary, select a data Source or server from the **Data Source or Server** list. If no data Source or server appear in the list follow these steps:

a. Click the **Configure** button. The **Select a Database** dialog box appears. The information in this dialog box must be entered correctly.

b. Select either the **SQL Server/MSDE** option or the **Oracle** option to select a data Source type.

c. Select a data Source name from the **Data Source Name** list.

d. Type the database name in the **Database Name** field. If using SQL Server, this should be Default.

e. Type a schema or owner name in the **Schema/Owner Name** field. If using SQL Server, this should be Default.

f. Type a valid user name in the **User Name** field.

g. Type a valid password in the **Password** field.

h. Click **Test** to ensure the connection works.

i. Click **Ok** to save the connection information and return to the Login screen.

If a message appears stating that the Test Management Server is not running:

1. Continue logging in to *QA*Director.
2. Before running tests, start the Test Management Server.

About the Centers and Test Library

The *QA*Director interface lets you easily access the test management functions. *QA*Director's workspace is center-oriented and is designed for both novices and experienced users:

Center	Description
Defect Information Center	Use the **Defect Information Center** to view and edit defects associated with tests in the project. These defects are submitted through the **Job Information Center**.
Job Information Center	The *QA*Director **Job Information Center** lists all scheduled jobs, jobs in progress, and results. From within this center, perform job execution tasks. The Jobs panel is divided into two sections: result folders and the job related tabs.
Project Information Center	Use the **Project Information Center** to create or edit projects, assign users to specific projects, and specify dates for projects. Since the tasks performed in the **Project Information Center** are administrative, only users with Administrative, Team Lead, and QA Manager roles can access this center.
Suite Information Center	Use the **Suite Information Center** to create and manage test suites. Test suites are groups of tests applied to one application or one component of an application. Test suites create order and control by defining which tests are needed, grouping the tests logically, and designating the order in which the tests should be run.
System Administration Center	The *QA*Director **System Administration Center** enables designated *QA*Director administrators to perform a variety of tasks such as assigning user passwords and permissions and maintaining the *QA*Director

	database.
Script Information Center	Use the **Script Information Center** to access test scripts after they have been created in third party tools. Additionally, use the **Script Information Center** to create and organize user defined and manual tests. Create categories to group scripts together. These categories can be copied to the **Test Library** for easy access.
Test Library	The **Test Library** is a repository for all available tests within a project. It contains test classes, class templates, test procedures, and test scripts. The **Test Library** makes it possible to use these tests in different suites within a project. As tests are used across suites, changes are saved to the **Test Library** and all suites.

Setting User Options

*QA*Director allows you to set personal preferences that control how *QA*Director appears and behaves while you are using it. You can choose options that affect the appearance of the *QA*Director main window, default test permissions, and type of information displayed during test execution.

To set user options:

1. Click **Tools>User options.** The User Options Dialog Box appears.
2. Click the various tabs to view and set options.
3. Click **Apply** to apply the options and select another tab.
4. Click **OK** to save the options and exit the dialog box.

The User Interface

About the User Interface

*QA*Director's main window is the starting point for all test management activities. The Menu bar across the top of QADirector provides access to many of the functions and features necessary for testing.

The items on the Tool bar below the menu are enabled in various centers and also provide an efficient way to access these functions and features.

When clicking an icon from the Centers bar, the Center opens in the workspace. From this single toolbar, navigate from center to center without disrupting work. *QA*Director retains the session information and state in each center.

When opening *QA*Director for the first time, the Test Library will appear on the right side of the workspace. The Test Library contains classes, procedures, and scripts for each project.

Using the *QA*Director Main Window

*QA*Director's main window is the starting point for all test management activities. The window is divided as follows:

Centers Toolbar: This toolbar accesses all *QA*Director centers. From this single toolbar, navigate from center to center without disrupting workflow. This is accomplished not only from the convenience of the Centers toolbar, but also from *QA*Director retaining the session information and state in each center. For example, while working in the Project Information Center, click the Suite Information Center to perform another task, and then return to the Project Information Center. The Project Information Center opens to the last task performed.

Workspace: When clicking an icon from the Centers toolbar, the Center opens in the workspace. Most work is performed within the panels of each center in the workspace. Additionally, the workspace hosts both windows and web forms.

Test Library: The test library contains classes, procedures, and scripts for each project. The Test Library offers a way to reuse tests from suite to suite. To copy scripts, classes, or procedures from the Test Library into the Suite, drag the test from the library to the Suite in the Suite Information Center. There is only one Test Library per project.

When opening *QA*Director for the first time, the Test Library will appear on the right side of the workspace.

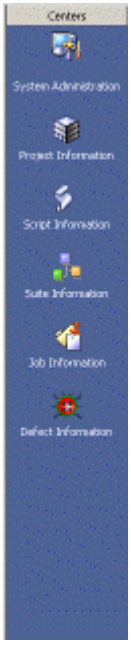
Panels: Panels provide at-a-glance information. Perform tasks from each panel's Actions menu or the right-click shortcut menu. Panels will vary depending upon the center.

Status Bar: The status bar displays menu and toolbar icon descriptions and the status of any process. It also displays current server or database, user, and role information.

Menus and Icons: Throughout *QA*Director's centers, there are Action menus and icons. These tools offer flexibility when accessing different functions. For example, access the Filters Manager dialog box from the Action menu of the Suite Center, Script Information Center, or Job Information Center panels. It is also possible to access the Filters Manager dialog box from the right-click menu on the Test Library, the Tools menu on the main menu bar, the filter icon on the toolbar, and the Filter icon on the Suite Information Center's Suite panel.

Using the Centers Bar

This toolbar accesses all *QA*Director centers. From this single toolbar, navigate from center to center without disrupting workflow.



System Administration Center: Opens the System Administration Center panel

Project Information Center: Opens the Project Information Center panel

Script Information Center: Opens the Script Information Center panel

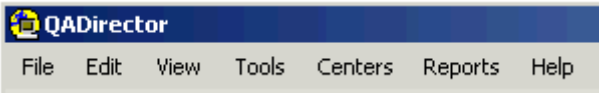
Suite Information Center: Opens the Suite Information Center panel

Job Information Center: Opens the Job Information Center panel

Defect Information Center: Opens the Defect Information Center panel

Using the Menu Bar

The menu bar is located at the top of the QADirector screen:



The following describes the sub-menu items available from the menu items.

Menu Name	Sub Menu		Description
File	New	Suite	Creates a new Suite
		Class	Creates a new Class
		Procedure	Creates a new Procedure
		Script	Creates a new Script
		User	Creates a new user
		Manual Test	Creates a new Manual Test
		Tool Domain	Creates a new Tool Domain
		TM Server	Creates a new TM Server
		Testing Tool	Creates a new Testing Tool
		Filter	Creates a new Filter
		Project	Creates a new Project
	Open...		Opens a Project
	Close Project		Closes a Project
	Import	04.05.01 Suite Exchange File...	Imports 04.05.01 Suite Exchange File

	Note: This feature is not available if using *QA*Director with CARS Workbench.	05.00 or greater versions of Project Exchange File...	Imports 05.00 Project Exchange File
		MS Word/Excel Import Wizard for Suites...	Opens the MS Word/Excel Import Wizard for Suites
	Export... **Note:** This feature is not available if using *QA*Director with CARS Workbench.		Opens the Export Project Information dialog box to save the project as an Exchange file
	Recent Projects		Displays a sub-menu of recent projects
	Exit		Exits *QA*Director
Edit	Properties...		Displays the User Properties Dialog Box
	Cut		Removes a selected item
	Copy		Copies a selected item
	Paste		Pastes a selected item
	Paste Multiple...		Pastes multiple versions of a selected item
	Duplicate		Duplicates a selected item, retaining its parents' settings
	Delete		Permanently removes a selected item
	Rename		Allows an item to be renamed
View	Test Library		Displays or hides the Test Library
	Centers bar		Displays or hides the Centers bar
	Status bar		Displays or hides the Status bar
	Collapse		Collapses a selected item on a tree
	Collapse All		Collapses the entire tree
	Expand		Expands a selected item on a tree
	Expand All		Expands the entire tree
	Refresh		Updates information on the current screen

Tools	User Options...		Displays the User Options Dialog Box
	Change Password...		Displays the Change Password Dialog Box
	Manage Custom Menus...		Displays the Customize Custom Menu Dialog Box
	Manage Filters...		Displays the Manage Filters Dialog Box
	Manage Template Attributes...		Displays the Template Attributes Dialog Box
	*QA*Center		Displays a list of QACenter products. Installed products are enabled on the toolbar.
	Manage Product Integrations...		Displays the Manage Product Integrations Dialog Box
	Export Attributes to Text File... **Note:** This feature is not available if using *QA*Director with CARS Workbench.		Displays a Browse Dialog Box
Centers	System Administration Center	Go to System Administration Center	Changes the panel to System Administration Center
		Launch ODBC Manager...	Displays the ODBC Data Source Administrator Dialog Box
		Change Datasource/Server...	Displays the Select a Database Dialog box
		Role Permissions...	Displays the Manage Role Permissions Dialog Box
		Shut Down TE Server	Shuts down the Test Execution Server
	Project Information Center	Go to Project Information Center	Changes the panel to Project Information Center
		Manage Users...	Displays the Manage Users Dialog Box
	Script Information Center	Go to Script Information Center	Changes the panel to Script Information Center
	Suite Information Center	Go to Suite Information Center	Changes the panel to Suite Information Center

		Suite Permissions...	Displays the Suite Permissions Dialog Box
		Manage Change Logs...	Displays the Managing Logs Dialog Box
		Manage Locks...	Displays the Managing Locks Dialog Box
	Job Information Center	Go to Job Information Center	Changes the panel to Job Information Center
		Manage Job Folders...	Displays the Manage Job Folders Dialog Box
	Defect Information Center	Go to Defect Information Center	Changes the panel to Defect Information Center
Reports	Launch *QA*Center Portal **Note:** This feature is not available if using *QA*Director with CARS Workbench.		Accesses the *QA*Center Portal
Help	Tutorial	Install	Installs the tutorial application
		Help	Opens the help for the tutorial
	Contents		Opens the online help Contents
	Index		Opens the online help Index
	Search		Opens the online help Search
	About *QA*Director		Displays product About information

Using the Toolbar

The toolbar is located at the top of the *QA*Director screen just under the Menu Bar:

The following describes the items available from this toolbar.

Toolbar Name		Description
New	Suite	Creates a new Suite
	Class	Creates a new Class
	Procedure	Creates a new Procedure
	Script	Creates a new Script

	User	Creates a new User
	Manual Test	Creates a new Manual Test
	Tool Domain	Creates a new Tool Domain
	TM Server	Creates a new TM Server
	Testing Tool	Creates a new Testing Tool
	Filter	Creates a new Filter
	Project	Creates a new Project
Properties		Displays the Properties Dialog Box
Refresh Current View		Updates information on the current screen
Save		Saves the changes made to the panel
Save All		Saves all changes made to *QA*Director
Publish		Publishes a specific test
Publish All		Publishes all tests
Cut		Removes a selected item
Copy		Copies a selected item
Paste		Pastes a selected item
Delete		Permanently removes a selected tool or test
Manage Filters		Displays the Manage Filters Dialog Box
Expand		Expands a selected item on a tree
Expand All		Expands the entire tree
Collapse		Collapses a selected item on a tree
Collapse All		Collapses the entire tree
Manage Users		Displays Manage Users Dialog Box
Launch ODBC Manager		Displays the ODBC Data Source Administrator Dialog Box **Note:** This feature is not available

	if using *QA*Director with CARS Workbench..
Change Datasource	Displays the Select a Database Dialog Box **Note:** This feature is not available if using *QA*Director with CARS Workbench.
Manage Role Permissions	Displays the Manage Role Permissions Dialog Box
Show All Scripts	Shows all scripts for a project
Hide Selected Group / Category	Hides a selected group or category
Run Selected Test	Runs a selected test in the project
Manage Change Logs	Displays the Managing Logs Dialog Box
Manage Test Locks	Displays the Manage Locks Dialog Box
Manage Job Folders	Displays the Manage Job Folders Dialog Box
Edit Schedule	Displays Scheduler Dialog Box
Run Now	Runs the job
View Result	Displays the results for a current job
Abort Job	Aborts current job
View Test Result Detail	Displays the Test Properties dialog box for the specific test
View Defect	Displays the *QA*Director View Defect window **Note:** This feature is not available if using *QA*Director with CARS Workbench.
Edit Defect	Opens and displays the actual TrackRecord defect **Note:** This feature is not available if using *QA*Director with CARS Workbench.

Delete Defect Association	Deletes the defect association from the selected test, but the defect still remains in the database. **Note:** This feature is not available if using *QA*Director with CARS Workbench.

Adding Custom Menus

To use third-party tools with *QA*Director, add custom menu commands for specific actions. For example, if integrating a defect tracking tool with *QA*Director, create menu commands for submitting and viewing defects. If no other custom commands exist, the **Custom** menu is not visible in *QA*Director's menu. Once the first command is created, the **Custom** menu will appear.

Using Keyboard Shortcuts

The following table contains all *QA*Director keyboard shortcuts.

Keyboard Shortcut	Description
Ctrl+C	Copies selected suite, class, procedure, script, or text.
Ctrl+V	Pastes copied or cut suite, class, procedure, script, or text to selected location.
Ctrl+X	Cuts selected suite, class, procedure, script, or text.
F1	Launches the dialog level help for the active dialog box.

Using Mouse Keys

The standard Window's Mouse Keys function enables users to use the keyboard's number pad to navigate the mouse.

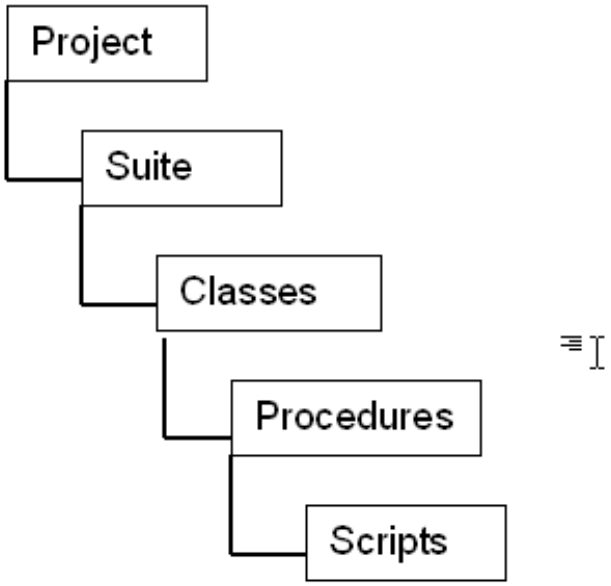
To Enable Mouse Keys:

1. From the Window's Control Panel, click Accessibility Options.
2. Click the Mouse tab.
3. Click the Use Mouse Keys checkbox.
4. To change the settings, click the Settings button.
5. Click Ok and then click Apply. The Mouse Keys icon will appear in the system task tray .The following table describes the actions of the number pad when Use Mouse Keys is enabled.

Number	Direction
1	Down and left

2	Down
3	Down and right
4	Left
5	Enter button
6	Right
7	Up and left
8	Up
9	Up and right
0	Left mouse button click
+	Right mouse button click

Understanding QADirector Objects



Projects are the highest level in the test hierarchy. Projects contain requirements, test suites, defects, product integration information, user information, and the test library.

Suites contain all the test assets (classes and procedures) necessary to run scripts and complete a test. Test suites create order and control by defining which tests are needed, grouping the tests logically, and designating the order in which the tests should be run.

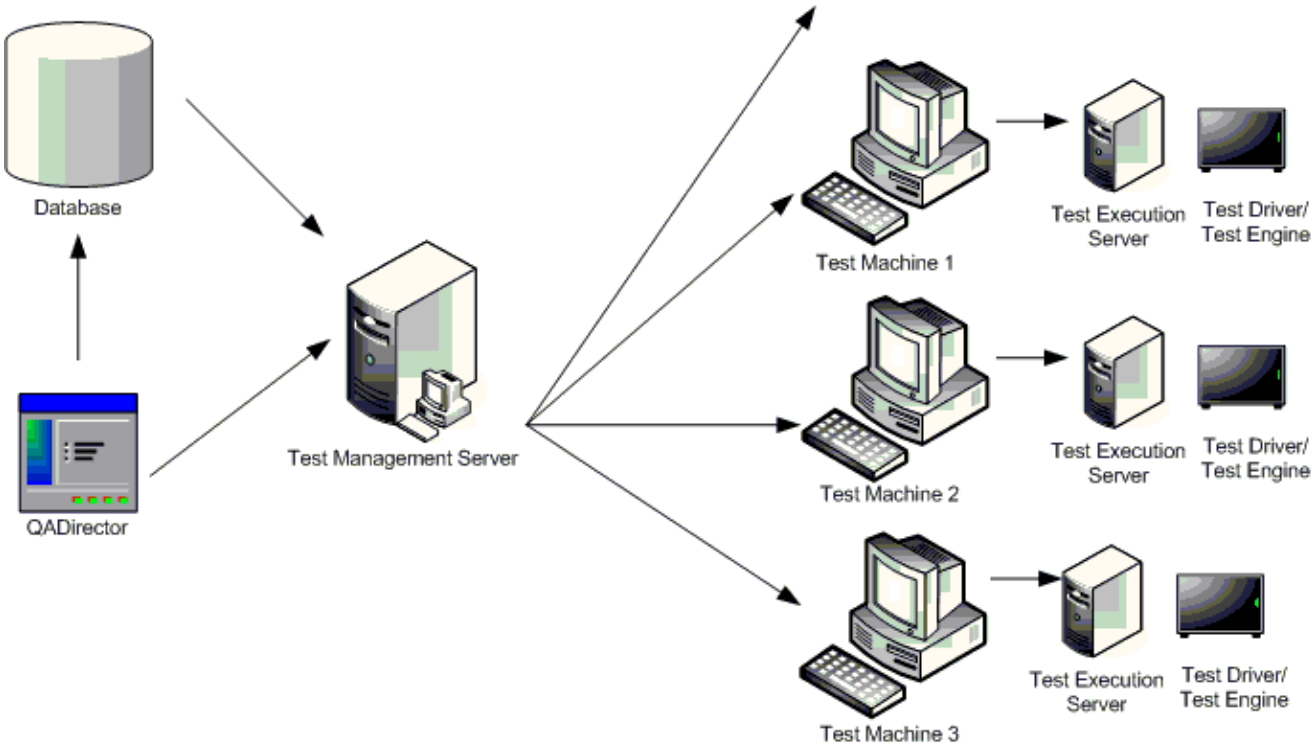
Classes are logical groupings of test procedures and other test classes in the test suite. Create test classes to group tests based on features, functions, or structure of an application, or by any other useful criteria. A test suite can have as many levels of test classes as is necessary.

Procedures contain the scripts that run against test applications. Thus, test procedures are the part of the test suite that check the behavior of the test application and report on the result.

Scripts are commands in test procedures that run a test of an application.

QADirector Processes

The following diagram illustrates how *QA*Director works various servers, databases and test machines.



*QA*Director uses a database repository to store suites, tests, and job results. This database is use automatically unless the user connects to a different database. For more information on creating and configuring databases refer to the *QA*Director Installation and Configuration Guide.

The Test Management Server (TM) reads job submissions from the database, manages jobs, tracks the machines available for test execution, and manages the licenses used for manual test Web execution. It also reports on the status of jobs in progress.

The Test Execution Server allows jobs to run on a computer. When requested by the Test Management Server, the Test Execution Server starts the test driver, which subsequently starts the test engine. The Test Execution Server also stores information about who is permitted to execute jobs on the computer.

*QA*Center Testing Process Outline

Whether you use Compuware's QualityPoint methodology, risk-based testing, or another testing methodology, the *QA*Center suite of products can automate all aspects of the testing process. The following outline provides an overview of the *QA*Center testing process.

1. Manage business requirements in Reconcile.
   Refer to the Reconcile Help system for detailed information about managing business requirements.
2. Import business requirements into *QA*Center for risk-based test planning in *QA*Center Portal.
   Refer to the topic **Importing Requirements into *QA*Center Portal** in the *QA*Center Portal Help system.
3. Create Functional Test Requirement (FTR) from business/imported requirement in *QA*Center Portal.
   Refer to the topic **About Functional Test Requirements** in the *QA*Center Portal Help system.

   a. Assign risk (0-10). Refer to the following topics in the *QA*Center Portal Help system: **Creating Requirements, Test Requirement Percentile Values**, and **Functional Test Requirement Weights**.

   4. Decompose Test Requirements (TR) and assign risk in *QA*Center Portal.

   a. Create Testing Requirement (TR) from FTRs. Refer to the topic **Creating Requirements** in the *QA*Center Portal Help system.

   b. Assign risk to TR (0-10). Refer to the following topics in the *QA*Center Portal Help system: **Creating Requirements, Test Requirement Percentile Values**, and **Functional Test Requirement Weights**.

   c. Determine cycle. Refer to the topic **Test Planning Cycles** in the *QA*Center Portal Help system.

   5. Create test assets from Test Planning in *QA*Center Portal.
   Refer to the following topics in the *QA*Center Portal Help system: **Asset Generation** and **Generating Suites from *QA*Center Portal.**

      a. Create test assets based on risk/cycle. Refer to the following topics in the *QA*Center Portal Help system: **Asset Generation** and **Run-Time Attributes.**

   6. Create test scripts and assign to procedures in *QA*Director.

   a. Create Manual Test Scripts.

   b. Create User-defined Test Scripts.

   c. Create Automated Test Scripts.

   7. Schedule test execution.

   8. Execute the test.

      a. Run Manual tests or Web Tests.

      b. Run the Automated Testing Tools.

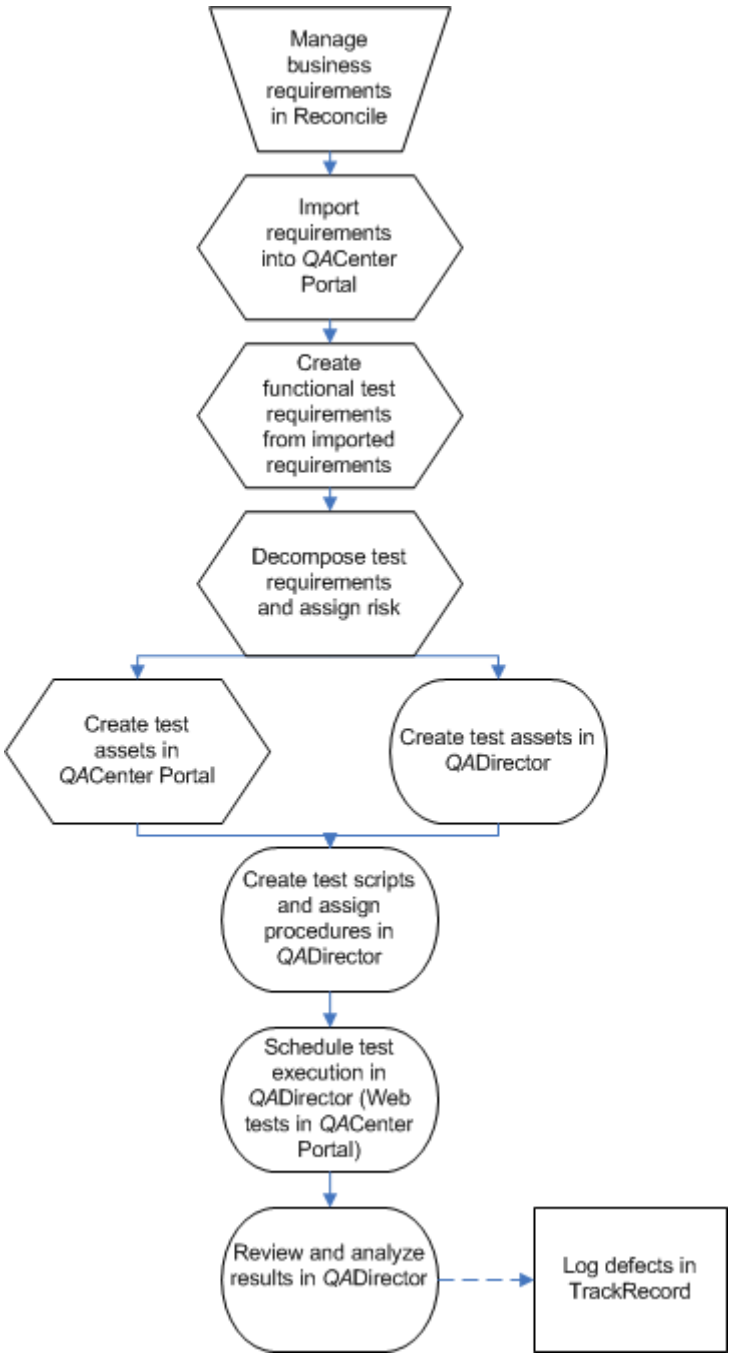
9. Review and analyze project data.

      a. View detailed results.

      b. Manage defects from *QA*Center Portal or *QA*Director. Refer to the topic **Defect Tracking** in the *QA*Center Portal Help system.

*QA*Center Testing Process Diagram

The following diagram illustrates the testing process using the *QA*Center suite of products.



About Risk-Based Testing

*QA*Center uses risk-based testing methodology. Risk-based testing is a testing methodology that reduces the risk of distributing an application that doesn't meet business requirements and that does not function reliably. It helps test organizations determine what things to test and prioritize testing based on the cost of failure. The greater the probability of expensive failure, the more important it is to test that feature.

Adopting risk-based testing greatly improves the quality of applications, maximizes the utilization of resources, and supports an on-time distribution. This is accomplished by:

- aligning business requirements with testing efforts
- using risk to prioritize the amount and kind of testing that's required

identifying and quickly responding to changing business requirements

tracing the results back to the business requirements to measure risk in business terms

make necessary go/no-go decisions

Risk-based testing is a critical part of an overall testing methodology that aligns with the software development life cycle. Risk-based testing, when used with a comprehensive testing methodology, provides a repeatable process that facilitates continuous quality improvement.

Basic Risk-Based Testing Steps

The five basic steps to risk-based testing are:

1. Identify time-critical business issues.
2. Perform risk assessment.
3. Establish baseline.
4. Execute tests.
5. Validate and document test results.

*QA*Center and Risk-Based Testing

Compuware's risk-based testing solution provides organizations with an objective mechanism to weigh and prioritize what to test. It provides management visibility into their organization's testing efforts and the knowledge to make informed go/no go decisions. Compuware's approach links testing to business requirements by defining functional test requirements.

Assign risk to Functional Test requirements based on various business, technical and historical factors.

Create Test Cases to prove the requirements and executed based on their assigned priorities.

Summarize quality statistics by business requirement and priority.

Should requirements change, Compuware's risk-based testing solution automates the creation of a revised testing plan. The solution allows reuse of testing assets throughout iterative testing cycles to expedite on-time delivery.

QACenter risk-based testing provides an approach to prioritizing testing that relies on the quality assurance users to assign risk and priority to tests. It provides management visibility into their organization's testing efforts on a project-by-project basis and the knowledge to make informed go/no go decisions. Compuware's *QA*Center products make this process accurate, repeatable, and easy. *QA*Center walks organizations through the entire testing process.

Benefits of Risk-Based Testing

The benefits of risk-based testing are numerous. A few are:

Defined process for improvement

Measure quality in business terms

- Quickly realign testing efforts
- Repeatability
- Effective use of resources
- Reduction in customer reported problems
- Reduction in maintenance cost
- Reduction in test phase length
- Find faults
- Plan for the future

Tutorial

Tutorial Overview

The *QA*Director Tutorial will guide you through each of the Centers used to Plan and execute tests, from the System Administration Center to the Defect Information Center.

The *QA*Director Tutorial uses a sample test project, a sample suite, sample test classes, sample TestPartner scripts, and a demonstration application called the Millennium International Bank (MIB) to introduce you to the basic concepts of *QA*Director. By executing the sample TestPartner scripts against the MIB demonstration application, you will learn the basic functions of *QA*Director.

You will not access the MIB demonstration application directly. The work you will perform for this tutorial is performed in *QA*Director and TestPartner.

Requirements

Current version of *QA*Director

TestPartner version 5.2.1 or later and all its requirements including MS Access

Valid license for TestPartner

TrackRecord version 06.02

Components of the Tutorial

To support the tutorial, the following components are installed on your computer:

Demo Application

A demo application named Millennium International Bank (MIB) is installed with *QA*Director. The MIB demo application is a simple banking application with seven main functions:

Create Customer Account

Inquire on Account

Deposit Funds

Withdraw Funds

Transfer Funds

Close Account

Exit

Tutorial Project

The Tutorial - {Username} project is created when the tutorial is selected from **Help>Install Tutorial**. This is a project created to test the demo application. You will be required to add six test classes and six test procedures. To each test procedure, you will add a test script.

TestPartner Scripts

TestPartner scripts are in a database named QAD4.MDB. Add the scripts to the test procedures in the MIBTest suite in order to test the demo application.

Terminology

Before beginning the tutorial lessons, you may find it helpful to review some *QA*Director terminology:

Test Suite

Test Class

Test Procedure

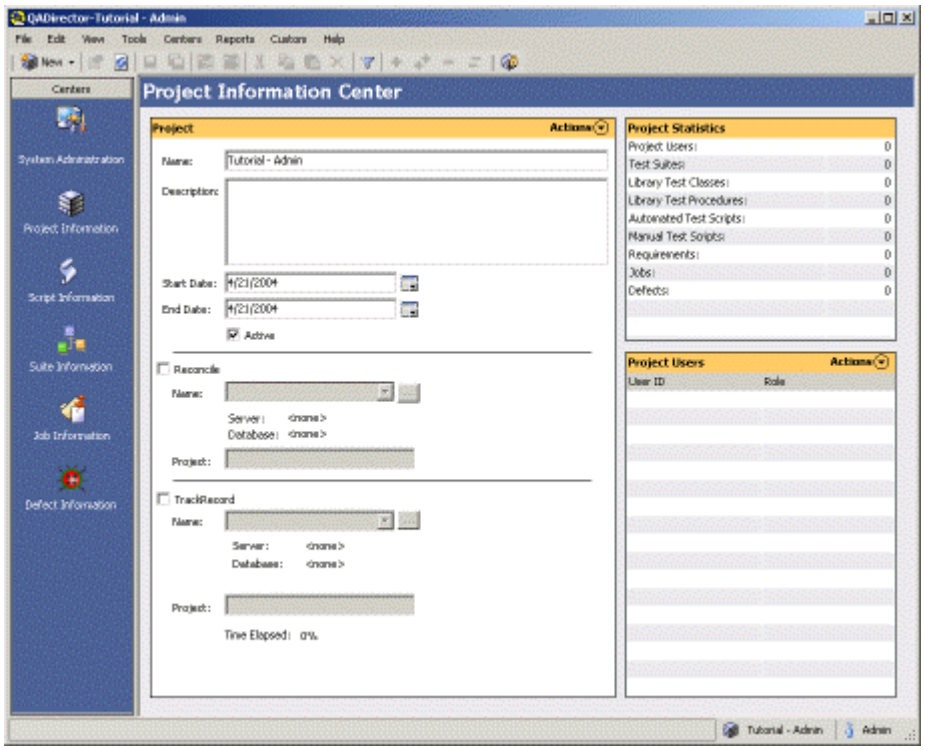
Test Script

Test

Tool Domain

Exploring the Main Window

The project for this tutorial is already created in *QA*Director when the tutorial is selected from **Help>Install Tutorial**. It resides in the Project Information Center. Take a minute to study the main window.



For more information see About the GUI.

Where to go next: Tutorial Setup Tasks

Tutorial Overview

The *QA*Director Tutorial will guide you through each of the Centers used to Plan and execute tests, from the System Administration Center to the Defect Information Center.

The *QA*Director Tutorial uses a sample test project, a sample suite, sample test classes, sample TestPartner scripts, and a demonstration application called the Millennium International Bank (MIB) to introduce you to the basic concepts of *QA*Director. By executing the sample TestPartner scripts against the MIB demonstration application, you will learn the basic functions of *QA*Director.

You will not access the MIB demonstration application directly. The work you will perform for this tutorial is performed in *QA*Director and TestPartner.

Requirements

Current version of *QA*Director

TestPartner version 5.2.1 or later and all its requirements including MS Access

Valid license for TestPartner

TrackRecord version 06.02

Components of the Tutorial

To support the tutorial, the following components are installed on your computer:

Demo Application

A demo application named Millennium International Bank (MIB) is installed with *QA*Director. The MIB demo application is a simple banking application with seven main functions:

Create Customer Account

Inquire on Account

Deposit Funds

Withdraw Funds

Transfer Funds

Close Account

Exit

Tutorial Project

The Tutorial - {Username} project is created when the tutorial is selected from **Help>Install Tutorial**. This is a project created to test the demo application. You will be required to add six test classes and six test procedures. To each test procedure, you will add a test script.

TestPartner Scripts

TestPartner scripts are in a database named QAD4.MDB. Add the scripts to the test procedures in the MIBTest suite in order to test the demo application.

Terminology

Before beginning the tutorial lessons, you may find it helpful to review some *QA*Director terminology:

Test Suite

Test Class

Test Procedure

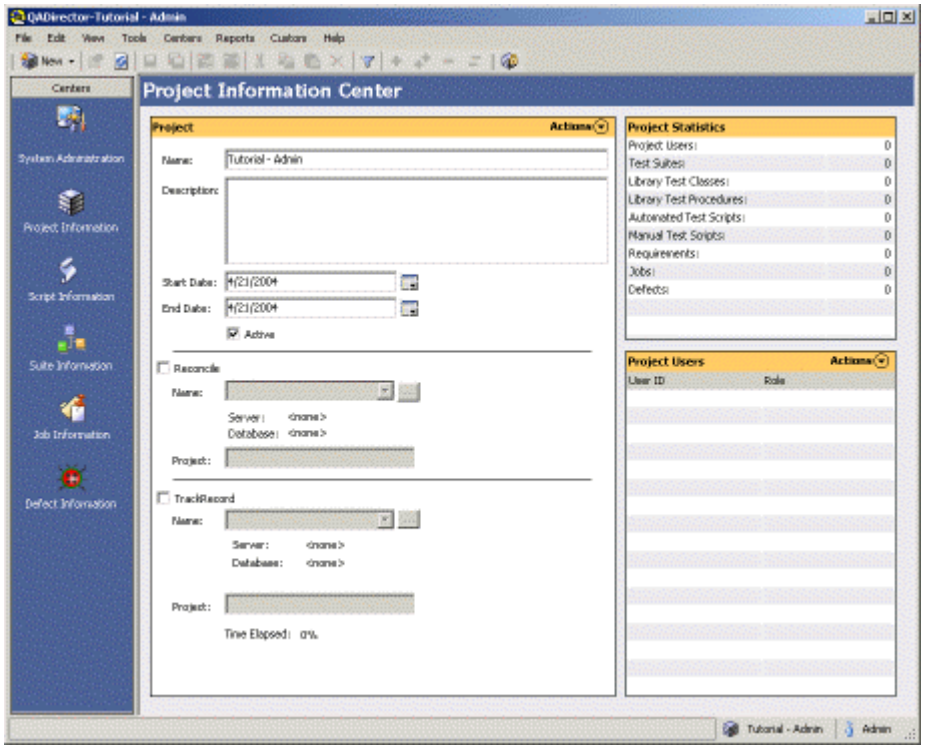
Test Script

Test

Tool Domain

Exploring the Main Window

The project for this tutorial is already created in *QA*Director when the tutorial is selected from **Help>Install Tutorial**. It resides in the Project Information Center. Take a minute to study the main window.



For more information see About the GUI.

Where to go next: Tutorial Setup Tasks

Tutorial Setup Tasks

The following describes setup tasks to be completed before beginning the tutorial. These tasks are normally completed by a user with administrator privileges to perform user and database maintenance.

Configure the User Names and Passwords

Both *QA*Director and TestPartner use a database, which allows team members to easily share project suites, tests, and scripts. To ensure database security, both *QA*Director and TestPartner employ a combination of user passwords and permissions to designate who has access to the databases.

Make sure that the User Name and Password used for *QA*Director are the same User Name and Password used for TestPartner.

In *QA*Director, user maintenance is performed in the **System Administration Center**.

Establish a TrackRecord / QADirector Integration

To complete the section of this tutorial that uses TrackRecord, you must establish a TrackRecord / *QA*Director integration before starting the tutorial. For integration information see Creating a Product Integration.

In order to complete this tutorial, the administrator must give the user privileges to view and edit the tool domain. For information on setting up a tool domain see Creating Tool Domains.

Create an ODBC Datasource and Add Scripts

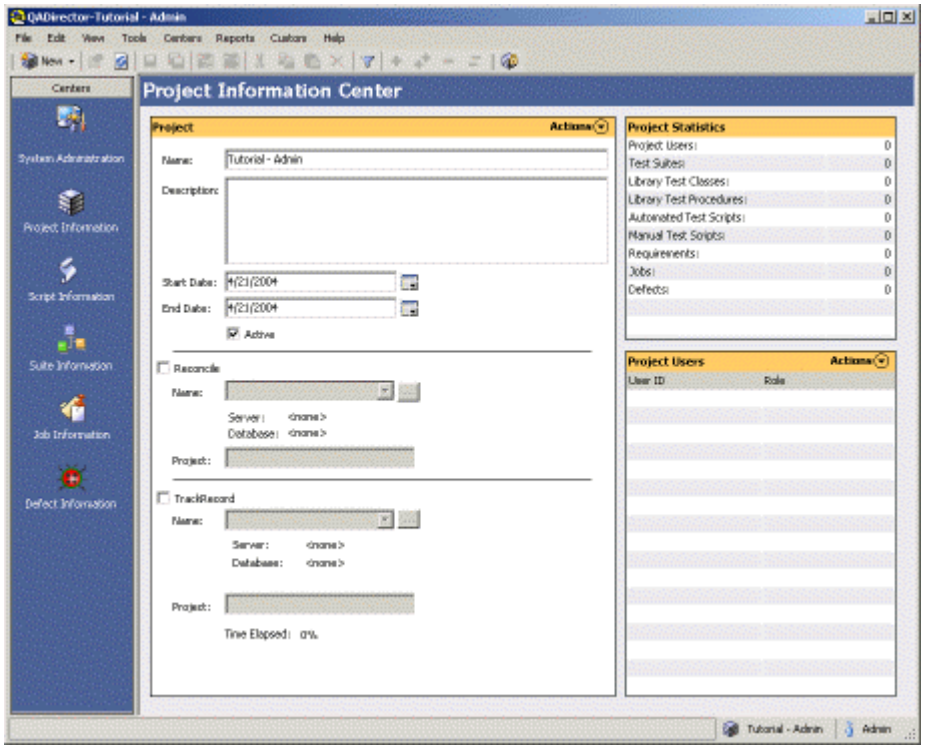
Create an Access ODBC datasource and add scripts to use with TestPartner:

1. From the **System Administration Center** Datasource panel, click **Actions>Launch ODBC Manager**. The ODBC Data Source Administrator dialog box appears.

2. Click **System DSN>Add.** The Create New Data Source dialog box appears.
3. Select Microsoft Access Driver and click **Finish**. The ODBC Microsoft Access Setup dialog box appears.
4. In the **Data Source Name** and **Description fields** type *QAD Tutorial*.
5. Click **Select** and navigate to the \Program Files\Compuware*QA*Director\Tutorial\Mibqadb directory.
6. Double click the file QAD4.mdb. This is the TestPartner database that contains the sample scripts for testing the MIB application.
7. Click **OK**.

Open the Tutorial

Open the *QA*Director Tutorial by clicking **Help>Tutorial>Install**. The Tutorial Project opens in the Project Information Center.



Where to go next: Lesson One: System Administration Center

Tutorial Lessons

Lesson One: The System Administration Center

In the System Administration Center, you will create a TestPartner tool Domain. A tool domain is a set of information that you enter about the testing tool that allows you to browse, select, and create scripts right from *QA*Director. For example, a TestPartner tool domain includes the name and location of the TestPartner script database, the database type, whether or not the database is public (can be used by others), and so on. Make sure a Test Management Server and Test Execution Server are available.

1. In the Tool Domains Pane click **Actions>New Tool Domain.** The **Add Tool Domain** dialog box appears. Fill in this dialog box with the information below:

   Name Field: Type *MIBTest TestPartner Scripts*.

   Description Field: Type *MIBTest TestPartner Scripts*.

   Type list: Choose *TestPartner*

   Public: Select

   Testing Tool Database field: Click the **Browse** button and select QAD Tutorial.

   User Name field: Type **Admin**.

   Password field: Type **Admin**.

2. Click **Apply**
3. Click **OK**

Where to go next: Lesson Two: The Project Center

Lesson One: The System Administration Center

In the System Administration Center, you will create a TestPartner tool Domain. A tool domain is a set of information that you enter about the testing tool that allows you to browse, select, and create scripts right from *QA*Director. For example, a TestPartner tool domain includes the name and location of the TestPartner script database, the database type, whether or not the database is public (can be used by others), and so on. Make sure a Test Management Server and Test Execution Server are available.

1. In the Tool Domains Pane click **Actions>New Tool Domain.** The **Add Tool Domain** dialog box appears. Fill in this dialog box with the information below:

   Name Field: Type *MIBTest TestPartner Scripts*.

   Description Field: Type *MIBTest TestPartner Scripts*.

   Type list: Choose *TestPartner*

   Public: Select

   Testing Tool Database field: Click the **Browse** button and select QAD Tutorial.

   User Name field: Type **Admin**.

   Password field: Type **Admin**.

2. Click **Apply**
3. Click **OK**

Where to go next: Lesson Two: The Project Center

Lesson Two: Project Information Center

From the Project Information Center, you will set up an integration to TrackRecord. TrackRecord is Compuware's defect-tracking tool for distributed applications.

In the Project Pane of the Project Information Center you will see that the Tutorial Project is displayed.

1. Choose the **TrackRecord** checkbox.
2. From the **Name** list choose the **TrackRecord** Integration. This should have been established before starting the Tutorial lessons. See Tutorial Setup Tasks.

Where to go next: Lesson Three: The Script Information Center

Lesson Three: Script Information Center/ Test Library

From the Script Information Center, add the scripts that you will use in the project to the Test Library. This is a repository for all available tests within a project. It contains test classes, test procedures, and test scripts. The Test Library makes it possible to use these tests in several different suites within a project. As tests are used across several suites, changes are saved to the Test Library and all suites. For more information see About the Script Information Center.

1. From the **Test Tool** list choose TestPartner. The TestPartner tool domain appears.
2. Select all the MIB TestPartner scripts:

   CloseAccount

CreateNewAccount

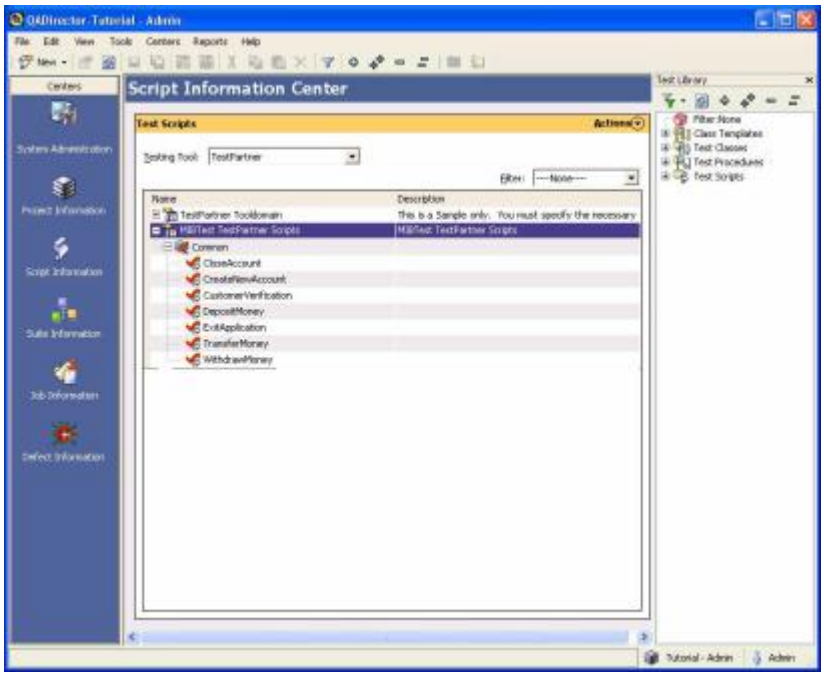
CustomerVerification

DepositMoney

ExitApplication

TransferMoney

WithdrawMoney



3. Do one of the following to move the scripts into the Test Library:

   Right click each script and choose **Add to Library.**

   Drag the scripts to the library and drop them in.

The scripts are now in the library. If you go back into the Project Center, you will notice in the Project Statistics pane that the Automated Test Scripts category is now populated with the number of scripts you moved to the library.

Where to go next: Lesson Four: The Suite Information Center

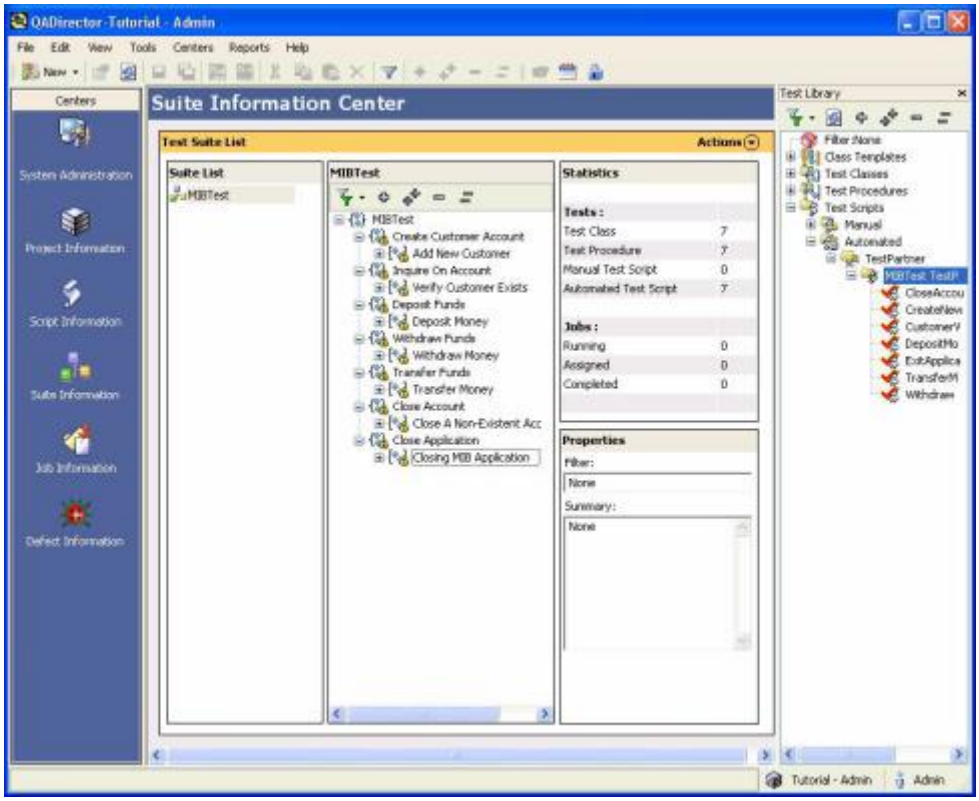
Lesson Four

Lesson Four: Suite Information Center

The **Suite Information Center** contains all of the test assets (classes and procedures) necessary to run scripts and complete a test. In *QA*Director, tests are organized in test suites. Typically, a test suite contains all the tests needed that cover one application or one component of an application. Test suites create order and control by defining which tests are needed, grouping the tests logically, and designating the order in which the tests should be run.

A test suite is comprised of test classes and test procedures, all of which descend from a single test class called the root class. The root class has the same name as the test suite and cannot be deleted. All other test classes and test procedures descend from the root class.

Test classes help maintain order within a suite. The MIBTest suite is designed to include seven test classes, one for each of the six main components in the MIB demo application. You will create these seven test classes, procedures and scripts using the identical process.

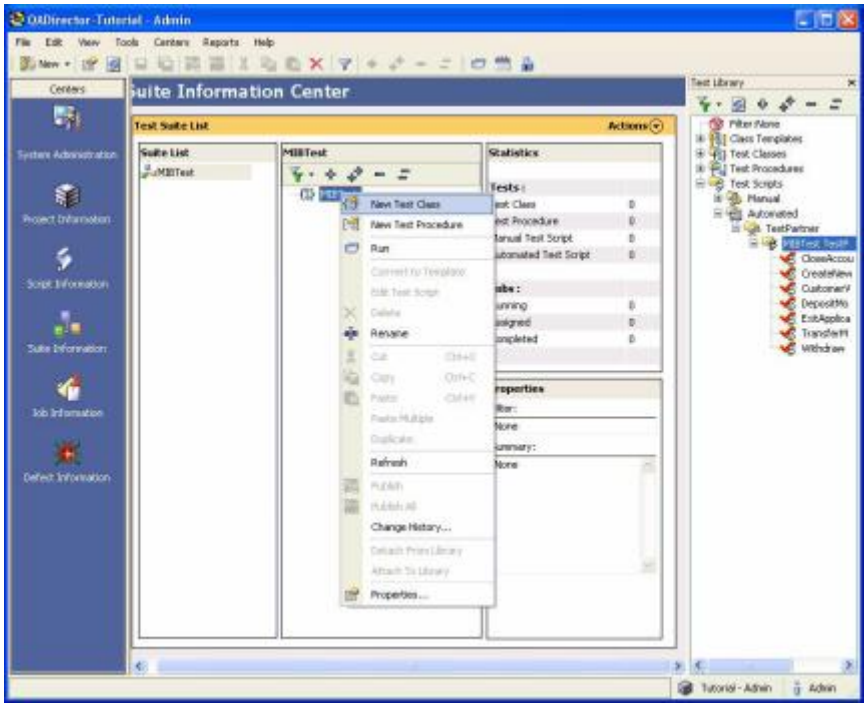


Where to go next: Step 1: *Create Customer Account* test class and Associated Procedure and Script

Step 1: *Create Customer Account* Test Class and Associated Procedure and Script

Create a Suite, Test Class, Procedure and Script. You will create the Suite only once, in this step. In the subsequent steps you will only add the class, procedure and script to this suite.

1. Click **Actions>New Suite**
2. Name the suite *MIBTest*
3. In the center pane, right click the *MIBTest* suite and choose **New Test Class**.



4. Name the test class *Create Customer Account*.
5. Right click this test class and choose **New Test Procedure**.
6. Name the test procedure *Add New Customer*.

Now add the script to the procedure.

1. Double click the test procedure you have just created.
2. Click the **Scripts** tab.
3. Click **Insert**. The Add Test Scripts from Library dialog box appears.
4. Choose the TestPartner script *CreateNewAccount,* and click **Add**. If the script is not there, go back to Lesson 3 and add it.
5. Click **OK**.

Where to go next: Step 2: *Inquire on Account* Test Class and Associate Procedure and Script

Step 2: *Inquire on Account* Test Class and Associated Procedure and Script

1. In the center pane, right click the *MIBTest* suite and choose **New Test Class**.
2. Name the test class *Inquire on Account*.
3. Right click this class and choose **New Test Procedure**.
4. Name the test procedure Verify Customer Exists.

Now add the script to the procedure.

1. Double click the test procedure you have just created.
2. Click the **Scripts** tab.
3. Click **Insert**. The Add Test Scripts from Library dialog box appears.
4. Choose the TestPartner script *CustomerVerification*, and click **Add**. If the script is not there, go back to Lesson 3 and add it.

5. Click **OK**.

Where to go next: Step 3: *Deposit Funds* Test Class and Associated Procedure and Script

Step 3: *Deposit Funds* Test Class and Associated Procedure and Script

1. In the center pane, right click the *MIBTest* suite and choose **New Test Class**.
2. Name the test class *Deposit Funds*.
3. Right click this class and choose **New Test Procedure**.
4. Name the test procedure *Deposit Money*.

Now add the script to the procedure.

1. Double click the test procedure you have just created.
2. Click the **Scripts** tab.
3. Click **Insert**. The Add Test Scripts from Library dialog box appears.
4. Choose the TestPartner script *DepositMoney* and click **Add**. If the script is not there, go back to Lesson 3 and add it.
5. Click **OK**.

Where to go next: Step 4: *Withdraw Funds* Test Class and Associated Procedure and Script

Step 4: *Withdraw Funds* Test Class and Associated Procedure and Script

1. In the center pane, right click the *MIBTest* suite and choose **New Test Class**.
2. Name the test class *Withdraw Funds*.
3. Right click this class and choose **New Test Procedure**.
4. Name the test procedure *Withdraw Money*.

Now add the script to the procedure.

1. Double click the test procedure you have just created.
2. Click the **Scripts** tab.
3. Click **Insert**. The Add Test Scripts from Library dialog box appears.
4. Choose the TestPartner script *WithdrawMoney* and click **Add**. If the script is not there, go back to Lesson 3 and add it.
5. Click **OK**.

Where to go next: Step 5: *Transfer Funds* Test Class and Associated Procedure and Script

Step 5: *Transfer Funds* Test Class and Associated Procedure and Script

1. In the center pane, right click the *MIBTest* suite and choose **New Test Class**.
2. Name the test class *Transfer Funds*.
3. Right click this class and choose **New Test Procedure**.
4. Name the test procedure *Transfer Money*.

Now add the script to the procedure.

1. Double click the test procedure you have just created.
2. Click the **Scripts** tab.
3. Click **Insert**. The Add Test Scripts from Library dialog box appears.

4. Choose the TestPartner script *TransferMoney* and click **Add**. If the script is not there, go back to Lesson 3 and add it.
5. Click **OK**.

Where to go next: Step 6: *Close Account* Test Class and Associated Procedure and Script

Step 6: *Close Account* Test Class and Associated Procedure and Script

1. In the center pane, right click the *MIBTest* suite and choose **New Test Class**.
2. Name the test class *Close Account.*
3. Right click this class and choose **New Test Procedure**.
4. Name the test procedure *Close a Non-Existent Account.*

Now add the script to the procedure.

1. Double click the test procedure you have just created.
2. Click the **Scripts** tab.
3. Click **Insert**. The Add Test Scripts from Library dialog box appears.
4. Choose the TestPartner script *CloseAccount* and click **Add**. If the script is not there, go back to Lesson 3 and add it.
5. Click **OK**.

Where to go next: Step 7: Close MIB Application

Step 7: *Close MIB Application* Test Class and Associated Procedure

1. In the center pane, right click the *MIBTest* suite and choose **New Test Class**.
2. Name the test class *Close Application.*
3. Right click this class and choose **New Test Procedure**.
4. Name the test procedure *Closing MIB Application.*

Now add the script to the procedure.

1. Double click the test procedure you have just created.
2. Click the **Scripts** tab.
3. Click **Insert**. The **Add Test Scripts from Library** dialog box appears.
4. Choose the TestPartner script *ExitApplication* and click **Add**. If the script is not there, go back to Lesson 3 and add it.
5. Click **OK**.

Where to go next: Step 8: Add Rules

Step 8: Add Rules

A rule is a command that prepares a script for execution, collects information during an execution, determines if a script failed, or cleans up after the execution. Rules can be defined and applied to test classes, test procedures, and jobs. For this task, you will add an environment variable rule and a setup rule. The environment variable will point to the MIB demo application path, and the setup rule will start the MIB demo application.

After the Suite has been set up and all the classes, procedures and scripts are added, follow these steps:

1. In the center pane, double click the Suite *MIBTest*. The Test Class dialog box appears.
2. Click the Execution tab

3. Click the Rules button. The Rules dialog box appears.
4. The path will be appended to your existing path environment variable.
5. Click the **Add** button. The Rule dialog box appears.
6. Click **Environmental Variable**.
7. Set the remaining fields to the following values:

   Name: Type *Path*

   Value: Type C:\PROGRA~1\COMPUW~1\QADIRE~1\TUTORIAL\MIBApp;%PATH%. The path will be appended to your existing path environment variable.
8. Click **OK**.
9. Click the **Add** button again. The Rule dialog box appears.
10. Click **Setup**. Setup rules run commands that prepare for test execution, such as starting processes or copying files. You will add a setup rule that will start the MIB demo application. Note that the execution of the MIB demo application could be performed via a TestPartner script, but for this lesson it is performed via a setup rule.
11. In the Command field, type: *Start MIB.EXE* . You must precede the command with the word "start."
12. Set the remaining fields to the following values:

    Bind to Machine: Leave this blank

    Apply To: Choose *Locally*

    Time Out: Default

    Exit Status: Default
13. Click **OK**. The Rules dialog box appears. The value is added to Rules for this Test field.
14. Click **Close**

Where to go next: Step 9: Run a Job

Step 9: Run a Job

The first step in running a job is to select the tests to run. You can choose to run an entire test suite, which means all the test classes and test procedures will run, or you can run specific test classes or test procedures. For this task, you will run the entire tutorial test suite. The first step in running a job is to select the tests to run. Then, describe how and when the tests will run. This is called the job description. After submitting the job, it normally runs as soon as a Test Execution Server is available on a computer.

Right click on the MIBTest Suite and click **Run**. The Job Description dialog box appears.

1. Click **Submit**. A message appears verifying that the job was submitted.
2. Click **OK**.

The job should take a few minutes to run, during which time you will see the TestPartner scripts perform tests on the MIB application. Remember, the MIB application is started via the environment and setup rules that you defined.

Note: The demo runs automatically. Do not perform any other actions on your computer while the job is running.

Where to go next: Lesson Five: Job Information Center

Lesson Five: Job Information Center

The *QA*Director **Job Information Center** lists all scheduled jobs, jobs in progress, and results. From within this center, perform job execution tasks. The **Jobs** panel is divided into two sections: result folders and the job related tabs. The job related tabs are **Schedule**, **Execution**, and **Results**. The **Schedule** tab displays jobs that are in queue to execute The **Execution** tab displays jobs that are currently running. The **Results** tab displays the results of jobs that have executed.

1. Click the **Results** tab in the main Job Information Center pane.
2. Right click the MIBTest result and choose **View Results**. The Job Results window appears with detailed information.
3. Note that the Transfer Money test failed. Right click the Transfer Money test and choose **View Details.** The Test Procedure dialog box appears and includes logs from the test.
4. Double click **TP-view**. The information in the log view states that the script failed due to the failure of a TestPartner check.
5. After reading the TestPartner Logview, click **File>Exit** from the TestPartner menu.
6. Click **OK** to close the Test Procedure dialog box.

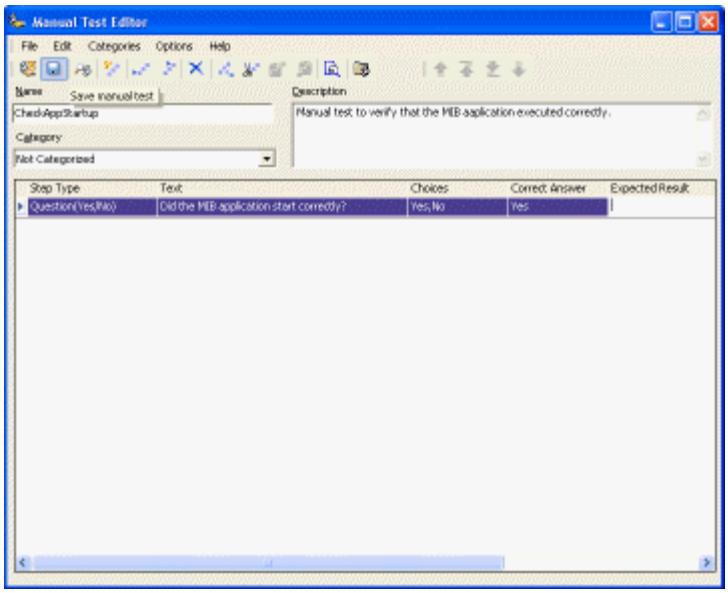
Where to go next: Lesson Six: Creating a Manual Test

Lesson Six: Create a Manual Test

A manual test is a test that requires interaction from the tester, whether it be answering questions or performing some action. It is possible to use manual tests when automated testing is impractical or not yet implemented. Even an automated test plan is likely to include tasks that cannot be easily automated. In that case, use manual tests to complete these tasks. Manual tests also provide a way for Quality Assurance testers to comment on software interface and usability.

To create a manual test, you design a series of steps that the tester reads and responds to when the test script is executed. For this task, you will use the Manual Test Manager to create a manual test, which you will later add to a test procedure.

1. From the Script Information Center, choose **Manual** from the **Testing Tool** list.
2. Click **Action>New Script.** The **Manual Test Editor** appears.
3. In the Name field, type **CheckAppStartup.**
4. In the **Description** field, type: `Manual test to verify that the MIB application executed correctly`
5. In the **Step Type** list, choose Question (Yes/No).
6. In the Text field, type the question: `Did the MIB application start correctly?`
7. "Yes, No" appear in the Choices field.
8. In the **Correct Answer** field, choose Yes from the list.
9. If necessary, add more steps to the test, but this is not required for the tutorial.



10. Click **File>Save** or the **Save** icon.
11. Click **File>Close**.
12. Add the script to the **Test Library.**

Where to go next: Step 1: Insert the Manual Test into a Test Procedure

Step 1:Insert a Manual Test Into a Manual Procedure

1. From the **Suite Center**, double click the **Add New Customer test procedure** dialog box.
2. On the Scripts tab, click **Insert** to open the **Add Test Scripts from Library** dialog box.
3. From the **Testing Tool** list, choose Manual Test.
4. Choose the CheckAppStartup manual test.
5. Click **Add.**
6. Click **OK.**
7. Choose the CheckAppStartup test in the Test Scripts list. Click the up arrow to the right of the **Test Scripts** list to change the order of execution for the tests. You must run the manual test before the *QA*Run script to verify that the MIB application ran successfully.
8. Click **Apply**.
9. Click **OK.** In the Tree View, you should see that the CheckAppStartup manual test has been added to the Add new customer test procedure.
10. Right click MIBTest in the middle pane and choose Run. The **Job Description** dialog box appears.
11. Choose the **Manual Test** checkbox.
12. Click **Submit**. A confirmation dialog box appears. Click **OK**. The box disappears within five seconds if **OK** is not clicked. The **Manual Test Execution** screen appears.
13. Choose Pass from the **Actions** list.
14. Click **Finish Job**.

Where to go next: Step 2: Submitting a Defect

Step 2: Submit a Defect

1. From the **Job Information Center**, right click the MIBTest.
2. Choose **View Results.** The Job Results window appears.
3. Click **Transfer Money Procedure.**
4. Right click **Submit Defect**. If this is not available, make sure you have configured TrackRecord. The **Submit Defect** dialog box appears.
5. Click **Close**.
6. Click **Close** on the Job Results window.

Where to go next: Lesson Seven: Defect Information Center.

Lesson Seven: Defect Information Center

TrackRecord is Compuware's defect-tracking tool for client/server and mainframe applications. To complete this lesson, you must have TrackRecord installed on the same machine as *QA*Director or on an accessible network drive.

In previous lessons, you found that the Transfer Money test procedure failed, due to a failed TestPartner check. You will now log a defect in TrackRecord for the failed test procedure.

1. Open the **Defect Information Center**.
2. Click the MIBTest job in the pane to the left. All the defects associated with this job appear in the pane to the right.
3. Choose the defect you submitted.
4. Click **Actions>View Defect.** The View Defect window appears with all the pertinent information for the defect, including the ID number, status, and the test name. From here you can edit the defect by clicking **Edit Defect**.
5. Click **Close** to close the View Defect window.

Administrating and Configuring in the System Administration Center

About Administrating and Configuring the System

After installing *QA*Director, open the **System Administration Center** to perform a variety of tasks such as assigning user passwords and permissions and maintaining the *QA*Director database. A designated *QA*Director administrator must perform tasks associated with passwords and permissions.

Also, use the **System Administration Center** to connect to databases and add testing tools.

Refer to the *QADirector Installation and Configuration Guide* for more information on administering and configuring *QA*Director.

About Administrating and Configuring the System

After installing *QA*Director, open the **System Administration Center** to perform a variety of tasks such as assigning user passwords and permissions and maintaining the *QA*Director database. A designated *QA*Director administrator must perform tasks associated with passwords and permissions.

Also, use the **System Administration Center** to connect to databases and add testing tools.

Refer to the *QADirector Installation and Configuration Guide* for more information on administering and configuring *QA*Director.

Passwords and Permissions

Only *QA*Director administrators and users with User Management permissions can perform user maintenance. A *QA*Director administrator is a team member who is assigned all permissions. Administrator privileges are granted from the **Administrator** checkbox located in the user's properties. When *QA*Director is first installed, administrators can use the default Admin ID to log on to *QA*Director and to begin creating other User Names. Also, Administrators and users with User Management permissions must have Windows Administrator privileges.

*QA*Director's architecture is based on a multi-user repository, or database. This database allows teams to work together, sharing tests and test results. The *QA*Director administrator, controls access to the database by specifying the following:

> **IDs and passwords:** Set up a user name and password for each person who must have access the database.
>
> **Access permissions:** Set the access permissions for each user. Access permissions specify what actions the user can perform in the database—create or delete tests, add tool domains, etc.

Users have unique passwords and are assigned permissions for working in *QA*Director. Use the same user name and password among *QA*Director, *QA*Run, and TestPartner. By having the same ID and password throughout these products, the user will not be prompted for logon information whenever they access *QA*Run, and TestPartner while editing and creating tests in *QA*Director. Remember that, during execution, *QA*Director uses the Tool Domain user name and password.

Do not use spaces in a user name. If used, spaces will automatically be replaced with underscores. Passwords must be more than five characters. Passwords with less than five characters will be replaced with the word "*QA*Director." User names with more than 50 characters will be truncated.

Compuware recommends that you change the default password for the Admin user so that unauthorized users cannot access user maintenance.

To assign user passwords and permissions:

1. Click **Start** and open *QA*Director.
2. When prompted, type a user name and password.
3. For an administrator installing *QA*Director for the first time, type `admin` in both the **ID** and **Password** fields. This is a default user name with administrator privileges.
4. From the **Centers** toolbar, choose **System Administration**.

Note: If the **System Administration** icon is disabled, your user name does not have administrator or sufficient privileges.

Because user information is stored in the database, you must be connected to the appropriate database before adding users: To view your current database connection, review the data within the **Data Source** panel. To connect to a different database, refer to Connecting to a Database.

5. From the **Users** panel **Action** menu, click **New**. The Add User dialog box appears.
6. In the **Login Information** area, type the user name and password. Type the password again in the **Confirm Password** field.
7. Select a role from the **Role** list.
8. To give the user administrator privileges, select the **Administrator** check box. Administrator and sufficient privileges allow users to assign user names and perform database maintenance. When assigning administrative privileges to a user, all of the access permissions are automatically assigned to them as well.

Required: All users with *QA*Director Administrative, User Management, or DB Management privileges must have Windows Administrative privileges.

9. In the **User Information** section, type the user's name, department, e-mail address, and phone number in the appropriate fields. These fields are optional.
10. Click **OK**.

Available Access Permissions

The following is a list of all available access permissions in *QA*Director. The permissions are organized by management function. For more information about roles, management functions, passwords, and permissions. See Assigning user passwords and permissions.

Tip: In *QA*Director, there are three levels of user access permissions: global, project, and suite.

Global permissions are set by the administrator on the system level, and they control a user's ability to create, to delete, or to modify projects, tools, tool domains, users, TM servers, TE servers, and scripts throughout *QA*Director.

Project permissions are set by users with administration rights or appropriate permissions, and they control actions on the project level. They control the user's ability to create, to delete, or to modify suites, library test assets, and template attributes for the specific project in which the permissions were set.

Suite permissions are controlled on the suite level, and they are assigned by the user who creates a suite. They determine if users can do the following:

Add or delete suite test assets

View and delete results

Execute tests

Change default folder options

Management Function	Permissions
Project Management	**Manage Projects**—create, delete, and modify projects. **Manage Suites—**create, delete, and modify suites, classes, and procedures**.** **Create Suite**—create new suites. **Change Non-Owner Suite Default Folder**—reassign default folders of other users. **Change Non-Owner Suite Permissions**—modify suite permissions of other users. **Create/Delete/Modify Suite**—create, delete, or modify their own suites. **Delete/Modify Non-Owner Suite**—delete or modify suites of other users. **Execute Tests**—run tests. **Manage Results**—delete or view results. **Delete Results**—delete their test results. **Delete Non-Owner Result**—delete results from other user tests. **View Non-Owner Result**—view results of other user tests. **Manage Test Assets** — create, delete, and modify **Create/Delete/Modify Test Assets**—create, delete, and change test assets. **Delete/Modify Non-Owner Test Assets**—delete or change test assets created by other users. **Manage Filters**—create, delete, modify, and view filters depending upon selected subpermissions. This permission is subdivided into: **Create/Delete/Modify Filter**—create, delete, and modify filters. **View Filters**—view filters. **Manage Template Attributes**—create, modify, and delete template attributes, which are custom fields on the Test Procedure dialog box. This permission are subdivided as follows: **Create/Delete/Modify Template Attributes**—create, modify, and delete template attributes **View Template Attributes**—view template attributes. **Manage Locks and Logs**—break locks and delete and view log files.

	Break Locks—break a test lock that was set by another user. **View Log Files**—view logs.
User Management	**Manage Users**— add, modify, or delete project users. **Manage Role Permissions**—modify permissions within roles.
QA Reports	**Manage Default Report Configurations**— modify the input parameters, display, and layout of *QA*Center Portal reports. **Project Content Management**— Add project specific external content, such as a static HTML report, or dynamic content identified by a URL. **Manage Logos**— Add, edit, or delete logos to display in report headers or footers.
DB Management	**Launch ODBC Manager**—open the ODBC Manager. **Upgrade DB**—upgrade the database.
Product Integrations	**Manage Product Integrations**— create, modify, or delete *QA*Center Portal project integrations with TrackRecord and Reconcile.
Integrations Management	**Create/Delete/Modify Tools**—create, delete, or modify tools. **Create/Delete/Modify ToolDomains**—create, delete, or modify tool domains. **View Tools**—view available tools. **View Tool Domains**—view available tool domains.
Test Machine Management	**Add TM Servers**— create a prioritized list of where to run the Test Management Server. **Change Execution Ports**—open and change specified ports for all executables. **Change TM Server Priority**—choose priority of Test Management machines. **Delete TM Servers**—remove Test Management machines from the priority list. **View TE Servers**—view Test Execution Machines from the System Administration Center. **View TM Servers**—view Test Management Machines from the System Administration Center.
Test Script Management	**Create/Delete/Modify Test Scripts**—create, delete, or modify a test scripts. **Delete/Modify Non-Owner Test** Scripts—delete or modify tests of other users. **View Test Scripts**—view test scripts.

Changing the Administrator Password

When *QA*Director is first installed, use the default **Admin** ID to log on to *QA*Director. Compuware recommends changing the default password for the **Admin** user so that unauthorized users cannot access higher level permissions.

To change the default password for the Admin user:

1. Start *QA*Director and open the **System Administration Center**.
2. Select the Admin user.
3. From the **Users** panel **Action** menu, choose **Properties**. The User Properties dialog box appears.
4. In the **Password** field, delete the old password and type the new password.
5. In the **Verify Password** field, type the new password again.
6. Click **OK**.

Managing Role Permissions

To add flexibility to permissions, add or remove permissions to specified roles throughout *QA*Director. Test-specific roles customized at the System Administration Center level will set the default role for all projects. To bypass the default settings, change user roles for a project in the Project Administration Center.

Note: After migrating from 04.05.01 or 05.00.00 to 05.00.01, new Teamleader and QAManager roles are **not** added as default users. Manually add these to the system when migrating. These roles are added by default only with a new installation of 05.00.01.

To customize role permissions:

1. From the **System Administration Center**, select **User Actions>Manage Role Permissions**. The Manage Role Permissions dialog box displays.
2. Select a role from the **Role** list.
3. Select or clear the check boxes according to the desired customized permissions.
4. Click **OK**.

Tip: Anything the user has not been given permission to access will be disabled on the menus in *QA*Director. This means the menu item or field will appear gray.

Datasources and Databases

Configuring a Datasource or Server at Login

A datasource configures *QA*Director as a stand-alone installation or connection. Datasources are only available for Win32. Datasources do not offer single sign-on capabilities. This means that a user must log on to each application separately.

A server configures *QA*Director as an Enterprise installation or connection. Select a server if using *QA*Center Portal. This feature provides single sign-on capabilities. This means that a user logs on once to access all applications.

To configure a datasource or server:

1. Click **Start>Programs>Compuware>*QA*Director** to log on to *QA*Director. The **Login** screen appears.
2. Click **Configure**. The Select a Database Dialog Box appears.

Refer to the *QADirector Installation and Configuration* guide for more information.

Note: If installing the **Test Management Server** only, configure the datasource by clicking **Start>Programs>Compuware>*QA*Director>Database Configuration.** The Select a Database Dialog Box appears.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Connecting to a Database

*QA*Director's architecture is based on a multi-user database. This database makes it possible for colleagues to work together to do such things as sharing suites, tests, and job results. The default *QA*Director database is an MSDE database and will be used if a different data source name is not used. However, you may want to connect to a different database. For example, if working with a team, you may need to connect to a database residing on a server machine.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To connect to a database:

Note: If connecting to an ODBC database, a database administrator must first set up the ODBC database using the procedure outlined in the *QADirector Installation and Configuration Guide*. The database administrator should also provide the data source name, database or schema name, and user name and password for the database.

1. Close *QA*Director, the test management server, and the test execution server if they are running.
2. Click the **System Administrator Center** icon on the **Centers** bar.
3. Click **Data Source Actions**>**Change Data Source**. The Select a Database dialog box appears.

This dialog box displays the data source name of the database currently in use. When first installing *QA*Director, the default data source is *QA*Director.

At this point, connect to either a data source or a *QA*Director server.

If selecting a data source, *QA*Director will act as a stand-alone installation or connection. Data source is only available for Win32 clients. This makes it possible to log into *QA*Director, and within a project, define integration information for TrackRecord and Reconcile.

Connecting to a *QA*Director data source requires users to log in multiple times for each application.

Creating a New Database

*QA*Director's architecture is based on a database. This multi-user database offers centralized control of system access rights and facilitates the sharing of test data. The default *QA*Director database is a sample MSDE database named *QA*Director. However, you can configure and use an Oracle or SQL Server database if desired. Refer to the *QADirector Installation and Configuration Guide* for more information.

Creating and Managing Projects in the Project Information Center

Projects are the highest level in the test hierarchy. Projects contain requirements, test suites, defects, product integration information, user information, and the test library. Manage projects in the **Project Information Center**.

The **Project Information Center** is divided into three sections:

The **Project** pane lets you create a project.

The **Project Statistics** pane lists statistics unique to the open project.

The **Project Users** pane lists names and roles for users unique to the open project.

Use the **Project Information Center** to create or edit projects, assign users to specific projects, and specify start date and end date for projects.

Since the tasks performed in the **Project Information Center** are administrative, only users with Administrative, Team Lead, and QA Manager roles can access this center.

Also, use the **Project Information Center** to integrate with TrackRecord, Request Management, or Reconcile.

About Creating and Managing Projects

Projects are the highest level in the test hierarchy. Projects contain requirements, test suites, defects, product integration information, user information, and the test library. Manage projects in the **Project Information Center**.

The **Project Information Center** is divided into three sections:

The **Project** pane lets you create a project.

The **Project Statistics** pane lists statistics unique to the open project.

The **Project Users** pane lists names and roles for users unique to the open project.

Use the **Project Information Center** to create or edit projects, assign users to specific projects, and specify start date and end date for projects.

Since the tasks performed in the **Project Information Center** are administrative, only users with Administrative, Team Lead, and QA Manager roles can access this center.

Also, use the **Project Information Center** to integrate with TrackRecord, Request Management, or Reconcile.

Creating a Project

Projects house testing information. Only project administrators (Team Leads, QA Managers), and anyone with Manage Projects permission can create a project.

To create a new project:

1. Click the **Project Information** icon on the **Centers** bar.
2. Click **Actions>New**. The New Project dialog box appears.

Note: If another project is open, *QA*Director prompts you to save it, then opens the New Project dialog box.

3. Type the project name in the **Project Name** field.

4. Select the standard template or a project on which to base the new project from the **Based on the following Project** list. When the standard template is selected a set of global templates is copied to the new project. When a current project is selected, the libraries, suites job results, users, and template attributes are copied to the new project.

5. Click **Create**.

6. In the **Project** panel, type a description in the **Description** field.

7. Click **Start Date Calendar**. The Calendar dialog box appears.

8. Select a date by clicking the forward or back arrows to the desired month, then clicking the date. After the date is selected, the window automatically closes and the date is saved to the **Start Date** field.

9. Click **End Date Calendar**. The Calendar dialog box appears.

10. Select a date by clicking the forward or back arrows to the desired month, then clicking the date. Once the date is selected, the window automatically closes and the date is saved to the **End Date** field.

11. Select the **Active** check box to make the project available to project users. If the project is not currently active, clear the check box. When not selected, only project administrators (Team Leads, QA Managers), and anyone with Manage Projects permission can access the project.

12. If using Reconcile with this project, select the **Reconcile** check box.

13. Select the Reconcile Integration name from the **Name** list, or click **Configuration (...)** to configuring a new integration. The server and database will populate.

14. Type the Reconcile project name in the **Project** field.

15. Click Defect Management , if using this feature and select TrackRecord or Request Managment.

16. Select the TrackRecord or Request Management Integration name from the **Name** list, or click **Configuration (...)** to configure new integration. The server and database will populate. If the integration uses the Web Server, this field will populate.

17. Type the Defect Management project name in the **Project** field for TrackRecord if TrackRecord is selected.

Opening a Project

Open a project to access requirements, test suites, defects, product integration information, user information, and the test library.

To open a project:

1. Click the **Project Information** icon on the **Centers** toolbar.

   Tip: *QA*Director stores session information and state. If you close *QA*Director and then reopen it, it will open to where you were when you closed the application.

2. Click **File>Open>Project** or if you already have a project open and wish to open a new one, click **Actions>Open**. The Open Project dialog box appears.

   Note: Only projects for which permissions are granted are viewable. All other projects are hidden.

3. Select a project from the **Project** list.

4. Click **Open**.

Deleting a Project

To delete a project:

1. From the **Project Information Center**, open the project to delete.
2. Click **Action>Delete**. A deletion confirmation message appears.

Note: Deleting the project will delete all the project-related information such as the Test Library.

3. Click **OK**.

Importing and Exporting Projects

As a test application moves from one release to the next, the associated project is likely to change as well. You may need to add, remove, or re-structure projects for each release of the application. Before making changes to the project, save a version of the project as it stands. By periodically saving versions of the project, it is possible to track the changes made to the project for each stage in the test application's development. If you need to reference the old information or return to a previous version of the project, the information can be easily retrieved.

To save a version of your project, export the project as a *QA*Director Exchange file. The file can reside on a local or network drive. Then, use any standard version-control software to manage the *QA*Director Exchange file. At any time, you can import the *QA*Director Exchange file back into *QA*Director to display the project. If the project already exists in *QA*Director (which is possible if you saved a version of the project and continued to work on it), *QA*Director creates a new project with the same name but appends a number to it. For example, if you import the *QA*Director Exchange file for a project named Demo and a Demo project already exists, *QA*Director creates a Demo_1 project.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Creating a Product Integration

While creating or editing a project within the **Project Information Center**, integrate a project with TrackRecord or Reconcile:

To create a project integration:

1. On the **Project** panel, select either the **TrackRecord** or **Reconcile** checkbox.
2. Click **Configuration (...)**, or click **Tools>Manage Product Integrations**. The Manage Product Integrations dialog box appears.
3. Choose a product with which to integrate.
4. Click the **New** icon on the toolbar. The integration details field on the right are enabled.
5. Type an integration name in the **Name** field.
6. Type a server path in the **Server** field.
7. Type a database name in the **Database** field.
8. For informational purposes, type a user name and password in the **Username** and **Password** fields.
9. Optionally type a Web Server path in the **Web Server** field.
10. Click the **Save** icon in the toolbar.
11. Click **OK** to close the dialog box.

Creating a Single Sign-On Environment

Single sign-on capabilities within a project provide ease of use among applications such as TrackRecord and Reconcile. With a single sign-on environment, open applications without logging on to each one.

To simulate a single sign on environment:

1. Open the **Project Information Center** or the **System Administration Center**.
2. Right-click a user in the **Project Users** panel and choose **Properties**. The User Properties dialog box appears.
3. Click the **Single Sign On** tab.
4. Select an integration from the **Available Product Integrations** list.
5. Type the user name for the specific application in the **User ID** field.
6. Type the password in the **Password** field.
7. Click **OK**.

Users and Permissions

After adding users to the project, customize their project permissions. Project permissions override the system role permissions for the specified project only.

To customize a project role:

1. From the **Project Users Actions** menu, select **Manage Users**. The Manage Users dialog box appears.
2. Select a user from the **Project Users** list. If there are no users in the list, Add one at this time.
3. Click **Permissions** to assign a project permission. The Project Permissions dialog box appears.
4. Select the desired permissions. Click **OK**.
5. Click **Apply** on the Manage Users dialog box to apply the changes and manage another project user, or click **OK** to save the changes and close the dialog box.

Managing Project Users

Users with Administrator or sufficient role permissions can manage *QA*Director users in a current project. In addition, Administrators, QA Managers, and Team Leaders can add users, assign roles, and customize permissions of a user.

Tip: Remember that there are three levels of permissions:

Project is set from the **Project Information Center** or the **System Administration Center** and affects only projects.

Global is set from the **System Administration Center** and affects all *QA*Director functions

Suite is set from the **Suite Information Center** and affects only suites.

To manage users in any capacity other than from within projects, refer to Assigning User Passwords and Permissions.

Managing Test Scripts in the Script Information Center

A test script can be an **automated script** that was captured using a record/playback testing tool such as *QA*Run, for example. A test script can also be, but is not limited to, a **manual test**, or an MVS batch job. To run a test script, you must first add the script to the Test Library and to a test procedure, and then run the test procedure.

Additionally, use *QA*Director to run a **user defined test script.** To run a User Defined *QA*D test script, include a fully qualified path in the string used to launch the test application, or ensure that the test application resides in the system path.

Use the **Script Information Center** to access test scripts after they have been created in third party tools. Additionally, use the **Script Information Center** to create and organize user defined and manual tests. Create categories to group scripts together. These categories can be copied to the **Test Library** for easy access.

About Test Scripts

A test script can be an **automated script** that was captured using a record/playback testing tool such as *QA*Run, for example. A test script can also be, but is not limited to, a **manual test**, or an MVS batch job. To run a test script, you must first add the script to the Test Library and to a test procedure, and then run the test procedure.

Additionally, use *QA*Director to run a **user defined test script.** To run a User Defined *QA*D test script, include a fully qualified path in the string used to launch the test application, or ensure that the test application resides in the system path.

Use the **Script Information Center** to access test scripts after they have been created in third party tools. Additionally, use the **Script Information Center** to create and organize user defined and manual tests. Create categories to group scripts together. These categories can be copied to the **Test Library** for easy access.

Creating Scripts

*QA*Director makes it possible to open the testing tool from within the Script Information Center to create test scripts.

To create test scripts:

1. Click the **Script Information Center** icon in the Centers toolbar. The Script Information Center appears.
2. Select a testing tool from the **Testing Tool** field.
3. Click **Action**>**New Script** or select **New Script** from the Right-click menu. The selected testing tool script editor opens.

Adding Test Scripts to Test Procedures

Each test procedure must contain at least one script. These scripts may be manual scripts, automated scripts, or other types of scripts. When running the test procedure, the scripts execute in the order they appear in the test procedure.

To add a script to a test procedure:

1. From the **Suite Information Center** or the **Test Library**, double-click a test procedure to which to add the script. The Test Procedure dialog box appears.
2. On the **Scripts** tab of the Test Procedure dialog box, click **Insert**. The Add Test Scripts From Library dialog box appears.
3. Select the type of test from the **Testing Tool** list. Only scripts which have been added to the library appear.
4. Click **Add** to add the script to the list.

Viewing Scripts

View the test scripts that have been created for each tool domain, or category.

To view scripts for a selected testing tool:

1. On the **Centers** toolbar click the **Script Information** icon. The Script Information Center appears.
2. Select a testing tool from the **Testing Tool** list. The scripts display by tool domain for automated testing tool and by category for manual tests and user defined scripts.
3. To change the view, either collapse unwanted tool domains and categories or select a filter from the **Filter** list.

Tip: Right click on the top node and choose **Show Hidden** to display all items. Choose **Hide** to hide them.

About Manual Testing

A manual test is a test that requires interaction from the tester, whether it be answering questions or performing some action. It is possible to use manual tests when automated testing is impractical or not yet implemented. Even an automated test plan is likely to include tasks that cannot be easily automated. In that case, use manual tests to complete these tasks. Manual tests also provide a way for Quality Assurance testers to comment on software interface and usability. With *QA*Director, it is possible to perform comprehensive manual testing to define the instructions and questions that the tester responds to during test execution.

There are two ways to run manual tests:

Standard Windows execution: The job runs on a computer where *QA*Director is installed. The job begins as soon as a computer is available.

Web Execution: The job is assigned to a specific tester who completes the job using a Web browser, not *QA*Director. The job begins when the tester starts it.

Creating a Manual Test Script

To create a manual test, design a series of questions and instructions via the Manual Test Editor. Use this to create a large number of tests at one time. Add these tests to procedures later if designing a test suite from the bottom up or add manual tests as needed as test procedures are defined.

When creating a test script add images, files, URLs, etc. as attachments in **Instruction** text fields.

To create a manual test script:

1. Click the **Script Information Center** icon in the Centers toolbar. The Script Information Center appears.

2. Click **Testing Tool**>**Manual**.
3. Click **Action**>**New Script** or select **New Script** from the Right-click menu. The Manual Test Editor appears.
4. Complete the applicable fields:

   Name: Type a name for the new manual test. The name must be unique across all categories and cannot contain the less than (<), greater than (>), at (@), pipe (|), dollar ($), ampersand (&), and caret (^) symbols.

   Category: Verify that the category is correct. If it is not correct, select another category in the list.

   Description: Optionally type a description, such as the purpose or a synopsis of the test.

5. The first step defaults to an **Instruction**. In the **Step Type** field, select a type:

   Instruction: Comments on the test or instructs the tester to perform some action. At execution time, pass or fail the test.

   Question(Yes/No): Asks a question and displays yes or no as possible answers. The test is marked passed or failed depending on whether or not the tester selects the correct answer.

   Question(True/False): Asks a question and displays true or false as possible answers. The test is marked passed or failed depending on whether or not the tester selects the correct answer.

   Question(Text Answer): Asks a question which does not have a defined list of responses. The tester responds and chooses whether the test passed or failed.

   Question(Multiple Choice): Asks a question and displays a list of possible answers, one of which is the correct answer. The test is marked passed or failed depending on whether or not the tester selects this answer.

6. Depending on the type of step selected, enter the appropriate information:

   Question(Yes/No): In the **Text** field, type the question. Yes and No appear in the **Choices** column. In the **Correct Answer** list, select the correct answer.

   Question(True/False): In the **Text** field, type the question. True and False appear in the **Choices** column. In the **Correct Answer** list, select the correct answer.

   Question(Text Answer): In the **Text** field, type the question. When the test is run, the tester responds to the question.

   Question(Multiple Choice): In the **Text** field, type the question. Type the choices separated by a delimiter. The default delimiter is a comma. See the Manual Test Editor Preferences Dialog Box to change the default. In the **Correct Answer** list, select the correct answer.

   Instruction: In the **Text** field, type the instructions or comments. This step requires no response from the tester.

9. Click **Insert New Step** , or tab through the column, to add another step and repeat the process.
10. When finished adding steps, click **Save.**

Reserved Symbols and Characters

The following symbols cannot be used in the names or descriptions of test scripts:

pipe (|)

dollar ($)

ampersand (&)

caret (^)

If a valid character is used that is not the defined separator, a message appears indicating what separator should be used.
If an invalid character is used, a message appears indicating that an invalid character is being used.

User-defined scripts can use the dollar ($) and ampersand (&) symbols, but cannot use the pipe (|) or caret (^) symbols.

Adding Test Scripts to the Library From the Script Information Center

After scripts are created in the Script Information Center, add them to the Test Library for use in the a project.

To add scripts to the library:

1. Click the **Script Information Center** icon in the Centers toolbar. The **Script Information Center** appears.
2. Select tool type from the **Tool Type** drop-down list.
3. Right-click on the script to add to the **Test Library.**
4. From the Right-click **Script** menu, select **Add to Test Library**. *QA*Director adds the selected scripts to the library.

Tip: To select more than one script in consecutive order, click on the first script, hold down the Shift key, then click the last script. To select several random scripts, click the first script, hold down the CTRL key, then select each individual script.

Rules

A rule is a command that prepares a test script for test execution, collects information during a test execution, determines if a test script failed, or cleans up after the test execution. Thus, rules enable separation of the actual test from pre-test and post-test commands.

The command can be any executable, or it is possible to use the commands provided with *QA*Director. These executables reside in \program files\Compuware*QA*Director.

Rules cannot be set up at the suite level. To use a rule throughout a suite, define rules at the project level. This will apply the rule to all suites within a project.

Defining Rules for Tests

After creating a test, define rules for the tests to separate the actual test from pre-test and post-test commands. These help prepare a test script for execution.

To define a rule for a test:

1. Choose the test class or test procedure to which to add the rule.
2. For example, add the rule to a test procedure if the rule is required by that specific test procedure only. Add the rule to a test class if the rule is required by multiple tests within the test class.
3. In the Suite Organization Tree, double-click the test class or procedure. The Test Class or Test Procedure dialog box appears.
4. Click the **Execution** tab and click **Rules**.
5. Click **Add**. The Rule dialog box appears.

6. Select the type of rule: **environment variable**, **setup command**, **pass/fail command**, or **cleanup command**.
7. Depending on the type of rule, complete the appropriate fields:

Environment Variables Fields

- **Name**: Enter the name of the environment variable.
- **Value**: Enter the value of the environmental variable.

Setup, Pass/Fail, and Cleanup Fields

- **Command**: Enter the path and name of the command to execute, along with any parameters such as path and name of a file to create, open, copy etc. Remember to type `Start` before an executable path and command.
- **Apply command to**: This option is disabled for test procedures. For test classes, choose how to apply the rule to descendants.
- **Bind to machine**: To execute the command on a specific machine, enter the machine name.
- **Timeout**: Enter the number of minutes for *QA*Director to wait after the command starts before ending the process.
- **Exit status**: Enter an exit status. This will require the command to have a specific exit status in order to be successful. Also, select the **Require Exit Status** option.
- **Require exit status**: Select this check box to require *QA*Director to mark that the command failed unless it produced a specific exit status.

9. Click **OK**.

Designing and Organizing Tests in the Suite Information Center and Test Library

After a project is created in the **Project Information Center**, design and organize tests that will reside in the project. A test is the combination of procedures, classes and scripts. This is done in the **Suite Information Center**.

Typically, a test suite is created. A test suite contains all the tests applicable to one application or one component of an application. A test suite is comprised of test classes and test procedures, otherwise known as test assets. These descend from a single test class called the root class. The root class has the same name as the test suite and cannot be deleted. All other test assets descend from the root class.

Test suites perform the following functions:

Define which tests are necessary

Group the tests logically

Designate the order in which the tests should be run

For example, you may create one test suite for the entire application, with separate unit testing and system testing. To accomplish this create a single test suite that contains two high-level test classes, one for unit tests and one for system tests.

To access test scripts after they have been created with third party tools, use the **Script Information Center**. Use the **Script Information Center** also, to create and organize user defined and manual tests.

The **Test Library** is a repository for all available tests within a project. It contains test classes, class templates, test procedures, and test scripts. The **Test Library** makes it possible to use these tests in different suites within a project. As tests are used across suites, changes are saved to the **Test Library** and all suites.

About Designing and Organizing Tests

After a project is created in the **Project Information Center**, design and organize tests that will reside in the project. A test is the combination of procedures, classes and scripts. This is done in the **Suite Information Center**.

Typically, a test suite is created. A test suite contains all the tests applicable to one application or one component of an application. A test suite is comprised of test classes and test procedures, otherwise known as test assets. These descend from a single test class called the root class. The root class has the same name as the test suite and cannot be deleted. All other test assets descend from the root class.

Test suites perform the following functions:

Define which tests are necessary

Group the tests logically

Designate the order in which the tests should be run

For example, you may create one test suite for the entire application, with separate unit testing and system testing. To accomplish this create a single test suite that contains two high-level test classes, one for unit tests and one for system tests.

To access test scripts after they have been created with third party tools, use the **Script Information Center**. Use the **Script Information Center** also, to create and organize user defined and manual tests.

The **Test Library** is a repository for all available tests within a project. It contains test classes, class templates, test procedures, and test scripts. The **Test Library** makes it possible to use these tests in different suites within a project. As tests are used across suites, changes are saved to the **Test Library** and all suites.

Copying, Duplicating, and Moving a Test

After tests are created move and copy them from one class to another within the same suite or from one suite to another within the same project. It is also possible to copy from the **Test Library** to a suite.

When copying a class or procedure to a suite the entire hierarchy is included, but works independently of the source. Whereas, when copying a Class Template to the suite, the hierarchy is bound to the Test Library.

Duplicating is the same as copying, however, the test is pasted directly under the test that was duplicated. The duplicate has the same parent as its source.

To copy, duplicate, or move tests, use the right-click menu or drag and drop.

Changing a Test Name

It may be necessary to change a test name if, for example, it was a duplicate test and you want it to have a unique name.

To change the name of a test:

1. Right-click on a test class, test procedure, or test script and choose **Rename** from the menu.
2. Type the new name.
3. Press **Enter**.

Note: It is not necessary to open the file to rename it.

Locking, Saving and Publishing a Test

*QA*Director's architecture is based on a repository, or database. This multi-user database offers centralized control of system access rights and facilitates the sharing of test data. To ensure data security in this multi-user environment, *QA*Director employs an effective locking mechanism that prevents users from overwriting each other's work.

To allow users to design and modify tests before sharing their changes with others, *QA*Director provides separate commands for saving and publishing tests in the database.

Viewing a Test Association

To view associations between tests (classes, procedures, scripts, and suites), right-click a test in the Test Library and choose View Test Association from the menu. This displays all the tests one level above the item selected, that use the item.

For example, if you right-click procedure "Abc," and click **View Test Association** from the menu, the dialog box displays classes and suites associated with "Abc" since these items are one level or more above the procedure level in the test hierarchy. Any scripts associated with "Abc" will not display since scripts are one level below procedures.

To view test associations within a project:

1. Open the Test Library.
2. Right-click on a test procedure, test class, or test script.
3. Choose **View Test Association** from the menu. The Test Association dialog box appears.

Associating a File with a Test

Some tests require additional files in order to execute properly. For example, suppose you have created a simple user-defined script that opens a text file in Notepad. Because the text file resides on your computer, you can successfully execute the test. The test, however, cannot be executed on another computer because the text file does not exist on that computer. To solve this problem, associate the text file with the test procedure so that the text file is saved in the database. When the test is executed by another user, the text file is copied to that person's computer.

Determine whether to associate the file with the test procedure or with the parent class. If several test procedures in the same test class use the file, it is best to associate the file at the test class level.

Tip: If several test procedures in the same test class use the file, it is convenient to associate the file at the test class level, rather than associate the same file with each test procedure. However, if using this method, the test procedures must include the fully qualified path to the file. Type the path in the Test Scripts Manager when creating user-defined scripts.

Note: It is possible to leave the file in its original folder and not copy it to a common directory. When associating a file within the original folder, the entire path to the file will be saved in the database. Therefore, ensure that the path is logical on all other team member's computers. For example, if the file resides on a mapped drive, all team members must map to the drive using the same letter.

Synchronizing an Associated File

*QA*Director provides a method for synchronizing the associated files saved in the database with the associated files in the test's working directory or other local directory. It is possible to update the file on the database with a local file, or update the local file with the file on the database. It is good practice to synchronize a test before modifying, executing, or closing it. This ensures that the most recent files are on the hard drive and updated files are saved.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To synchronize an associated file:

1. Choose a suite.
2. Click **Action Menu>Synchronize Associated Files**. The Synchronize Associated Files dialog box appears.
3. Choose the file to synchronize.
4. Click **OK** to synchronize the files according to these settings.

Accessing a Test Property

Two ways to access test properties are:

Double-click the test class, test procedure, or test script in the **Suite Information Center** or the **Test Library**.

Select the test class, test procedure, or test script from the suite or **Test Library**. In the **Suite Information Center**, click **Action**>**Properties**. In the **Test Library,** choose **Properties** from the right-click menu. The Test Procedure dialog box, or Properties dialog box appears.

Test Suites

Tests are organized in test suites. Typically, a test suite contains all the tests applicable to one application or one component of an application. Test suites create order and control by defining which tests are needed, grouping the tests logically, and designating the order in which the tests should be run.

A test suite is comprised of test classes and test procedures, all of which descend from a single test class called the root class. The root class has the same name as the test suite and cannot be deleted. All other test classes and test procedures descend from the root class. The **Suite Information Center** contains all of the test assets (classes and procedures) necessary to run scripts and complete a test. It contains four panels:

Suite Lists: Displays all suites available for the current project.

Suite Organization: Displays the organization tree.

Statistics: Displays test information and job statuses.

Properties: Displays filter information and a summary of the selected suite which is originally typed in the Test Procedure Dialog Box.

It is possible to work with assets selected from the **Suite Information Center** or the **Test Library** as long as the assets are attached to both the center and the library. Detached assets can only be accessed in the **Suite Information Center**.

Creating a Suite

In *QA*Director, tests are organized in test suites. Typically, a test suite contains all the tests necessary for one application or one component of an application. Test suites create order and control by defining which tests are needed, grouping the tests logically, and designating the order in which the tests should be run.

A test suite is comprised of test classes and test procedures, all of which descend from a single test class called the root class. The root class has the same name as the test suite and cannot be deleted. All other test classes and test procedures descend from the root class.

Enter information for the suite into the Test Class dialog box.

To create a test suite:

1. Open or create a project.
2. From the Centers toolbar, click the **Suite Information** icon.
3. Click **Actions**>**New Suite**. A new suite is created with a default name.
4. Click **OK**. The new test suite displays in the **Suite List** and the root class appears in the **Suite Organization Tree**.

Note: When creating a test suite, *QA*Director automatically creates a single root class and gives the root class the same name as the new suite. All other test classes and test procedures descend from the root class.

5. Double-click the root class to open the Test Class dialog box.
6. Complete the applicable fields on the Test Class dialog box.

Copying a Test Suite

To save time, copy a test suite when using the same basic test suite structure or when using the same test classes/procedures.

To copy a test suite:

1. From the **Suite Information Center** choose the existing test suite to copy.
2. Click **Action**>**Copy Suite**.
3. Click **OK**.

Note: When copying a test suite, *QA*Director automatically assigns unique ID numbers to the tests and their descendants. The copy of the test suite retains its classes, procedures, and scripts. The copy also retains test names and settings as well as their own environment variable, setup, cleanup, and pass/fail rules.

Deleting a Test Suite

When a test suite is no longer necessary delete it.

To delete a test suite:

1. Open the **Suite Information Center**.
2. Select a suite in the **Suite List**.
3. Click **Action>Delete Suite**. Or click **File**>**Edit**>**Delete**. The Confirm Delete dialog box appears.
4. Click **OK**.

Opening a Test Suite

The test suite contains all the tests for one application or one component of an application.

To open a test suite:

1. Open the Suite Information Center.
2. Select a test suite from the list. It appears in the Suite Organization pane.
3. With administrative privileges, view the entire suite by selecting the **Administrative View** check box.
4. In the administrative view, unpublished and locked test icons appear in red. Also within the administrative view, users with administrative privileges can delete tests that they do not own.

Note: This field is disabled if administrative privileges are not granted.

9. Click **OK**.

   If the owner has not assigned this permission, a message will appear.

Exporting a Suite to a *QA*D Exchange File

Periodically saving versions of the test suite allows for tracking the changes made to the test suite for each stage in the test application's development. To save a version of the test suite, simply export the test suite as a *QA*DExchange file.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To export a suite to a QADExchange file:

1. Open the **Suite Information Center**. In the Suite List, click on a suite.
2. From the **Action** menu, choose **Export**.
3. In the **Export** field, enter the folder and file to which you want to export the test suite. The file must have a **.qad** extension.
4. Select the information to include in the export:

Suite information: Exports the names, properties, and structure of all the tests in the test suite. Note that this option does not export the actual scripts.

All results: Exports **all** job results for the test suite. This includes jobs run by all users, not just the current user.

Manual tests: Exports manual test scripts associated with the test suite.

Manual test results: Although the overall pass/fail result of manual tests are included when you select **All Results**, this option goes further to export detailed manual test results such as the responses entered by the tester for each step in the manual test.

5. Click **OK**.
6. After the export is complete, you can use any standard version-control software to manage the QADExchange file.

Importing a *QA*DExchange File

After exporting a test suite as a *QA*Director Exchange file, import the *QA*DExchange file back into *QA*Director to display the test suite.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To import a *QA*DExchange file:

1. In *QA*Director, close any project that may be open.
2. From the **File** menu, click **Import>5.0 or greater versions of Project Exchange File,** or click **Actions>Import 5.0 or greater versions of Project Exchange File** from the Project pane of the **Project Information Center**, and from the **Suite Information Center**.
3. Select the **.qad** file you want to import.
4. Click **Open**.

When the import of a QAD 5.0 file is done, and a QAD 5.1 project of the same name exists, a number is appended to the project name. The suite within the project retains the original name.

MSWord/Excel Import Wizard for Suites

The Import Wizard helps to import existing testing documents into *QA*Director. Supported document types are Microsoft Excel spreadsheets and Word documents. The Import Wizard walks you through the process of importing one of these document types. The following fields and buttons are available on this wizard:

Field Name	Description
Select File	Click the Browse button to find the file to be imported.
Create Suite	Type the test suite name that will contain the imported document.
Select Class	Choose a style tag (Word) or a column that contains the class (Excel) to represent a Test Class in *QA*Director.
Select Procedure	Choose a style tag (Word) or a column that contains the procedure (Excel) to represent the test procedure.
Select Test Script	Assign style tags to import manual test scripts.
Style Tag option	Choose to assign a style tag from the list.

Ignore option	Choose to ignore style tags that are not applicable.
Select Attributes	Assign any remaining style tags. These can be attributes or ignored.
Attribute option	Choose to assign an attribute from the list.
Ignore option	Choose to ignore attributes that are not applicable.
Finish	Ends the information collecting process
Command Buttons	
Help button	Click to access dialog level help.
Cancel button	Click to close the dialog box without saving any changes.
Back button	Click to go to the previous step.
Next button	Click to go to the next step.
Finish button	Click to import the document.

<<**Previous:** Importing a QAD Exchange File | **Next>>:** Exporting a Suite to a QAD Exchange File

Renaming a Test Suite

To rename a test suite:

1. Open the Suite Information Center or the suite list.
2. Choose the test suite to rename.
3. Right click and choose **Rename**.
4. Type the new name for the test suite.
5. Click **OK**.

Exporting Attributes into a Text File

Use this feature to export some or all of the test attributes in a suite into a text file. This information may be useful for creating custom reports with third-party reporting tools.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To export test attributes to a text file:

1. Open the **Suite Information Center**.
2. Select the test suite.
3. Click **File**>**Export attributes into Text File**. The Save as dialog box appears.

4. Navigate to the directory to save the text file, type a name for the file, and click **Save**.
5. On the Select Attributes to Export dialog box, select the test attributes to export. Click **Select All** to select all of the attributes in the list.
6. Click **OK**.

Version Management

As the test application moves from one release to the next, the associated test suite is likely to change as well. It may be necessary to add, to remove or to re-structure tests for each release of the application. Before making changes to the test suite, save a current version of the test suite. Periodically saving versions of the test suite allows for tracking the changes made to the test suite for each stage in the test application's development. To reference the old information or return to a previous version of the test suite, the information can be easily retrieved.

To save a version of the test suite, export the test suite as a *QA*Director Exchange file. The file can reside on any local or network drive. Use any standard version-control software to manage the QADExchange file. At any time, import the QADExchange file back into *QA*Director to display the test suite, if the test suite already exists in *QA*Director (which is likely if the version of the suite was saved and work on it continued). *QA*Director creates a new suite with the same name but appends a number to it. For example, if importing the QADExchange file for a test suite named Demo and a Demo test suite already exists, *QA*Director creates a Demo_1 test suite.

About Test Assets

A test suite is comprised of test classes and test procedures, otherwise known as test assets. These descend from a single test class called the root class. The root class has the same name as the test suite and cannot be deleted. All other test classes and test procedures descend from the root class.

Attaching an Asset

Attached and detached describe assets and their relationship to the **Test Library**. An asset (class or procedure) is attached if it belongs to the **Test Library**. It is available for different projects, suites, etc. An asset is detached if it does not belong to the test library. It is used only in one capacity. Tests can be attached and detached at the root node level only. An attached parent can't have a detached child.

When an asset is attached, the following icon is associated with it: 

To attach an asset:

1. Right-click the asset.
2. Select **Attach to Library**.
3. The asset appears in the **Test Library** panel.

Detaching an Asset

When an asset is detached, the original remains in the library, and a copy of the asset appears in the tree followed by a number in parenthesis. For example, Asset A (1). If the detached asset is detached again, the asset appears in the tree followed by a series of two numbers in parenthesis. For example, Asset A (1) (1) and so on.

When an asset is detached the icon changes from attached to detached: 

Note: Tests can be attached and detached at the root node level only. The detached asset cannot be reattached to the original. An attached parent cannot have a detached child.

To detach an asset:

1. Right-click the asset.
2. Select **Detach from Library.**
3. The following message appears:

   ```
   Separate test will be created with new name. Detached test cannot be attached
   back to original test. Do you want to continue with detach?
   ```

4. Click **Yes.**
5. The detached asset appears followed by *(1)*.

Modifying Attach and Detach Asset Default

*QA*Director attaches each asset by default. It is possible to change the default to detach (or vice versa).

To modify the Attach and Detach default:

1. Click **Tools>User options.**
2. Select the **Test Information** tab.
3. Select or clear the **Attach test to library** check box.
4. Click **Apply** to apply the changes.
5. Click **OK** to save the changes.

Procedures

Test procedures contain the scripts that run against test applications. They check the behavior of the test application and report on the result. When creating a procedure from the suite, *QA*Director adds the procedure to the Test Library first, then adds it to the suite.

To create a test procedure:

1. From the **Suite Information Center** or the **Test Library**, right-click the parent of the new test procedure. This may be the root if there are no other procedures.
2. From the right-click menu, choose **New Procedure.** A new procedure is created with a default name, if **in-place naming** is selected from the User Options dialog box . If this option has not been selected, the New Test Procedure dialog box appears.
3. Type a name for the new test procedure.
4. If using the New Test Procedure dialog box, click **OK**. The new test procedure appears in the Suite Organization tree-view and the library. It is automatically attached or detached depending on the parent class and options selected in the User Options dialog box. *QA*Director attaches the class by default.

5. Double click the procedure. The Test Procedure Properties dialog box appears. Complete the appropriate fields. Click **OK**.
6. To quickly move existing test procedures into the new test class, use **Cut** and **Paste** on the right-click menu.

Creating a Test Procedure

Test procedures contain the scripts that run against test applications. They check the behavior of the test application and report on the result. When creating a procedure from the suite, *QA*Director adds the procedure to the Test Library first, then adds it to the suite.

To create a test procedure:

1. From the **Suite Information Center** or the **Test Library**, right-click the parent of the new test procedure. This may be the root if there are no other procedures.
2. From the right-click menu, choose **New Procedure.** A new procedure is created with a default name, if **in-place naming** is selected from the User Options dialog box . If this option has not been selected, the New Test Procedure dialog box appears.
3. Type a name for the new test procedure.
4. If using the New Test Procedure dialog box, click **OK**. The new test procedure appears in the Suite Organization tree-view and the library. It is automatically attached or detached depending on the parent class and options selected in the User Options dialog box. *QA*Director attaches the class by default.
5. Double click the procedure. The Test Procedure Properties dialog box appears. Complete the appropriate fields. Click **OK**.
6. To quickly move existing test procedures into the new test class, use **Cut** and **Paste** on the right-click menu.

Adding Test Scripts to Test Procedures

Each test procedure must contain at least one script. These scripts may be manual scripts, automated scripts, or other types of scripts. When running the test procedure, the scripts execute in the order they appear in the test procedure.

To add a script to a test procedure:

1. From the **Suite Information Center** or the **Test Library**, double-click a test procedure to which to add the script. The Test Procedure dialog box appears.

2. On the **Scripts** tab of the Test Procedure dialog box, click **Insert**. The Add Test Scripts From Library dialog box appears.

3. Select the type of test from the **Testing Tool** list. Only scripts which have been added to the library appear.
4. Click **Add** to add the script to the list.

Classes

Creating a Test Class

When creating a class in a suite, *QA*Director adds the class to the Test Library and the suite. It is possible to detach the class from the library by selecting **Detach from Library** from the right-click menu.

To create a test class:

1. From the **Suite Information Center** or the **Test Library**, right-click the parent class of the new test class. This may be the root class of the test suite if there are no other test classes.
2. From the right-click menu, choose **New Class.** A new class is created with a default name, if **in-place naming** is selected from the **User Options** dialog box . If this option has not been selected, the **New Test Class** dialog box appears.
3. Enter a name for the new test class.
4. If using the New Test Class dialog box, click **OK**.

6. The new test class appears in the **Suite Organization** tree view and the library. It is automatically attached or detached depending on the parent class and options selected in the **User Options** dialog box. *QA*Director attaches the class by default.

7. Complete the appropriate fields on the **New Test Class** dialog box.

8. Quickly move existing test procedures or classes into the new test class by dragging and dropping them into the new test class. To add a new test procedure, see Creating a test procedure .

Creating a Test Class

When creating a class in a suite, *QA*Director adds the class to the Test Library and the suite. It is possible to detach the class from the library by selecting **Detach from Library** from the right-click menu.

To create a test class:

1. From the **Suite Information Center** or the **Test Library**, right-click the parent class of the new test class. This may be the root class of the test suite if there are no other test classes.
2. From the right-click menu, choose **New Class.** A new class is created with a default name, if **in-place naming** is selected from the **User Options** dialog box . If this option has not been selected, the **New Test Class** dialog box appears.
3. Enter a name for the new test class.
4. If using the New Test Class dialog box, click **OK**.

6. The new test class appears in the **Suite Organization** tree view and the library. It is automatically attached or detached depending on the parent class and options selected in the **User Options** dialog box. *QA*Director attaches the class by default.

7. Complete the appropriate fields on the **New Test Class** dialog box.

8. Quickly move existing test procedures or classes into the new test class by dragging and dropping them into the new test class. To add a new test procedure, see Creating a test procedure .

Test Class Organization

A test class is a logical grouping of test procedures and other test classes within the test suite. You determine the logic behind each test class grouping. To determine the necessary test classes, first consider the preferred way to test the application:

Testing function or structure: Will the testing be primarily functional from the point of view of the user or structural from the point of view of the implementation or both? Do you want to separate functional from structural tests? If so, create two test classes at the highest level of the test suite, corresponding to functional and structural tests.

Testing units or systems: Will the tests follow a progression from unit/component testing to system testing? If so, do you want to separate unit/component from system tests? Create two test classes at the highest level of the test suite, corresponding to unit and system tests.

Grouping features together: Does the application have many features that belong together or that depend on each other? Do you want to organize tests according to these aspects of the application? Create classes at the highest level of the test suite to cover the major features of the application and create child test classes to cover subfeatures. Create as many levels of classes as needed.

About Class Templates

Class templates are classes arranged in a permanent structure. They retain class structures across suites. Whereas, classes available from the classes folder in the **Test Library** are not confined to structures and can be used independently of each other. Use class templates to enforce business rules. In addition, classes used from the classes folder in the **Test Library** will act independently of the source. Conversely, class templates enforce the hierarchy in the source and in each suite. In each suite, it is possible to combine both class templates and independent classes.

When working with class templates, remember:

A class can exist only once as a root node in the Test Library. It can exist as a child in multiple locations.

Drag, drop, or copy the root node of the class template into the suite. It is not possible to drag, drop, or copy class template children into a suite. However, it is possible to drag, drop, or copy any part of a class template to another class template.

Adding a Class Template to the Test Library

Create a class structure in the Test Library by right-clicking on the class template folder and selecting **New Test Class** from the menu. Repeat for each class and subclass to continue building the class structure or perform the following steps.

To add a class template to the Test Library:

1. Open the **Suite Information Center**.
2. Create a suite or open an existing suite.
3. Create a class structure or select an existing class structure.
4. Right-click on the top class node and select **Convert to Class Template**.

Categories

Create categories to help organize manual and user-defined scripts.

To create a category for a manual test script and user defined script:

1. Click the **Test Script Information** icon on the **Centers** toolbar. The Script Information Center appears.
2. Click **Testing Tool**>**Manual Test** or **Testing Tool**>**User Defined**.
3. Click **Action>Manage Categories**. The Manage Categories dialog box appears.
4. Click **New.**
5. In the **Category Name** field, type a unique name.

6. In the **Description** field, type a description.
7. Click **OK**.
8. To add existing scripts to the category, select scripts and drag and drop them into the desired category.

Note: To add existing manual tests to a category while creating new manual tests, select the category from the **Category** list in the Manual Test Editor dialog box.

Editing a Category

Edit categories to help organize manual and user-defined scripts.

To edit a category for Manual and User-Defined tools:

1. Click the **Test Script Information** icon in the Centers toolbar. The Script Information Center appears.
2. Click **Testing Tool**>**Manual** or **Testing Tool**>**User Defined**.
3. Click **Action> Manage Categories**. The Manage Categories dialog box appears.
4. Choose the category to be edited.
5. Click **Edit.** The Edit Category Dialog Box appears.
6. Edit the name in the **Category Name** field.
7. Edit the description in the **Description** field.
8. Click **OK**.

Deleting a Category

Delete categories, if necessary, when they are no longer used.

To delete a category for a manual test script or a user-defined script:

1. Click the **Test Script Information** icon on the **Centers** toolbar. The Script Information Center appears.
2. Click **Testing Tool**>**Manual** or **Testing Tool**>**User Defined**.
3. Click **Action> Manage Categories**. The Manage Categories dialog box appears.
4. Select category to delete.
5. Click **Delete.** If the category contains one or more scripts, click **Yes** to delete the scripts with the category. Click **No** to delete the category and move the scripts to **Not Categorized**.
6. Click **OK**.

Attributes

Test attributes are name-value pairs that store information about each test class and test procedure in a test suite. The value of a test attribute is specific to the test and cannot be passed down to descendants. Most test attributes describe both test classes and procedures. Some, however, are unique to classes or procedures to reflect the differences between classes and procedures.

Availability

All test attributes are available dynamically during execution with the qc_get_attr.exe command. Some attributes are available as environment variables while the test is running because *QA*Director automatically sets them to environment variables before execution. If an attribute is available as an environment variable, it is noted in the Value column.

Test Attribute	Value
QC_CHILDREN_IDS	The QC_IDs of the descendants of the class. Set automatically when you add/delete descendants of a class. You cannot change the value of this attribute directly. Node: Classes Available as environment variable while test runs
QC_CLEANUP*n*	A cleanup rule, numbered in the order it appears in the rule window, starting from 1. Set automatically when you add/edit/delete cleanup commands in rule window. Command-line changes to clean up rules work differently. See qc_ch_proc and qc_ch_class. Node: Classes and Procedures
QC_CLEANUP*n*_APPLY	Where the cleanup command numbered n is applied: local, classes, procedures, all, as entered in the cleanup rule definition dialog. Default is local. Set automatically when you add/edit/delete cleanup commands or change their scope. Command-line changes to cleanup rules work differently. See qc_ch_proc and qc_ch_class. Node: Classes
QC_CLEANUP*n*_TIMEOUT	Timeout of the cleanup command numbered n as entered in the cleanup rule dialog. Set automatically when you change timeout of cleanup command numbered n. Command-line changes to cleanup rules work differently. See qc_ch_proc and qc_ch_class. Node: Classes and Procedures
QC_CREATED_BY	User who created the test. Set automatically at test creation. Cannot be changed. Node: Classes and Procedures Available as an environment variable while test runs
QC_CREATED_ON	Date and time the test was created. Set automatically at test creation. Cannot be changed. Node: Classes and Procedures Available as an environment variable while test runs
QC_CYCLES	Number of repetitions to run a test between 1 and 20. This is commonly used with *QA*Center Portal. Node: Classes and Procedures

	Note: Can not be deleted or modified
QC_DEFECT_ID	ID number of a defect in a defect bug tracking system. Set automatically when you associate or disassociate a defect number with a test. Node: Classes and Procedures Available as an environment variablel while test runs
QC_EXP_RESULT	The expected outcome when the test procedure runs: pass, fail, or Not Executed. The default is pass. Set automatically when you change the expected test outcome. Node: Procedures Available as an environment variable while test runs
QC_HAS_DIR	Specifies whether the test procedure has its own directory: 1 if test procedure has its own directory; 0 if test procedure does not have its own directory. Set automatically at test creation. Cannot be changed. Node: Procedures Available as an environment variable while test runs
QC_ID	ID number of the test. Set automatically at test creation. Cannot be changed. Node: Classes and Procedures Available as environment variable while test runs
QC_IS_PROC	Specifies whether the test is a procedure. Set automatically at test creation. Cannot be changed. 1 if test is procedure; 0 if it is class. Node: Classes and Procedures Available as environment variable while test runs
QC_IS_CLASS	Specifies whether the test is a class. 1 if test is class; 0 if it is procedure. Set automatically at test creation. Cannot be changed. Node: Classes and Procedures Available as environment variable while test runs
QC_KEYWORDS	Keywords by which to find the test in searches. Set automatically when you add/edit/delete keywords. Node: Classes and Procedures Available as environment variable while test runs
QC_LOCK_DIR	Specifies whether class directory and its descendant directories are locked during execution to prevent other users from running the class. 1 if locked; 0 if

	not locked. Default is 0. Set automatically when you specify directories to be locked/unlocked during execution. Node: Classes Available as environment variable while test runs
QC_MANUAL	Specifies whether the test procedure is manual or automated. 0 if automated; 1 if manual. Default is 0. Set automatically when you specify that a procedure is manual or automatic. Node: Procedures Available as environment variable while test runs
QC_NAME	Name of test. Set automatically when you create/rename a test. Node: Classes and Procedures Available as environment variable while test runs
QC_NCHILDREN	Number of children the class has. Set automatically when you add/delete descendants of a test class. Node: Classes Available as environment variable while test runs
QC_NCLEANUPS	Number of cleanup rules defined in the test. Set automatically when you add/delete cleanup commands. Command-line changes to cleanup rules work differently. See qc_ch_proc and qc_ch_class. Node: Classes and Procedures
QC_NPASSFAILS	Number of pass/fail rules defined in the test. Set automatically when you add/delete pass/fail commands. Command-line changes to pass/fail rules work differently. See qc_ch_proc and qc_ch_class. Node: Classes and Procedures
QC_NSETUPS	Number of setup rules defined in the test. Set automatically when you add/delete pass/fail commands. Command-line changes to setup rules work differently. See qc_ch_proc and qc_ch_class. Node: Classes and Procedures
QC_PARENT_ID	ID number of the parent of the test. Set automatically when you create/move test.

	Node: Classes and Procedures Available as environment variable while test runs
QC_PASSFAIL*n*	Pass/fail commands defined in the test numbered in the order they appear in the pass/fail rule window, starting from 1. Set automatically when you add/edit/delete pass/fail commands in a test. Command-line changes to pass/fail rules work differently. See qc_ch_proc and qc_ch_class. Node: Classes and Procedures
QC_PASSFAIL*n*_APPLY	Where the pass/fail commands are applied. In a class, the value must be procedures. Set automatically when you add/edit/delete pass/fail commands. Command-line changes to pass/fail rules work differently. See qc_ch_proc and qc_ch_class. Node: Classes
QC_PASSFAIL*n*_TIMEOUT	Timeout of the pass/fail command numbered n, as entered in the pass/fail rule dialog box. Set automatically when you change the timeout of pass/fail command numbered n. Command-line changes to pass/fail rules work differently. See qc_ch_proc and qc_ch_class. Node: Classes and Procedures
QC_PATH	Absolute path to the test. Set automatically at test creation. Cannot be changed. Node: Classes and Procedures Available as environment variable while test runs
QC_PERMS_OWNER	Permissions of the test owner in the form rwx if owner can read, write, and execute the test, or rw - if owner can read and write the test but cannot execute it. You cannot remove owner's read and write permissions. The default is to give the owner read, write, and execute permissions at test creation. Set automatically when the owner changes the owner's permission. Node: Classes and Procedures Available as environment variable while test runs
QC_PERMS_PUBLIC	Permissions of users who are not the owner of the test in the form rwx if the public can read, write, and execute the test. The default is to give the public no permissions. Set automatically when you change the public's

	permissions. Node: Classes and Procedures Available as environment variable while test runs
QC_PLATFORMS	Platforms you designate on which the test may run. Set automatically when platform is changed. Node: Procedures Available as environment variable while test runs
QC_PURPOSE	Purpose of the test. Set automatically when you add/edit/delete the purpose of the test. Node: Classes and Procedures Available as environment variable while test runs
QC_RECORDER	UNIX record-playback tool used to record the test procedure. Set automatically when you record the script. Cannot be changed. Node: UNIX Procedures Available as environment variable while test runs
QC_REL_PATH	Relative path from the root node of the test suite to the test. Set automatically at test creation. Cannot be changed. Node: Classes and Procedures Available as environment variable while test runs
QC_REQUIREMENT	The requirement this test fulfills. May be a cross-reference to a functional specification. Set automatically when you associate/disassociate a requirement with a test. Node: Classes and Procedures Available as environment variable while test runs
QC_RISKS	Specifies risk type of test: low, medium or high. This is commonly used with *QA*Center Portal. Node: Classes and Procedures Note: Can not be deleted or modified
QC_RUN_IN_PARALLEL	Specifies which child tests should run in parallel: all, classes, procedures, or none (run children serially). Default is classes. Set automatically when you change the child tests that should run in parallel. Node: Classes Available as environment variable while test runs

QC_RUN_TIMEOUT	Timeout of the script. Set automatically when you change the timeout of the script. Node: Procedures
QC_SCRIPT_ALL_INFO	Names, descriptions, tool names, and command lines of all scripts in a procedure. Set automatically as you add/delete/change scripts. Node: Procedures Available as environment variable while test runs
QC_SCRIPT_ALL_NAMES	Names of all scripts in a procedure. Set automatically as you add/delete/change scripts. Node: Procedures Available as environment variable while test runs
QC_SETUP*n*	Setup commands defined in the numbered test in the order they appear in the setup rule definition dialog, starting from 1. Set automatically when you add/edit/delete setup commands in a test. Command-line changes to setup rules work differently. See qc_ch_proc and qc_ch_class. Node: Classes and Procedures
QC_SETUP*n*_APPLY	Location where the setup command numbered n is applied: local, classes, procedures, all, as entered in the setup rule definition dialog. Default is local. Set automatically when you add/edit/delete setup commands or change their scope. Command-line changes to setup rules work differently. See qc_ch_proc and qc_ch_class. Node: Classes
QC_SETUP*n*_TIMEOUT	Timeout of the setup command numbered n as entered in the cleanup rule dialog. Set automatically when you change the timeout of setup command numbered n. Command-line changes to setup rules work differently. See qc_ch_proc and qc_ch_class. Node: Classes and Procedures
QC_STATE	One of the following state-of-test values: Unwritten, In Progress, Completed, or Broken. Default is Unwritten. Set automatically when you change the state of the test. Node: Classes and Procedures Available as environment variable while test runs

QC_SUITE_ID	ID number of the test suite.Set automatically. Cannot be changed. Node: Classes and Procedures Available as environment variable while test runs
QC_SUITE_NAME	Name of the test suite. Set automatically when you create/rename the test suite. Node: Classes and Procedures Available as environment variable while test runs
QC_SUITE_PATH	Absolute path to the test suite. Set automatically. Cannot be changed. Node: Classes and Procedures Available as environment variable while test runs
QC_SUITE_DESCR	Description of the test suite. Set automatically if you add/edit/delete the test suite description. Node: Classes and Procedures
QC_SUITE_PROJECT	The name of the project with which the test suite is associated. Set automatically when you add/edit/delete the name of a project. Node: Classes and Procedures
QC_SUMMARY	Summary of the test. Set automatically when you add/edit/delete a summary. Node: Classes and Procedures Available as environment variable while test runs

Creating Template Attributes

Use template attributes to organize tests by various kinds of data. Use an attribute on specific kinds of tests, and filter on that attribute to find all tests with that attribute.

A number of template attributes are provided with *QA*Director, but you can also create your own custom attributes.

To create a new template attribute:

1. Open a *QA*Director project by clicking **File>Open** and select a project from the project list.
2. From the **Tools** menu, choose **Manage Template Attributes**. The Template Attributes dialog box appears.
3. From the **Assign Template Attributes to** list select the project to which to assign attributes.
4. In the **Attributes Details** section, in the **Name** field, enter the new template attribute. Begin the name with **QC_** for consistency with other attribute names.
5. In the **Display Name** field enter the name which will appear on the Custom Attributes tab of the Test Class or Test Procedure properties dialog box.
6. In the **Type** field select the attribute type from the list.

7. In the **Value** field select the value from the list.
8. Select the **Value can be edited** check box to have the ability to change the value that appears in the field.
9. Select the **Display multi-line field text** check box for fields that require a lengthy value.
10. Click the **Add** button to add the new attribute to the **Custom Attributes** list.
11. If Global Attributes are selected in the **Assign Template Attributes to field**, click **Synchronize** to update all the suites in the project.
12. Click **Apply** to apply the changes.
13. Click **OK** to close the dialog box.

Deleting Template Attributes

It is possible to delete template attributes when they are no longer necessary.

To delete a template attribute:

1. Open a *QA*Director project by clicking **File>Open.** Select a project from the project list.
2. Click **Tools**>**Manage Template Attributes**. The Template Attributes dialog box appears.
3. In the **Custom Attributes** field, select the attribute to delete.
4. To delete the attribute click **Delete**.
5. A confirmation message appears. Click **Yes**.
6. Click **OK** to close the dialog box.

Updating Template Attributes

Using the Template Attribute dialog box, assign attributes to projects, list custom attributes, choose types and values, and more.

To update a template attribute:

1. Open a *QA*Director project by clicking **File>Open** and select a project from the project list.
2. Click **Tools>Manage Template Attributes**. The Template Attributes dialog box appears.
3. From the **Assign Template Attributes to** dropdown list select the project to which the attribute is assigned.
4. In the Attributes Details section, in the **Name** field, update the template attribute. Begin the name with **QC**_ for consistency with other attribute names.
5. In the **Display Name** field update the display name which will appear on the Custom Attributes tab of the Test Class or Test Procedure properties dialog box.
6. In the **Type** field select the attribute type from the dropdown list.
7. In the **Value** field select the value from the dropdown list.
8. Select the Value can be edited check box to have the ability to change the value that appears in the field.
9. Select the **Display multi-line field text** check box for fields that require a lengthy value.
10. Click the **Add** button to add the new attribute to the **Custom Attributes** list.
11. If Global Attributes are selected in the **Assign Template Attributes to field**, click **Synchronize** to update all the suites in the project.
12. Click **Apply** to apply the changes.
13. Click **OK** to close the dialog box.

Exporting attributes

It is possible to export test attributes to a text file.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To export attributes:

1. Click **Tools>Export Attributes to Text File**. The **Save As** dialog box appears.
2. Navigate to the directory in which to save the attributes.
3. Type a name in the **Name** field.
4. Click **Open**. The **Select Attributes to Export** dialog box appears.
5. Check the attributes to export.
6. Click **OK**.

Run-Time Attributes

*QA*Director automatically gathers information about test execution while tests execute. It stores the information in name-value pairs called run-time attributes.

Note: All run-time attributes are available as **environment variables** while a test is running because *QA*Director automatically sets the attributes to environment variables before execution.

Run-Time Attribute	Executing Test Node	Value
QC_QADIRECTOR_HOME	Job	Path to the Compuware installation running the job.
QC_ARCH_NAME	Job	Name of the architecture on which the job is running, such as: pa-hpux10.20.
QC_ARCH_OS	Job	Name of the Compuware directory containing the *QA*Director binary for an architecture.
QC_CYCLES	Test	Cycle information based on Planning. **Note:** This attribute may not be deleted or changed
QC_FAIL_DESC	Test procedure	Description of the failure of the procedure, if the test procedure failed.
QC_JOB_ID	Job	ID number of the job.
QC_JOB_MACHINES	Job	List of machines designated in the job description.
QC_JOB_NAME	Job	Name of the job.
QC_JOB_TIME	Job	Time the job started.
QC_LICENSE_HOST	Job	Name of the machine on which a *QA*Director license is running.

QC_LICENSE_USER	Job	Name of the user of the *QA*Director license.
QC_MACHINE	Test procedure	Machine on which the test is running.
QC_OS_NAME	Job	Operating system of the machine on which the job is running.
QC_RISK	Test	Risk information based on Planning. **Note:** This attribute may not be deleted or changed
QC_RESULT_TOP	Job	Absolute path to the result directory.
QC_START_TIME	Test	Date and time the running test started running.
QC_START_TIME_GMT	Test	Date and time the running test started in Greenwich mean time.
QC_RESULT_DIR	Test	Result directory of the running test.
QC_UNEXPECTED	Test procedure	Set if the outcome of the running procedure was unexpected.

Importing Test Plans

*QA*Director's convenient Import Wizard helps import Microsoft Excel test plans into *QA*Director. The end result is a new test suite based on the imported test plan.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Requirements

The document must be an Excel 97 or Excel 2000 file.

Excel 97 or 2000 must be installed on the computer performing the import.

The Excel spreadsheet cannot be a workbook with multiple sheets.

Overview

To create a test suite from an Excel document, the Import Wizard converts the document's columns to test classes, test procedures, and attributes. The conversion is based on the column headings. During the import process, choose which columns to map to test classes, test procedures, and attributes. For example, choose that all items in a "Class" column be imported as test classes.

Example

This is an example of an Excel test plan that contains columns for test classes, test procedures, a description attribute, and a test purpose attribute:

Class	Proc	Description	Test Purpose
PO Module		Purchase Orders	

	PO1	vendor name	accuracy of vendor name
	PO2	vendor address	accuracy of vendor address
	PO3	vendor phone	accuracy of vendor phone
	PO4	vendor fax	accuracy of vendor fax
	PO5	vendor email	accuracy of vendor email
	PO6	vendor category	accuracy of vendor category
AR Module		Accts Receivable	
	AR1	customer name	accuracy of customer name
	AR2	customer address	accuracy of customer address
	AR3	customer phone	accuracy of customer phone
	AR4	customer fax	accuracy of customer fax
	AR5	customer email	accuracy of customer email
	AR6	customer category	accuracy of customer category

Importing a Microsoft Excel document

To import a Microsoft Excel document:

1. Click **File**>**Import>MSWord/Excel Import Wizard for Suites...**. The Import Wizard introduction dialog box appears.
2. Click **Next**. Step 1 appears.
3. Click **Browse** and select the Excel file to be imported.
4. Click **Next**. Step 2 appears.
5. In the **Suite Name** field, type the name to use for the new test suite.
6. Click **Next**. Step 3 appears.
7. Select a column that contains the test class.
8. Click **Next**. Step 4 appears.
9. Select a column that contains the test procedure.
10. Click **Next**. Step 5 appears.
11. Select a row to begin the import. This option is helpful if the first row or two of the spreadsheet contains header or descriptive information that you don't want to import.
12. Click **Next**. Step 6 appears.
13. Assign any remaining style tags to attributes. Attributes are fields that are used to gather and store information about the test. For example, the QC_Summary attribute corresponds to a field labeled "Summary" on the Test Class and Test Procedure dialog boxes. If the test plan includes a column for a summary of each test, assign the column to the QC_Summary attribute so that the Summary field is automatically completed for each test.

If the document contains a column that should be ignored, select the column in the list and then select the **Ignore** option. Information in this column will not be imported.

To map a column, select it in the list and then select the **Attribute** option. Map the column to one of *QA*Director's default attributes (displayed in the drop-down list), or create a custom attribute by typing a name in the Attribute field.

Default attributes: Map the style tag to one of *QA*Director's default attributes:

QC_SUMMARY: Corresponds to the **Summary** field on the **General** tab of the Test Class and Test Procedure dialog boxes. It is used to enter a short description of the test.

QC_PURPOSE: Corresponds to the **Purpose** field on the **Custom Attributes** tab of the Test Class and Test Procedure dialog boxes. It is used to describe the test's goals or rationale.

QC_KEYWORDS: Corresponds to the **Keywords** field on the **Custom Attributes** tab of the Test Class and Test Procedure dialog boxes. It is used to enter terms by which to find the test in searches.

QC_STATE: Corresponds to the **State** field on the **Custom Attributes** tab of the Test Class and Test Procedure dialog boxes. It is used to describe the status of the test: Unwritten, In Progress, Completed, or Broken.

QC_CYCLE: Corresponds to the **Cycle** field on the **Custom Attributes** tab of the Test Class and Test Procedure dialog boxes. It is used to describe planned testing cycles.

QC_RISK: Corresponds to the Cycle field on the **Custom Attributes** tab of the Test Class and Test Procedure dialog boxes. It is used to describe low, medium, and high risks.

Custom attributes: Create a custom attribute by typing a name in the **Attribute** field. For example, assume there is a priority status for each test in the test plan. If the priority information is contained within a separate column, map that column to a custom attribute named "Priority." Simply select the column from the list and enter the word `Priority` in the **Attribute** field. After the import is complete, look for the **Priority** field on the **Description** tab of the Test Class or Test Procedure dialog box.

14. Click **Next**.
15. Click **Finish**.

After the import is complete, the new test suite appears in the Tree View of *QA*Director's main window.

Importing a Test Plan from Microsoft Word

*QA*Director's convenient Import Wizard helps import Microsoft Word test plans into *QA*Director. The end result is a new test suite based on the imported test plan.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Requirements

The document must be a Microsoft Word 97 or Word 2000 file.

Microsoft Word 97 or 2000 must be installed on the computer performing the import.

Overview

To create a test suite from a Word document, the Import Wizard converts the document's paragraphs to test classes, test procedures, manual tests, and attributes. The conversion is based on the Microsoft Word **style** applied to each paragraph. Paragraphs tagged with a certain style, such as Heading 1, will be converted to test classes while paragraphs tagged with other styles are converted to test procedures, manual tests, and attributes. During the import process, choose which styles to map to test classes, test procedures, manual tests, and attributes.

The Import Wizard creates the test suite **hierarchy** based on the order that the test class, test procedure, and manual test paragraphs are listed in the Word document. This means that all test classes are listed under the root class, test procedures are listed under the preceding test class, and manual tests are assigned to the preceding test procedure. If a manual test happens to follow a test class, rather than a test procedure, then the manual test is imported but not assigned to a specific suite. In other words, the manual test will be imported directly into the Manual Test Manager in *QA*Director but not displayed in the test suite. Add the manual test to one or more test suites as desired.

Applying styles in the document

A Microsoft Word paragraph style is a set of character and paragraph formats that are saved under a style name. After creating a style, select a paragraph and use the style to apply a whole group of formats in one step. As explained in the preceding Overview section, the Import Wizard creates the test classes, test procedures, manual tests, and attributes based on the styles applied to the document's paragraphs. For example, all paragraphs tagged with a certain style will be imported as test procedures. Therefore, it is important to be consistent when tagging paragraphs.

Define and use any styles, but simplify the import process by using the **default styles** listed in this section. During the import process, the wizard scans the document for these default styles and uses them if they exist. To use these styles, create them in the document (define specific formatting for each style) and apply the styles to the appropriate paragraphs.

Note: The style names are **case sensitive**. Use all upper case as shown.

Create this style:	Apply it to this type of paragraph:
Heading 1	first level heading
QAD_TESTCLASS	test classes
QAD_TESTPROCEDURE	test procedures
Description	test description
Purpose	test purpose
Normal	default style
QAD_MANUALTEST	manual test names
QAD_MANUALDESC	manual test description
QAD_MANUALINST	manual test instructions
QAD_MANUALQUES	manual test yes/no questions
QAD_MANUALMULT	manual test multiple choice questions
QAD_CORRECTANS	the correct answer for the multiple choice question
QAD_POSSIBLEANS	the possible answers for the multiple choice question

Guidelines for manual tests

If the test plan contains manual tests with **multiple choice** questions, please note that the paragraph containing the multiple choice question must be immediately followed by the two paragraphs containing the correct answer and the possible answers, in that order. The possible answers must be separated with a carat symbol (^). For example:

How many minutes did the transaction require? (the question)

Less than 1 minute (the correct answer)

Less than 1 minute^1-3 minutes^More than 3 minutes (all possible answers)

Importing the Microsoft Word document

When finished formatting the Microsoft Word document, use the Import Wizard to import the document into *QA*Director.

To import the document:

1. Click **File>Import>MSWord/Excel Import Wizard for Suites...**. The Import Wizard introduction screen appears.
2. Click **Next**. Step 1 appears.
3. Click **Browse** and select the Word file to be imported.
4. Click **Next**. Step 2 appears.
5. In the **Suite Name** field, enter the name to use for the new test suite.
6. Click **Next**. Step 3 appears.
7. Select a test class style tag.

9. Click **Next**. Step 4 appears.
10. Select a test procedure style tag.
11. Click **Next**. Step 5 appears.

This dialog box lists the various attributes of manual tests:

MT Name: The name of the manual test.

MT Description: A description of the manual test.

MT Instruction: A manual test step that is an instruction.

MT Question: A manual test step that is a question.

MT Multiple Choice: A manual test step that is a multiple choice question.

MT Correct Answer: The correct answer for a multiple choice question.

MT Possible Answer: A possible answer for a multiple choice question.

If the test plan contains manual tests, use this dialog box to map the appropriate style to each manual test attribute. For example, select the **MT Name** attribute in the list, select the **Style Tag** option, and select the style assigned to names of manual tests. Select the **Ignore** option for any manual test attributes that are not applicable.

Note: If the document was tagged using the default styles (listed above), the appropriate styles are automatically selected for each attribute.

12. Select a script style tag.
13. Click **Next**. Step 6 appears.

14. Assign any remaining style tags to attributes. Attributes are fields that are used to gather and store information about the test. For example, the QC_Summary attribute corresponds to a field labeled "Summary" on the Test Class and Test Procedure dialog boxes. If the test plan includes paragraph for a summary of each test, assign the paragraph to the QC_Summary attribute so that the **Summary** field is automatically completed for each test.

If the document contains a paragraph that should be ignored, select the paragraph in the list and then select the **Ignore** option. Information in this paragraph will not be imported.

To map a paragraph, select it in the list and then select the **Attribute** option. Map the paragraph to one of *QA*Director's default attributes (displayed in the drop-down list), or create a custom attribute by entering a name in the **Attribute** field.

Default attributes: Map the style tag to one of *QA*Director's default attributes:

- QC_SUMMARY: Corresponds to the **Summary** field on the **General** tab of the Test Class and Test Procedure dialog boxes. It is used to enter a short description of the test.
- QC_PURPOSE: Corresponds to the **Purpose** field on the **Description** tab of the Test Class and Test Procedure dialog boxes. It is used to describe the test's goals or rationale.

- QC_KEYWORDS: Corresponds to the **Keywords** field on the **Description** tab of the Test Class and Test Procedure dialog boxes. It is used to enter terms by which to find the test in searches.
- QC_STATE: Corresponds to the **State** field on the **Description** tab of the Test Class and Test Procedure dialog boxes. It is used to describe the status of the test: Unwritten, In Progress, Completed, or Broken.
- QC_CYCLE: Corresponds to the **Cycle** field on the **Description** tab of the Test Class and Test Procedure dialog boxes. It is used to describe planned testing cycles.
- QC_RISK: Corresponds to the Cycle field on the **Custom Attributes** tab of the Test Class and Test Procedure dialog boxes. It is used to describe low, medium, and high risks.

9. **Custom attributes:** Create a custom attribute by typing a name in the **Attribute** field. For example, assume there is a priority status for each test in the test plan. If the priority information is contained within a separate paragraph, map that paragraph to a custom attribute named "Priority." Simply select the paragraph from the list and type the word `Priority` in the **Attribute** field. After the import is complete, look for the **Priority** field on the **Description** tab of the Test Class or Test Procedure dialog box.

10. Click **Next**.

11. Click **Finish**.

After the import is complete, the new test suite appears in the Tree View of *QA*Director's main window.

Testing Tools

Use Compuware testing tools or third party testing tools with *QA*Director.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Compuware Testing Tools

If using a Compuware testing tool such as *QA*Run, *QA*Hiperstation, File-AID/CS, or *QA*Load, set up a tool domain for the testing tool. A tool domain is a set of information entered about the testing tool. This information is used to browse, select, create, and execute scripts from within *QA*Director.

Third-Party Testing Tools

If using a third-party testing tool with *QA*Director, create a new integration for the tool. With this integration, it is possible to:

create or select a script to include in a test procedure

run the tool with the selected script

analyze the result to determine if the script passed or failed

communicate the result back to *QA*Director.

Optionally, the integration can help with browsing results of failed tests and editing scripts that were previously inserted.

Add the testing tool to be used first by clicking **File>New>Testing Tool**. The Add Tool Dialog Box appears. Use this to enter information about the tool, such as executable files.

Adding a Testing Tool

If using a Compuware testing tool such as *QA*Run, *QA*Hiperstation, File-AID/CS, or *QA*Load, set up a tool domain for the testing tool. A tool domain is a set of information entered about the testing tool. This information is used to browse, select, create, and execute scripts from within *QA*Director. If using a third-party testing tool with *QA*Director, create a new integration for the tool.

To add a testing tool:

1. Open the **System Administration Center**.
2. From the Testing Tools panel **Action** menu, choose **New**. The Add Tool dialog box appears.
3. Type the name of the testing tool and enter the following executables, commands and statuses:

   Browse information

   Tool Start

   Tool Edit

   Tool Run

   Tool Option

   Tool Exit

   Tool Help

   Tool Start, Tool Edit and Tool Run are executables that require familiarity with the testing tool and QADirector. See Test Tool Executables for a complete discussion.

4. Click **Apply** to apply the changes or click **OK** to save the changes and close the dialog box.

Testing Tool Executables

Note: This feature is not available if using *QA*Director with CARS Workbench.

For the new integration, create or edit the executable files using the Add Testing Tool Dialog Box. Some files are required while others are optional. The following table lists the directory, name, and purpose of each file. **When creating the files, replace XXX with the name of the testing tool.**

Directory and Name	Purpose
*QA*Director\bin\QC**XXX**Run.exe	**Required:** The program used to run the script.
*QA*Director\bin\QC**XXX**Start.exe	**Optional:** The program that starts the tool in order to record a script or lets you choose an existing script to use.
*QA*Director\bin\QC**XXX**Edit.exe	**Optional:** The program used to edit an existing script.

Write these programs in any language. Before beginning, become familiar with the files and file structure of both the testing tool and *QA*Director. Remember that the programs created are called from the test procedure directory. All directory movement, file movement, or permission changing will occur in this directory. Use the `%QC_RESULT_DIR%` variable to move files to the corresponding test procedure result directory.

QCXXXRun.exe

This program parses the command line and finds the name of the script file. It then starts the testing tool with this configuration file and runs the tool in "batch" mode. The `QCXXXRun.exe` program should not exit until the script/tool is finished. Before it exits, the program must communicate to *QA*Director whether the test passed or failed and if it failed, why. The easiest way to indicate a pass is to return a 0 exit status. To cause a script to fail with a specific error message, the program must shell off the following command:

```
qc_exec fail "your failure message here"
```

Instead of using the VB shell command, use the runproc utility defined in this document, as the VB program must wait for the `qc_exec` program to finish before exiting.

The `QCXXXRun.exe` program must be able to examine a log file, or otherwise identify the criteria for a passed or failed test, in order to know whether to call `qc_exec` or not. Often, this is done by looking at a log file or examining the return code of a spawned application.

If the `QCXXXRun.exe` program needs to store important files generated by the test run, the files can be copied from their original location to the new, special result area. Because the location of this area changes each time the test is run, it is referenced using the environment variable `QC_RESULT_DIR`. There are also many other environment variables that are available to `QCXXXRun.exe`, such as the name of the test being run, the name of the script, the purpose of the test, any other attributes defined in the default_template_attrs file, and so on. All can be accessed through environment variables.

QCXXXStart.exe

This program parses the command line to find the name of the script to create and then starts the program needed to create the script.

QCXXXEdit.exe

If a tooledit line is included in the XXX.tool file, the `QCXXXEdit.exe` program will be called when the user clicks **Edit** on the Test Scripts Manager dialog box. The `QCXXXEdit.exe` program must open the tool used to configure the script.

Advanced feature

If you call `qc_exec fail -file filename "message"`, the button named **More Info...** on the result tab of a test procedure result will run the filename specified.

For example, if the call to qc_exec was as follows:

```
qc_exec fail -file log.txt "The test failed"
```

When the user clicks **More Info...** in the test procedure result pane, the file "log.txt" runs, and since .txt files are associated with QADirector's internal viewer, that file will be automatically opened in the viewer when double clicked).

Note: The log.txt file must be in the result directory as specified by the %QC_RESULT_DIR% environment variable when the test is run. This is the directory where the test results are stored. If the file specified was "log.doc", then the file would automatically open in Microsoft Word (the associated program in Windows).

Tool Domains

If using a Compuware automated testing tool such as *QA*Run, *QA*Hiperstation, *QA*Load, TestPartner, MVS Batch or File-AID/CS, set up at least one tool domain for each tool. A tool domain is a set of information that is entered about the testing tool that allows for browsing, selecting, and creating scripts from within *QA*Director. For example, a *QA*Run tool domain includes the name and location of the script database, the database type, whether or not the database is public (can be used by others), and so on.

Note: The *QA*Director administrator assigns permissions. Specific permission must be assigned to view, to add, to modify or to delete tool domains.

Copying Tool Domains

If working with a testing tool on separate projects, it may be necessary to create identical tool domains.

To copy a tool domain:

1. Open the **System Administration Center**.
2. Select a tool domain from the **Tool Domain** panel list.

3. Click **Action>Copy**, or right-click and click **Copy.** A copy of the tool domain appears in the list preceded by `Copy (2)`. If another copy is made, the tool domain appears in the list preceded by `Copy of Copy of.`

Creating Tool Domains

Create a tool domain to use with an automated testing tool.

To create a tool domain:

1. Open the **System Administration Center**.
2. Click **Action>New** from the **Tool Domains** panel. The Add Tool Domain dialog box appears.
3. In the **Type** list, select the necessary application. The appropriate tool domain dialog box appears.
4. Enter the required information.
5. Click **OK**.

Deleting Tool Domains

To organize testing tools, delete a tool domain when work with it has finished. If a tool domain is deleted, scripts will not run until the tool domain is recreated and points to the same database.

To delete a tool domain:

1. Open the **System Administration Center**.
2. Select a tool domain from the **Tool Domain** list.
3. Click **Action>Delete**, or right-click and click **Delete**. A message appears asking for a confirmation.
4. Click **Yes**.

Sharing Tool Domains

To share a tool domain with other team members, select the **Public** checkbox on the Add Tool Domain dialog box. If it is necessary to enter a path to a script database, use a mapped drive letter in the path. If the database is located on a local drive, first map the local drive to a shared drive letter and specify the shared drive letter when creating the tool domain. Other team members should map to the database location using this same drive letter specified in the tool domain.

Compuware Tool Domain Properties

The following describes the properties of the File-AID/CS Compare tool domain:

General Tab

Name: Name of the tool domain.

Owner: User name for the tool domain creator.

Description: Description of the tool domain.

Type list: Testing tool type.

Version:

Before 3.2: Enables the Access database type.

3.2 and above: Enables the File-AID repository.

Database Type: Access or File-AID repository.

Public: Displays whether or not this tool is shared with other users.

Testing Tool Database: Database name.

Browse: Click to select a database, repository or file.

User Name: Database user name.

Password: Database password name

Default Options Tab

Result Directory: Path of the directory to which the results are saved.

Status: Status depending on how *QA*Director will manage the pass/fail status of File-AID/CS Compare scripts.

Help: Accesses dialog level help.

OK: Saves the changes and returns to the center.

Cancel: Returns to the center without saving changes.

Apply: Saves the changes without closing the dialog box.

File-AID/CS Compare Tool Domain Properties

The following describes the properties of the File-AID/CS Compare tool domain:

General Tab

Name: Name of the tool domain.

Owner: User name for the tool domain creator.

Description: Description of the tool domain.

Type list: Testing tool type.

Version:

Before 3.2: Enables the Access database type.

3.2 and above: Enables the File-AID repository.

Database Type: Access or File-AID repository.

Public: Displays whether or not this tool is shared with other users.

Testing Tool Database: Database name.

Browse: Click to select a database, repository or file.

User Name: Database user name.

Password: Database password name

Default Options Tab

Result Directory: Path of the directory to which the results are saved.

Status: Status depending on how *QA*Director will manage the pass/fail status of File-AID/CS Compare scripts.

Help: Accesses dialog level help.

OK: Saves the changes and returns to the center.

Cancel: Returns to the center without saving changes.

Apply: Saves the changes without closing the dialog box.

File-AID/CS Convert Tool Domain Properties

The following describes the properties of the File-Aide /CS Convert Tool Domain:

General Tab

Name: Displays the name of the tool domain.

Owner: Displays the user name for the tool domain creator.

Description: Displays a description of the tool domain.

Type list: Displays the testing tool type.

Database Type: Displays the database type.

Public: Displays whether or not this tool is shared with other users.

Testing Tool Database: Displays location and name of the database.

Help button: Accesses dialog level help.

OK button: Saves the changes and returns to the center.

Cancel button: Returns to the center without saving changes.

Apply button: Saves the changes without closing the dialog box.

MVS Batch Tool Domain Properties

The following describes the properties of the MVS Batch tool domain:

General Tab

Name: Name of the tool domain.

Owner: User name for the tool domain creator.

Description: Description of the tool domain.

Type: Testing tool.

Database Type: Database from the list if the testing tool requires a database.

Public: Displays whether or not this tool is shared with other users.

Testing Tool Database: Location and name of the database.

User Name: User name used for logging onto the MVS Batch database.

Password: Password used for logging onto the MVS Batch database.

Connect Using Tab

Emulator Session check box: Default emulator session to use when running scripts associated with this tool domain. It is possible to change this default before running the scripts.

Logon Macro field: Macro for logging onto the emulator session.

Auto Logon check box: If selected, an automatic TSO logon is enabled for running unattended mainframe jobs. This check box is disabled if no macro is selected in the **Logon Macro** field.

Emulator Session Shortcut field: Information necessary to start the emulator session: The session path and name, in addition to the executable path and name.

User ID field: The &tsouserid variable.

Password field: The &tsopassword variable.

Logoff Macro field: The logoff macro to log off the emulator session automatically.

Job Control Tab

Job Statement field: Displays the user-specific MVS job control information from User Options:

&jobname: Obtains job name from **User Options**.

&accounting: Obtains accounting information from **User Option**s.

&name: Obtains user name or run name from **User Options**.

&class: Obtains job class from **User Options**.

&msgclass: Obtains job scheduler message output class from **User Options**.

Name of Job Control Dataset (Optional) field: Job Control Dataset

DSN field: RunLog dataset name

Unit: The unit. The default is SYSDA.

Volume: The volume.

Attach user's High Level Dataset Qualifier in front of the dataset name check box: If this is selected, the user's high-level dataset qualifier will be attached to datasets such as the RunLog and the compare log. The user's high-level dataset qualifier is defined in **User Option**s. If everyone is using the same high level dataset qualifier, type it on the **Global Job Information** tab in this dialog box.

Retention Period field: Displays the number of days the dataset should be retained. The data set cannot be deleted during the retention period. Enter any number between 0 and 9999. This field is optional.

Global Job Information Tab

Release list: *QA*Hiperstation release.

Command Line: The command line.

***QA*Hiperstation Load Library field:** The library. This information is common for everyone who uses this tool domain.

High Level Dataset Qualifier: The high-level dataset qualifier. This information is common for everyone who uses this tool domain.

Help: Accesses dialog level help.

OK: Saves the changes and returns to the center.

Cancel: Returns to the center without saving changes.

Apply: Saves the changes without closing the dialog box.

*QA*Hiperstation Tool Domain Properties

The following describes the properties of the *QA*Hiperstation tool domain:

General Tab

Name: Name of the tool domain.

Owner: User name for the tool domain creator.

Description: Description of the tool domain.

Type: Testing tool type.

Database Type: Database type.

Public: Displays whether or not this tool is shared with other users.

***QA*Load Server Name:** Name of the computer where *QA*Load is installed.

Connect Using Tab

Emulator Session check box: Displays the default emulator session to use when running scripts associated with this tool domain. It is possible to change this default before running the scripts.

Logon Macro field: Displays the macro for logging on to the emulator session.

Auto Logon check box: If selected, an automatic TSO logon is enabled for running unattended mainframe jobs. This check box is disabled if no macro is selected in the **Logon Macro** field.

Emulator Session Shortcut field: Displays the information necessary to start the emulator session: The session path and name, in addition to the executable path and name.

User ID field: Displays the &tsouserid variable.

Password field: Displays the &tsopassword variable.

Logoff Macro field: Type the logoff macro to log off the emulator session automatically.

Job Control Tab

Job Statement field: Displays the user-specific MVS job control information from User Options:

&jobname: Obtains job name from User Options.

&accounting: Obtains accounting information from User Options.

&name: Obtains user name or run name from User Options.

&class: Obtains job class from User Options.

&msgclass: Obtains job scheduler message output class from User Options.

Name of Job Control Dataset (Optional) field: Displays the Job Control Dataset.

DSN field: Displays the RunLog dataset name.

Unit: Displays the unit. The default is SYSDA.

Volume: Displays the volume.

Attach user's High Level Dataset Qualifier in front of the dataset name check box: If this is selected, the user's high-level dataset qualifier will be attached to datasets such as the RunLog and the compare log. The user's high-level dataset qualifier is defined in **User Options**. If everyone is using the same high level dataset qualifier, enter it on the **Global Job Information** tab on this dialog box.

Retention Period field: Displays the number of days the data set should be retained. The data set cannot be deleted during the retention period. Enter any number between 0 and 9999. This field is optional.

*QA*Hiperstation Tab

Wait Times: Displays the time interval that *QA*Hiperstation will wait for various responses or processes to complete.

Other Options: Displays the check boxes next to the desired options.

APPLID Node Data: Displays the prefix and suffix used for *QA*Hiperstation APPLIDS defined during the *QA*Hiperstation installation.

Dynamic Reports Datasets: Displays the unit and volume on which the Journal or Log datasets will be allocated dynamically.

Format of Date Fields: Displays the format to use for date fields.

Global Job Information Tab

Release list: Displays the *QA*Hiperstation release.

Command Line: Displays the command line.

***QA*Hiperstation Load Library field:** Displays the library. This information is common for everyone who uses this tool domain.

High Level Dataset Qualifier: Displays the high level dataset qualifier. This information is common for everyone who uses this tool domain.

Help: Accesses dialog level help.

OK: Saves the changes and returns to the center.

Cancel: Returns to the center without saving changes.

Apply: Saves the changes without closing the dialog box.

*QA*Load Tool Domain Properties

The following describes the properties of the *QA*Load tool domain:

General Tab

Name: Name of the tool domain.

Owner: User name for the tool domain creator.

Description: Description of the tool domain.

Type: Testing tool type.

Database Type: Database type.

Public: Displays whether or not this tool is shared with other users.

***QA*Load Server Name:** Name of the computer where *QA*Load is installed.

Default Options Tab

Timing Files Directory Path on the Server: Displays the directory where *QA*Load timing files are stored.

Help: Accesses dialog level help.

OK: Saves the changes and returns to the center.

Cancel: Returns to the center without saving changes.

Apply: Saves the changes without closing the dialog box.

*QA*Run Tool Domain Properties

The following describes the properties of the *QA*Run tool domain:

General Tab

Name: Name of the tool domain.

Owner: User name for the tool domain creator.

Description: Description of the tool domain.

Type list: Testing tool type.

Database Type: Database type.

Public: Displays whether or not this tool is shared with other users.

Testing Tool Database: Location and name of the database.

User Name: User name used for logging onto the *QA*Run database.

Password: Password used for logging onto the *QA*Run database.

Default Options Tab

Run Environment: Displays the run environment for scripts in the *QA*Run database. These environments are defined in *QA*Run.

Command Line: Displays the command line parameter required by the *QA*Run scripts.

Help: Accesses dialog level help.

OK: Saves the changes and returns to the center.

Cancel: Returns to the center without saving changes.

Apply: Saves the changes without closing the dialog box.

TestPartner Tool Domain Properties

The following describes the properties of the TestPartner tool domain:

General Tab

Name: Name of the tool domain.

Owner: User name for the tool domain creator.

Description: Description of the tool domain.

Type list: Testing tool type.

Database Type: Database type.

Public: Displays whether or not this tool is shared with other users.

Testing Tool Database: Location and name of the database.

User Name: User name used for logging onto the TestPartner database.

Password: Password used for logging onto the TestPartner database.

Default Options Tab

Playback Environment: Displays the default playback environment for the TestPartner scripts. These are defined in TestPartner.

Command Line: Displays the command line parameter required by the TestPartner scripts.

Help: Accesses dialog level help.

OK: Saves the changes and returns to the center.

Cancel: Returns to the center without saving changes.

Apply: Saves the changes without closing the dialog box.

Using the Test Library

When opening *QA*Director for the first time, the **Test Library** appears on the right side of the application. Click the **X** button to close it. To move the library, click and hold the **Test Library** title bar, and drag it to the preferred location.

To open the **Test Library**, click **View>Test Library**.

The **Test Library** is a repository for all available tests within a project. It contains test classes, class templates, test procedures, and test scripts. The **Test Library** makes it possible to use these tests in different suites within a project. As tests are used across suites, changes are saved to the **Test Library** and all suites.

To use tests across several projects, make the test global. Right click the test and choose global from the menu. Note that the children will be set to global as well.

The **Test Library** contains four folders: test classes, class templates, test procedures, and test scripts. The test classes, test procedures, and test scripts folders act as repositories. Class Templates retain a fixed class structure that remains consistent across suites.

When copying a test from the **Test Library** to a suite or if creating a test in a suite and attaching it to the **Test Library**, the suite does not actually contain the test. Instead, a marker resides in the suite and references the actual test in **Test Library.** The suite always references the **Test Library** unless a test within a suite has been detached from the **Test Library**.

Viewing Test Associations in the Test Library

To view associations between tests (classes, procedures, scripts, and suites), right-click a test in the **Test Library** and choose **View Test Association** from the menu. The View Test Association dialog box appears. This displays all the items one level above the item selected, that use the item.

For example, if you right-click procedure "Abc," and click **View Test Association** from the menu, the dialog box displays classes and suites associated with "Abc" since these items are one level or more above the procedure level in the test hierarchy. Any scripts associated with "Abc" will not display since scripts are one level below procedures.

Test Library Rules

Keep in mind the following rules when using the Test Library:

Test class and test procedure names must be unique within a project.

Test script names within the **Script Information Center** are unique within the Tool Domain.

Tests will contain the same name across suites if they are referenced from the **Test Library**.

Tests can be used interchangeably in the **Suite Information Center** and **Test Library** as long as they are attached to both.

Test names within the **Test Library** and tests that are detached from the **Test Library** cannot share a name. When detaching the reference from the **Test Library**, *QA*Director creates a new test that resides in the suite. Since all tests in a project must have unique names, the detached test must be renamed.

Expanding or Collapsing Tests

As tests are added, use the toolbar to expand or collapse an entire test suite or single tests.

To expand or collapse the entire Test Library tree:

1. Choose a suite.
2. Click **View>Expand All**.

To expand or collapse a single test:

1. Choose a specific test.
2. Click **View>Expand**.

Tips for Creating Tests to Add to the Library

A few things to remember when adding tests to the Test Library are:

Test class and test procedure names must be unique within a project.

Test script names within the Test Script Center are unique within the Tool Domain.

Tests will contain the same name among suites if they are referenced from the Test Library.

Tests can be used interchangeably in the Suite Information Center and Test Library as longs as they are attached to both.

Test names within the Test Library and tests that are detached from the Test Library cannot share a name. When detaching the reference from the Test Library, *QA*Director creates a new test that resides in the suite. Since all tests in a project must have unique names, the detached test must be renamed.

Adding a Test Script to the Test Library While in the Library

If working in the **Test Library**, it is possible to add a test script to the library from the **Script Information Center** and remain in the library.

To add a script to the test library:

1. Open the **Test Library**.
2. Right-click in the **Test Library**, and choose **Add Test Script** from the menu. The Add From Script Information Center Dialog Box appears.
3. Select a tool from the **Testing Tool** list.
4. Select a script from the **Script** list, and click **Add**.
5. Click **OK**.

Tip: To select more than one script in consecutive order, click on the first script, hold down **Shift**, then click the last script. To select several random scripts, click the first script, hold down **CTRL** , then select each individual script.

Deleting Tests, Procedures, and Scripts

When deleting a test script from the **Test Library**, the script is deleted from all of the procedures that it is used in, but the test script remains in the **Test Script Information** Center.

When deleting a Class Template from the **Test Library**, a message appears offering the option to delete all references or convert all references to a class.

To delete a test or script:

1. From the **Suite Information Center** or the **Test Library**, right-click a test class, test procedure, or test script.
2. Choose **Delete**.
3. Confirm the deletion.

Executing Tests in the Suite and Job Information Centers

After suites and tests are designed, organized, and initially run in the **Suite Information Center**, the *QA*Director **Job Information Center** lists all scheduled jobs (tests), jobs in progress, and results. From within this center, perform further test execution tasks. The Jobs panel is divided into two sections: result folders and the job related tabs. The job related tabs are Schedule, Execution, and Results. The Schedule tab displays jobs that are in queue to execute The Execution tab displays jobs that are currently running. The Results tab displays the results of jobs that have executed.

The first step in executing a test, or running a job, is to select the tests to run. Then, describe how and when the tests will run. This is called the job description. After submitting the job, it normally runs as soon as a Test Execution Server is available on a computer. However, it is possible to schedule the job to run at a specific time or specify that the job run on a specific computer.

If the testing tool is a Compuware product such as *QA*Run, *QA*Hiperstation, File-AID/CS, or *QA*Load set up a tool domain for the testing tool. A tool domain is a set of information entered about the testing tool. This information is used to browse, select, create, and execute scripts from within *QA*Director.

Create a tool domain by opening the **System Administration Center** and accessing the tool domain dialog box from the tool domains panel. Type a description of the tool domain as well as the database name and other pertinent information.

A Test Execution Server (TES) must be running on each machine where the tests run. In addition, a user must be logged onto the workstation in order for the TES to be active. In most cases, the TES should already be running. If not it is necessary to start the TES.
Additionally, the Test Management Server (TM) reads job submissions from the database, manages jobs, tracks the machines available for test execution, and manages the licenses used for manual test Web execution. It also reports on the status of jobs in progress.

About Executing Tests (Running Jobs)

After suites and tests are designed, organized, and initially run in the **Suite Information Center**, the *QA*Director **Job Information Center** lists all scheduled jobs (tests), jobs in progress, and results. From within this center, perform further test execution tasks. The Jobs panel is divided into two sections: result folders and the job related tabs. The job related tabs are Schedule, Execution, and Results. The Schedule tab displays jobs that are in queue to execute The Execution tab displays jobs that are currently running. The Results tab displays the results of jobs that have executed.

The first step in executing a test, or running a job, is to select the tests to run. Then, describe how and when the tests will run. This is called the job description. After submitting the job, it normally runs as soon as a Test Execution Server is available on a computer. However, it is possible to schedule the job to run at a specific time or specify that the job run on a specific computer.

If the testing tool is a Compuware product such as *QA*Run, *QA*Hiperstation, File-AID/CS, or *QA*Load set up a tool domain for the testing tool. A tool domain is a set of information entered about the testing tool. This information is used to browse, select, create, and execute scripts from within *QA*Director.

Create a tool domain by opening the **System Administration Center** and accessing the tool domain dialog box from the tool domains panel. Type a description of the tool domain as well as the database name and other pertinent information.

A Test Execution Server (TES) must be running on each machine where the tests run. In addition, a user must be logged onto the workstation in order for the TES to be active. In most cases, the TES should already be running. If not it is necessary to start the TES.
Additionally, the Test Management Server (TM) reads job submissions from the database, manages jobs, tracks the machines available for test execution, and manages the licenses used for manual test Web execution. It also reports on the status of jobs in progress.

Basic Execution

Standard Windows execution requires that *QA*Director be installed on the machine where the job will run. In addition, someone must monitor the machine to complete the manual tests when they appear in the manual test viewer. The advantage of this method is that the job can include both automated and manual tests.

Submitting a job using Standard Windows execution:

1. Select the test suite that contains the manual tests.
2. From the toolbar, choose **Run <name of suite, class, or procedure>**. The Job Description dialog box appears.
3. Set up the job description as for any other job by selecting the Windows platform on which to run the job and, if appropriate, the specific computer. See Running a Job for information about setting up the job description.
4. Click **Submit**. When a manual test is encountered in the job, it is displayed in the manual test viewer.
5. In the manual test viewer, read and respond to each step:

   Question steps: Click the **Answer** box and type your response. Then, select whether the step passed or failed.

   Multiple Choice steps: Select an answer from the **Answer** list.

   Instruction steps: Simply follow the instructions as stated.

After reading and responding to all of the steps in a manual test, click **Submit** at the bottom of the screen.

Caution: You cannot change a manual test after it is submitted, so do not submit the test until you have completed all the steps in the test.

The following are some tips for Standard Windows Execution:

Recording comments: Use the comment box next to each step to record information about that step. You can also use the comment box at the end of the test to record general notes about the test. This is not mandatory.

Resetting the manual test: Click the **Reset** button to discard your changes and return to the default answers for each step.

Printing the manual test: To print a copy of the manual test for your records, click the **Print** button.

Attaching files: Use the **Attached Files** field at the end of each manual test to attach any supporting files. For example, you may want to attach a output log file or a screen capture of an unexpected error message.

Running a Job

The first step in running a job is to select the tests to run. Then, describe how and when the tests will run. This is called the job description. After submitting the job, it normally runs as soon as a Test Execution Server is available on a computer. However, it is possible to optionally schedule the job to be run at a specific time or specify that the job run on a specific computer.

A **Test Execution Server** (TES) must be running on each machine where the tests run. In addition, a user must be logged in to the workstation in order for the TES to be active. In most cases, the TES should already be running. If it is not, click `Start>Programs>Compuware>`*`QA`*`Director>Test Execution Server`. In Windows XP, click `Start>All Programs>Compuware>`*`QA`*`Director>Test Execution Server`.

To set up the job description and run a job:

1. From the **Suite Information Center,** open the test suite that contains the tests to run.

2. In the Tree view, select the item to run, either the root class (the entire test suite will run), a test class (all descendant classes and procedures will run), a single test procedure, or a single script. If a job has already been created for the tests to run, select a job from the **Job Information Center** . Then perform the remaining steps.
3. Click **Actions>Run.** The Job Description dialog box appears.

   The selected tests to run are displayed on the **Tests to run** list on the **Tests** tab. For example, if choosing to run the entire test suite, the name of the test suite appears in the list.

   On the **General** tab, select options that determine how and when to run the job.

5. Set the advanced testing options on the **Advanced** tab.

6. Click **Submit**. A confirmation dialog appears. Click **OK**. The dialog box will disappear within five seconds if **OK** is not clicked. The *QA*Director adds the job to the list of scheduled jobs. To view scheduled jobs, click the **Scheduled** tab on the **Job Information Center Jobs** panel.

7. When the job stops running, the results are available from the **Results** tab on the **Job Information Center Jobs** panel.

8. For execution details and test results, analyze job results.

Scheduling a Job

By default, a submitted job runs as soon as a Test Execution Server becomes available. When jobs are scheduled as recurring, *QA*Director schedules the next occurrence as the current job executes. It does not schedule all the occurrences at once.

To schedule a job to run at a specific time or unattended:

1. Select a job from the Schedule tab on the Jobs panel.
2. Click **Actions>Edit Schedule**. (It is also possible to access the Scheduler dialog box from the Job Description dialog box.) The Scheduler dialog box appears.
3. Select either the **Start Now** option to start the job immediately or select the **Start Date Time** option and select an execution date and time from the **Start Date Time** list to schedule the job for future execution.
4. In the **Recurrence Pattern** group box, select a pattern. Depending upon the pattern, the fields in the adjacent group box will change. Modify the fields accordingly.

Tip: To run a job every day at the same time, select **Daily** in the **Recurrence Pattern** area. To stop the daily executions on a specific date, select **Ends By** in the **Range of Recurrence** area and choose a date from the **Ends By Date** list.

To view the progress of a job, find the running job in the **Execution** tab of the **Job Information Center Jobs** panel. Then, right-click on the job and select **View Results**. The Results dialog box contains the number of tests in the job that finished running and the number of those that *Passed*, *Failed*, or were *Not Executed*. When the job starts running, *QA*Director automatically updates the Results as tests execute with the outcomes of finished tests.

Re-Running a Job

After running a job and viewing the results, elect to rerun the entire job or only certain tests in the job. Running a test again lets you verify that the problem originated with the set-up of the test, or the application being tested, etc. If you have determined what the problems were, and where they occurred, and have remedied them, running the test again lets you verify they are corrected.

To re-run only the tests that failed from within the Results screen:

1. Select a result.
2. Click **Actions > Rerun Failed**.
3. Choose **All** to re-run all the tests, or **Failed** to re-run only the failed tests. The Job Description Dialog Box appears.
4. The **Tests to Run** list will display either all the tests or the failed tests, depending on what was selected.

    3. Click **Submit**.

Advanced Execution

Remote execution refers to submitting a job from one computer and running the job on another computer. Submit a job right from a workstation, run the job on a remote computer such as a dedicated testing computer, and return later to check the progress of the job. When performing remote testing, note these guidelines:

The remote computer must be running and logged onto the network. A Test Execution Server must be running on the remote computer. *QA*Director itself does not need to be running on the remote computer.

The remote computer must be connected to the same *QA*Director database that the workstation computer is connected to.

On the workstation computer, run the job, but be sure to select the remote computer in the **Machines** field on the Job Description dialog box.

After submitting the job, close *QA*Director on the workstation computer or shut down the workstation computer if the *QA*Director database and the Test Management Server are not hosted on it.

At any time, start *QA*Director on the workstation computer to view the progress or results of the job.

Parallel Execution

With parallel execution, the children of a test class run at the same time on available machines in the network. This can only occur if more than one machine is available. Parallel execution minimizes running time and maximizes the use of network resources. Parallel execution is the opposite of serial execution, where sibling tests run one at a time in the order they appear in the test suite.

For each test class in the suite, it is possible to choose which child tests to run in parallel:

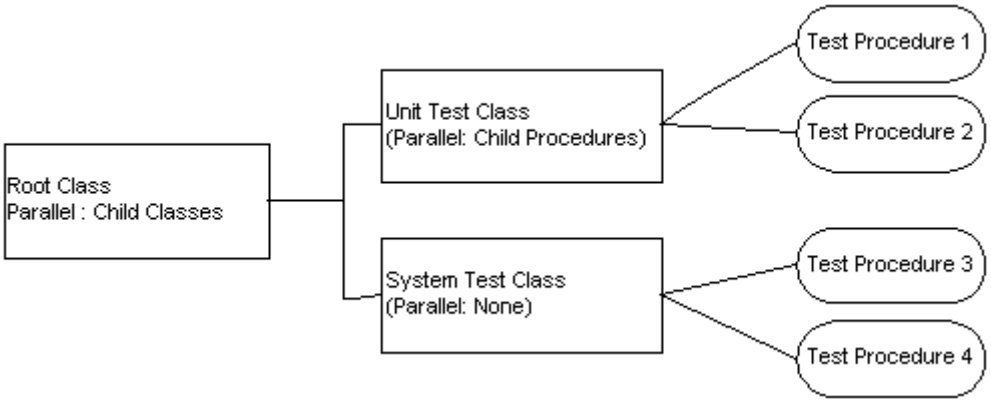
All child classes and procedures

Child classes only

Child procedures only

None (run all child tests consecutively)

The default option is classes only. This means that when running the test class, any child classes will run in parallel but child procedures will run serially. The option selected affects only the children of that class. It does not affect the children of the children or any other descendants. Thus, it is possible to set up the following configuration:



In this case, the root class is set to run child classes in parallel. Therefore, Unit Test Class and System Test Class will run at the same time on different machines (if two machines are available). However, Unit Test Class will run its procedures in parallel while System Test Class will run its procedures serially. This means that Test Procedure 1 and Test Procedure 2 can run at the same time (if machines are available) while Test Procedure 3 and Test Procedure 4 must run one at a time.

When tests are ready to run in parallel execution, they will be assigned to a machine as one becomes available. Note the following restrictions:

A test procedure must execute all its rules and scripts on one machine.

A test class must execute its pre-test rules and cleanup rules on one machine.

In the previous scenario, the job would run as follows, assuming there are two available machines running Test Execution Servers:

1. When the Root Class finishes its pre-test rules, Unit Test and System Test become ready to run.
2. Because the Unit Test and System Test classes are running in parallel, Test Procedure 1 runs on machine A while Test Procedure 3 runs on machine B. At this time, Test Procedure 2, which is allowed to run in parallel with Test Procedure 1, has no available machine.
3. After Test Procedure 1 or Test Procedure 3 finishes, Test Procedure 2 begins running on the available machine. Test Procedure 2 is run before Test Procedure 4 because it appears before Test Procedure 4 in the Suite Organization tree hierarchy.
4. Test Procedure 4 runs on the next available machine, provided that Test Procedure 3 has finished. Because these two tests are set to run serially, Test Procedure 4 cannot begin running until Test Procedure 3 is complete.

Note: When binding scripts, if the scripts of a test procedure are bound to a specific machine, the test procedure must run on that machine, even if another machine is available. For example, if Test Procedure 2 in the previous scenario was bound to machine A, it could not execute on machine B even if it became available before machine A.

Prerequisite

It is possible to execute tests in parallel if working in an environment where the *QA*Director Test Management Server resides on a server machine and manages execution on multiple client machines. A Test Execution Server must be running each client machine where the tests run. It is not possible to run scripts in parallel if running stand-alone installations of *QA*Director on various computers.

Specify Run in Parallel Option

Before running the job, make sure the test classes have the desired **Run in Parallel** option selected.

1. Open the test suite containing the test(s) to run.
2. In the **Suite Organization** tree, double-click the test class to run. The Test Class dialog box appears.
3. In the **Run in Parallel** list, select the desired option. Remember, the option selected affects only the children of the selected class. It does not affect the children of the children or any other descendants.
4. Repeat these steps as necessary for any other test classes.

Specify Available Machines

To set up and run the job, follow the steps outlined in Running a Job. However, on the **General** tab of the Job Description dialog box, be sure to select which machines are available for execution. To do so, click **Browse** next to the **Machines** field. In a team environment, the names of all client machines that are logged onto the network appear, connected to the team database, and match the platform specified in the **Platform** field. Select the computers to use for the parallel execution. Remember, if running a stand-alone installation of *QA*Director on one computer, only that single computer will appear.

Executing an Unattended Job

To run a job at night, or otherwise unattended, follow the procedure outlined in Running a job but note these additional guidelines:

Select only automated tests: On the **General** tab of the Job Description dialog box, select **Automated Tests Only** from the **Run Options** section. Automated tests are tests that do not require user interaction.

Enter the time to run the test: Schedule a time for the job to run. On the **Schedule** tab of the **Jobs** panel, select a job. Click **Actions>Edit Schedule**. Select the time from the **Start Time** list. For example, select **02:00am** to run the tests at 2:00 am tomorrow morning, or select **12:00am** from the **Start Time** list and select a specific date from the **Start Date** list to run the test at midnight on the specified date.

*QA*Director can execute unattended jobs on the mainframe. Schedule *QA*Hiperstation or MVS batch jobs to run at any time, even when no one is present to log onto TSO.

Executing an Unattended Mainframe Job

It is possible to run an unattended mainframe job by entering certain setup information in **User Preferences** and in the tool domains for *QA*Hiperstation and MVS batch jobs.

To run an unattended mainframe job:

1. Start *QA*Director.
2. Click **Tools>User Options**.
3. Click the ***QA*Hiperstation** tab.
4. Under **TSO Logon Information**, enter the user ID and password for logging onto TSO.
5. Click **OK**.
6. Click **System Administration Center** from the **Centers** toolbar.
7. From the Tool Domains panel **Action** menu, choose **Add**.
8. From the **Testing Tool** list, choose either ***QA*Hiperstation** or **MVS Batch**, depending on which type of tool domain to modify.
9. Type the tool domain name in the **Name** field, and click **Apply**. Or, if it is necessary to create a new tool domain, click **New**.
10. On the Tool Domain Detail dialog box, click the **Connect Using** tab.
11. In the **Logon Macro** field, click **Browse** and select your macro for logging onto the emulator session.

    Note: The macro must include the keyword **[getuipw]** (get user ID and password) in order to properly obtain the user's ID and password from User Preferences. See Logon and logoff macros for details. For an example of the required syntax, see the sample logon macro provided with *QA*Director in `\Program Files\Compuware\QADirector\Custom\Win32\hson.mac`.
12. Select the **Auto Logon** check box.

13. In the **Emulator Session Shortcut** field, enter the information needed to start the emulator session. If you are using the TN3270 emulator provided with *QA*Director, type TN3270 in this field. You can leave this field blank if your emulator session is normally running.

    You must enter the session path and name as well as the executable path and name. For example, for an EXTRA! Personal Client session, you must enter the EXTRA! executable and the path and name of the .EDP file. Separate the two paths with a space, as shown in the following example:

    ```
    c:\program files\E!PC\extra.exe c:\program files\E!PC\sessions\session1.EDP
    ```

14. In the **TSO Logon Information** fields, enter the **&tsouserid** and **&tsopassword** variables. At execution time, *QA*Director will use the ID and password entered in the tester's User Preferences.

    Note: You can enter a specific user ID and password here if you want all jobs to be executed under the same ID and password.

15. In the **Logoff Macro** field, click **Browse** and select your macro for automatically logging off the emulator session.
16. Repeat these steps for other *QA*Hiperstation or MVS batch job tool domains if applicable.

Executing a Job Immediately

If is not possible to wait for a job that is scheduled, run the job immediately on demand.

To bypass the job schedule and run a job immediately:

1. Click the **Job Information** icon from the **Centers** toolbar.
2. On the **Schedule** tab, select the job.
3. Click **Actions>Run Now** or right click and choose **Run Now**. The job schedule will change and the job will run immediately.

Executing Jobs Across a WAN

Executing Jobs Across a WAN

To execute a remote job across a WAN, testing tools must be installed on the local Test Execution Server as well as the remote Test Execution Server.

To execute a job across a WAN:

1. From the **TM Servers** panel in the **System Administration Center,** click **Actions>Manage Default Ports**. The Test Execution Administration dialog box appears.
2. Type the port numbers for each of the Test Execution Server ports.
3. Verify that the **Allocate Dynamically** check box is not checked.
4. Click **Apply** and **Ok**.
5. Restart the Test Management Server.
6. Ensure that the Test Execution Server is running by clicking **Start >Programs>Compuware>*QA*Director>Test Execution Server**. If using Microsoft Windows XP, click **All Programs>Compuware>*QA*Director>Test Execution Server**. The Test Execution Server appears as a hard drive icon in the system tray.
7. Choose **Change TE Type** from the menu. The Test Execution Server Configuration dialog box appears.
8. Configure the Test Execution Server.
9. Type the Test Execution Server port number in the **Test Execution Port** field or choose a port number by clicking the up and down arrows, and click **OK**.
10. Right-click the **Test Execution** icon again and choose **TM List** from the menu. The Test Management Server Configuration dialog box appears.
11. Configure the Test Management Server.
12. Right-click the **Test Execution** icon again and click **Exit**.

13. Restart the Test Execution Server by clicking **Start>Programs>Compuware>QADirector>Test Execution Server.**
14. Execute the test.

Note: The TM machine, local TE machines, and remote TE machines must communicate among each other via DNS servers or IP addresses. If a test does not execute, verify that the machines can indeed communicate.

About Remote Job Execution Across a WAN

Use *QA*Director to run a job remotely across a WAN. If a job is scheduled on a remote machine, *QA*Director retrieves the data for the job from the local machine and transmits the data to the remote machine. *QA*Director performs a tool domain look-up on the local machine. Therefore, the local and remote machines must be set up the same. The same database must reside on both remote and local machines. The environments, which include items like ODBC entries, paths, scripts, and database types, must be identical for both local and remote machines.

Before executing a job across a WAN, ensure that the following are available:

Test Management Server (TMS)

Test Driver

Test Execution Server (TES)

Test Engine

Note the port numbers available for test execution on the Test Management and Test Execution servers. These numbers are necessary for configuring *QA*Director on the Test Management Server Configuration dialog box to execute a remote job.

Executing Jobs Across a WAN

To execute a remote job across a WAN, testing tools must be installed on the local Test Execution Server as well as the remote Test Execution Server.

To execute a job across a WAN:

1. From the **TM Servers** panel in the **System Administration Center,** click **Actions>Manage Default Ports**. The Test Execution Administration dialog box appears.
2. Type the port numbers for each of the Test Execution Server ports.
3. Verify that the **Allocate Dynamically** check box is not checked.
4. Click **Apply** and **Ok**.
5. Restart the Test Management Server.
6. Ensure that the Test Execution Server is running by clicking **Start >Programs>Compuware>*QA*Director>Test Execution Server**. If using Microsoft Windows XP, click **All Programs>Compuware>*QA*Director>Test Execution Server**. The Test Execution Server appears as a hard drive icon in the system tray.
7. Choose **Change TE Type** from the menu. The Test Execution Server Configuration dialog box appears.
8. Configure the Test Execution Server.
9. Type the Test Execution Server port number in the **Test Execution Port** field or choose a port number by clicking the up and down arrows, and click **OK**.
10. Right-click the **Test Execution** icon again and choose **TM List** from the menu. The Test Management Server Configuration dialog box appears.
11. Configure the Test Management Server.

12. Right-click the **Test Execution** icon again and click **Exit**.
13. Restart the Test Execution Server by clicking **Start>Programs>Compuware>QADirector>Test Execution Server.**
14. Execute the test.

Note: The TM machine, local TE machines, and remote TE machines must communicate among each other via DNS servers or IP addresses. If a test does not execute, verify that the machines can indeed communicate.

Configuring the Test Execution Server

Configure the Test Execution Server (TES) to test locally or remotely. Via local testing, *QA*Director runs a job locally without binding scripts. Via remote testing, *QA*Director runs a job remotely using a WAN. If a job is scheduled on a remote machine, *QA*Director gets data from the local TES to run remote jobs, and binds each script to the selected machine.

To configure the TES:

1. Ensure that the **TES** is running by clicking **Start >Programs>Compuware>*QA*Director>Test Execution Server**. If using Microsoft Windows XP, click **All Programs>Compuware>*QA*Director>Test Execution Server**. The **TES** appears as a hard drive icon in the system tray.
2. Right-click the icon that appears and choose **Change TE Type** from the menu. The Test Execution Server Configuration dialog box appears.
3. Choose the **Local TE** option to run jobs locally or the **Remote TE** option to run remote jobs across a WAN.
4. If testing remotely, click the up or down arrow on the **Test Execution Port** box to choose a Test Execution port. Clear the **Dynamic** check box if specifying a port in the **Test Execution Port** box.

   Select the **Dynamic** check box to have Windows assign an available Test Execution port.
5. Click **OK** to configure the TES. Any changes will take effect after the TES is restarted.

Configuring the Test Management Server

When testing remotely, configure the Test Management Server (TMS) with the appropriate machine names and port numbers. Add, modify, or delete machines as necessary.

To configure the TMS:

1. Ensure that the **TES** is running by clicking **Start>Programs>Compuware>QADirector>Test Execution Server**. If using Microsoft Windows XP, choose **All Programs>Compuware>QADirector>Test Execution Server**. The **Test Execution Server** appears as a hard drive icon in the system tray.
2. Right-click the icon and choose **TM List** from the menu. The Test Management Configuration dialog box appears.
3. Add, update or delete a machine:

   To **add** a machine, type a machine name in the **Machine Name** field, and type a port number in the **Port field**, or select the port number using the arrows, and click **Add**. Make sure that the machine entered is accessible.

   To **update** an existing machine, select it from the **TM List**, modify it as necessary, and click **Update**.

   To **delete** an existing machine, select it from the **TM List**, and click **Delete**.
4. Click **OK**.

Customizing Jobs

Use the various settings, attributes, etc. to customize a job and run it in a particular manner. Define a rule for a job description to apply only to some jobs or temporarily override rules defined in the tests being run. Group jobs to see related items together, similar to an outline. Sort jobs by job name, then by owner to view all jobs related to specific owners. These are just a few of the ways jobs can be customized.

Customizing the View in the Job Information Center

Customize the way information is presented in the Job Information Center, by adding and removing columns.

To change the columns in the Job Information Center panel:

1. From the **Centers** toolbar, click the **Job Information Center** icon.
2. Click **Action>Customize Views**. The Customize View dialog box appears.
3. Click **Columns**. The Show Columns dialog box appears.
4. To add a column, select a column from the **Available Columns** list, and click **Add**.
5. To remove a column, select a column from the **Show these columns in this order** list, and click **Remove**.
6. To change the column order, select a column from the **Show these columns in this order** list, and click **Move Up/ Move Down** until the column is in the desired position.
7. Click **OK** to save the changes and return to the Customize View dialog box.
8. Click **OK** to close the Customize View dialog box.

Sorting the Jobs Panel View

Sorting is a way of arranging jobs in ascending or descending order. For example, sort jobs by job name, then by owner to view all jobs related to specific owners.

To sort the Jobs panel view:

1. From the **Centers** toolbar, click the **Job Information** icon.
2. Click **Action>Customize View**. The Customize View dialog box appears.
3. Click **Sort**. The Sort dialog box appears.
4. In the **Sort items by** list, select a field to sort by.
5. Select **Ascending** or **Descending** for the sort order.
6. To sort by an additional field, select a field from the **Then by** list.
7. Click **OK** to save the changes and return to the Customize View dialog box.
8. Click **OK** to close the Customize View dialog box.

Note: Sorting by more than one field sorts all items by the first field and then, within that sort, sorts again by the second field. For example, if you choose to sort by job name and then by defect number, the sort structure is "Job A 1 2 3, Job B 1 2 3."

Defining Rules for Job Descriptions

The first step in running a job is to select the tests to run. Then, describe how and when the tests will run. This is called the job description. Define a rule for a job description to apply only to some jobs, or temporarily override rules defined in the tests being run. After selecting a test in the Job Information Center and clicking **Actions>Run**, the Job Description dialog box appears.

To define a rule:

1. Select **Job Center** from the **Centers** toolbar.
2. Right-click a job from the job list and select **Edit Job**. The Job Description dialog box appears.
3. Click the **Advanced** tab.
4. Click **Set Rules**. The Job Description: Set Rules dialog box appears.
5. To update an existing rule, select the rule from the **Rules** list and click **Edit**. To create a new rule, click **Add**. The Add/Edit Rule dialog box appears.
6. Select the type of rule: environment variable, setup command, pass/fail command, or cleanup command.
7. Depending on the type of rule, complete the appropriate fields:

Environment Variable Fields:

Name: Enter the name of the environment variable.

Value: Enter the value of the environmental variable.

Setup, Pass/Fail, and Cleanup Fields:

Command: Enter the path and name of the command to execute, along with any parameters such as path and name of a file to create, open or copy. Remember to type **Start** before an executable path and command.

Bind to machine: Enter the machine name to execute the command on a specific machine.

Apply to: For test classes, choose only to define how to apply the rule to descendants. This option is disabled for test procedures.

Timeout: Enter the number of minutes for *QA*Director to wait after the command starts before ending the process.

Exit status: To require that the command have a specific exit status to be successful, enter the exit status that the command should have. Also select the **Require Exit Status** option.

Require exit status: Select this check box to require *QA*Director to mark that the command failed unless it produced a specific exit status.

8. Click **OK** to save the rule and return to the Job Description: Rules dialog box.
9. Close the dialog box.
10. Click **Submit** to run the job.

Deleting a Job

It is possible to delete a job from *QA*Director.

To delete a job:

1. From the **Centers** toolbar, click the **Job Information** icon.
2. On the **Schedule** tab, select the job.

3. Click **Actions>Delete Job**. A verification message appears.
4. Click **OK**.
5. If the job has a recurring schedule, the Job Delete dialog will appear. Select to delete the single occurrence or delete all occurrences, and click **OK**.

Editing a Job

To update a job with new information:

1. From the **Centers** toolbar, click the **Job Information Center** icon.
2. Click the **Schedule** tab, and select the job to edit.
3. Click **Actions>Edit Job**. The Job Description dialog box appears.
4. Make the necessary changes, and click **Save**.

Stopping a Job

It may be necessary to stop a job after it has begun running.

To stop a job that is currently running:

1. From the **Centers** toolbar, click the **Job Information** icon.
2. Click the **Execution** tab.
3. Select the running job.
4. Click **Actions>Abort Job**. The Abort Job dialog box appears.
5. Click **OK**.

Grouping the Jobs Panel View

Group jobs to see related items together, similar to an outline. For example, group jobs by job name to separate items according to their associated job. Expand or collapse the group headings to display or hide the items they contain.

To group the Jobs panel view:

1. From the **Centers** toolbar, click the **Job Information** icon.
2. Click **Action>Customize View**. The Customize View dialog box appears.
3. Click **Group By**. The Group By dialog box appears.
4. In the **Group items by** list, select a field to group by.
5. Select **Ascending** or **Descending** for the sort order of the group headings.
6. To group by subgroups, click a field in the **Then by** list.
7. Click **OK** to save the changes and return to the Customize View dialog box.
8. Click **OK** to close the Customize View dialog box.

Opening Saved Job Descriptions

Open previously saved descriptions to review information or use the information for another job. Job Descriptions describe how and when the tests will run. After selecting a test in the Job Information Center and clicking **Actions>Run**, the Job Description dialog box appears.

To open a saved job description if upgraded from QADirector 4.5.1:

1. Choose **Open Saved Job Description** from the **Job Information Center Action** menu. The Open Saved Job Description dialog box appears:
   - Select **From Saved Job Descriptions** option for job descriptions saved in *QA*Director 05.00 and later. After clicking **OK**, the Open Job Description dialog box appears.
   - Select **From Job Description files (prior to 05.00)** option for job descriptions saved in releases prior to *QA*Director 05.00. After clicking **OK**, the file directory opens. Browse to the necessary file.

Saving Job Descriptions

The first step in running a job is to select the tests to run. Then, describe how and when the tests will run. This is called the job description. After selecting a test in the Job Information Center and clicking **Actions>Run**, the Job Description dialog box appears. After setting up the job description, save it for use with future jobs. This can save time if normally using many of the same settings for several jobs.

To save a job description:

1. Click **Save** on the Job Description dialog box. The Job Description Properties dialog box appears.
2. Enter a file name and description.
3. Click **OK**.

Deleting Saved Job Descriptions

Job Descriptions describe how and when the tests will run. After selecting a test in the **Job Information Center** and clicking **Actions>Run**, the Job Description dialog box appears. When a Job Description becomes obsolete, delete it using the following procedure.

To delete a saved job:

1. Choose **Open Job Description** from the **Job Information Center Action** menu. The Open Saved Job Description dialog box appears.
2. Select the job description file or multiple job description files and click **Delete**.

Run-Time Attributes

*QA*Director automatically gathers information about test execution while tests execute. It stores the information in name-value pairs called run-time attributes.

Note: All run-time attributes are available as **environment variables** while a test is running because *QA*Director automatically sets the attributes to environment variables before execution.

Run-Time Attribute	Executing Test Node	Value

QC_QADIRECTOR_HOME	Job	Path to the Compuware installation running the job.
QC_ARCH_NAME	Job	Name of the architecture on which the job is running, such as: pa-hpux10.20.
QC_ARCH_OS	Job	Name of the Compuware directory containing the *QA*Director binary for an architecture.
QC_CYCLES	Test	Cycle information based on Planning. **Note:** This attribute may not be deleted or changed
QC_FAIL_DESC	Test procedure	Description of the failure of the procedure, if the test procedure failed.
QC_JOB_ID	Job	ID number of the job.
QC_JOB_MACHINES	Job	List of machines designated in the job description.
QC_JOB_NAME	Job	Name of the job.
QC_JOB_TIME	Job	Time the job started.
QC_LICENSE_HOST	Job	Name of the machine on which a *QA*Director license is running.
QC_LICENSE_USER	Job	Name of the user of the *QA*Director license.
QC_MACHINE	Test procedure	Machine on which the test is running.
QC_OS_NAME	Job	Operating system of the machine on which the job is running.
QC_RISK	Test	Risk information based on Planning. **Note:** This attribute may not be deleted or changed
QC_RESULT_TOP	Job	Absolute path to the result directory.
QC_START_TIME	Test	Date and time the running test started running.
QC_START_TIME_GMT	Test	Date and time the running test started in Greenwich mean time.
QC_RESULT_DIR	Test	Result directory of the running test.
QC_UNEXPECTED	Test procedure	Set if the outcome of the running procedure was unexpected.

Analyzing Tests in the Job Information Center

Job results can be viewed and analyzed in the **Job Information Center**. When analyzing the test results of one job or comparing the results of two jobs, it is helpful to understand how *QA*Director determines if a test procedure passed, failed, or was Not Executed, and also if the results were expected and unexpected:

Pass: The outcome of an automated test procedure is *Pass* unless a condition occurs to cause the outcome to be Fail or Not Executed.

Fail: The outcome of an automated test procedure is *Fail* if any of the following occurs:

- The procedure's actual output was not the expected output.
- A script or pass/fail rule of the test procedure issues the qc_exec fail command. A qc_exec fail command in a branch of a script or pass/fail rule indicates that a condition occurred which should mark the test procedure Fail.
- A pass/fail rule failed by exiting with a non-zero status.

Not Executed: The outcome of an automated test procedure is *Not Executed* if any of the following occurs:

- A setup rule exits with a non-zero status.
- A setup rule issues the qc_exec fail command. In a setup rule branch, this indicates that a condition occurred that should mark the setup rule as failed. *QA*Director does not execute the script and marks the test procedure Not Executed.
- One or more scripts could not run.

Expected: A test result of **expected** occurs if the expected result is the same as the actual result. By default, the expected exit status of a test procedure is 0.

Unexpected: A test result of **unexpected** occurs if one or more scripts in the test procedure exited with an unexpected status.

About Analyzing Job Results

Job results can be viewed and analyzed in the **Job Information Center**. When analyzing the test results of one job or comparing the results of two jobs, it is helpful to understand how *QA*Director determines if a test procedure passed, failed, or was Not Executed, and also if the results were expected and unexpected:

Pass: The outcome of an automated test procedure is *Pass* unless a condition occurs to cause the outcome to be Fail or Not Executed.

Fail: The outcome of an automated test procedure is *Fail* if any of the following occurs:

- The procedure's actual output was not the expected output.
- A script or pass/fail rule of the test procedure issues the qc_exec fail command. A qc_exec fail command in a branch of a script or pass/fail rule indicates that a condition occurred which should mark the test procedure Fail.
- A pass/fail rule failed by exiting with a non-zero status.

Not Executed: The outcome of an automated test procedure is *Not Executed* if any of the following occurs:

- A setup rule exits with a non-zero status.
- A setup rule issues the qc_exec fail command. In a setup rule branch, this indicates that a condition occurred that should mark the setup rule as failed. *QA*Director does not execute the script and marks the test procedure Not Executed.
- One or more scripts could not run.

Expected: A test result of **expected** occurs if the expected result is the same as the actual result. By default, the expected exit status of a test procedure is 0.

Unexpected: A test result of **unexpected** occurs if one or more scripts in the test procedure exited with an unexpected status.

Changing Results

After a job has been run, add to the results by typing a description or comments associated with the test result. To change results, a job must already have been run.

To change a job result:

1. Go to the **Job Information Center.**
2. Click on the **Results** tab
3. Choose a test and click **Actions>Change Results**. The Change Result dialog box appears.
4. Select a result from the **Result Status** list.
5. If desired, type a description of the failure in the **Failure Description** field.
6. If desired, type any comments in the **Additional Comments** field.
7. Click **OK**.

Deleting Results

After a test is executed it may become no longer necessary to retain the results. When deleting result information containing *QA*Run scripts, the result information is also deleted from the *QA*Run database.

To delete a result:

1. Go to the **Job Information Center**.
2. Click on the **Results** tab.
3. Choose a test and click **Actions>Delete Result** or right click and choose **Delete Result**. The Result is deleted.

Result Attributes

Result attributes are name-value pairs that store results when you execute tests. Some result attributes describe test classes, test procedures, or both. Other result attributes describe the job that ran the tests. Result attributes apply to one specific job.

Note: All result attributes are available as environment variables while a test is running because *QA*Director automatically sets the attributes to environment variables before execution.

Result Attribute	Test Node	Value
QC_ABORTED	Job	Set if the job was aborted.
QC_QADIRECTOR_HOME	Job	Path to the Compuware installation that ran the job.
QC_ACT_EXP_FAIL	Each class	Number of test procedures in the class that were expected to fail and did fail.
QC_ACT_EXP_PASS	Each class	Number of test procedures in the class that were expected to pass and did pass.
QC_ACT_EXP_UNRUN	Each class	Number of test procedures in the class that were

		expected to be Not Executed and were Not Executed.
QC_ACT_FAIL	Each class	Number of test procedures in the class that failed, expected and unexpected.
QC_ACT_PASS	Each class	Number of test procedures in the class that passed, expected and unexpected.
QC_ACT_UNEXP_FAIL	Each class	Number of test procedures in the class that failed unexpectedly.
QC_ACT_UNEXP_PASS	Each class	Number of test procedures in the class that passed unexpectedly.
QC_ACT_UNEXP_UNRUN	Each class	Number of test procedures in the class that were Not Executed unexpectedly.
QC_ACT_UNRUN	Each class	Number of test procedures in the class that were Not Executed, expected and unexpected.
QC_ARCH_NAME	Job	Name of the architecture on which the job ran, such as pa-hpux10.20.
QC_ARCH_OS	Job	Name of the Compuware directory containing the *QA*Director binary for an architecture.
QC_ELAPSED_SECS	Each test	Number of seconds the test took to run.
QC_ELAPSED_TIME	Each test	Hours, minutes, and seconds the test took to run.
QC_END_TIME	Each test	Date and time the test finished.
QC_END_TIME_GMT	Each test	Date and time the current test ended in Greenwich mean time.
QC_EXP_FAIL	Each class	Number of tests procedures in the class that were expected to fail.
QC_EXP_PASS	Each class	Number of tests procedures in the class that were expected to pass.
QC_EXP_PCT_FAIL	Each class	Percent of test procedures in the class that were expected to fail and did fail.
QC_EXP_PCT_PASS	Each class	Percent of test procedures in the class that were expected to pass and did pass.
QC_EXP_PCT_UNRUN	Each class	Percent of test procedures in the class that were expected to be Not Executed and were Not Executed.

QC_EXP_UNRUN	Each class	Number of tests procedures in the class that were expected to be Not Executed.
QC_FAIL_OUTPUT	Each test	Name of the result detail (.out) file of the script or rule that failed. Set only if a script or rule failed.
QC_JOB_ELAPSED_MINS	Job	Number of minutes the job ran.
QC_JOB_ELAPSED_SECS	Job	Number of seconds the job ran.
QC_JOB_END	Job	Time the job finished.
QC_JOB_ID	Job	ID number of the job.
QC_JOB_MACHINES	Job	Machines on which the job is allowed to run.
QC_JOB_NAME	Job	Name of the job.
QC_JOB_OWNER	Job	User that ran the job.
QC_JOB_START	Job	Time the job started.
QC_JOB_TEST_IDS	Job	ID numbers of the tests in the job.
QC_JOB_TIME	Job	Time the job was scheduled to run.
QC_JOB_USE_DEFAULT_ENV	Job	Set if the job ran with the user's default environment.
QC_NCOMPLETED	Each class	Number of test procedures in the class that finished running.
QC_NDESCENDANTS	Each class	Number of descendants the class has.
QC_NOUTPUTS	Each test	Number of result detail (.out) files that the test produced during execution.
QC_NTESTS	Each class	Number of descendant test procedures in the test class.
QC_NUNEXPECT	Each test	Number of test procedures in the class that had unexpected outcomes.
QC_OUTPUT*n*	Each test	List of the result detail (.out) files, numbered in the order in which the test executed them starting from 1.
QC_RESULT	Procedure	One of the three outcomes: Pass, Fail, or Not Executed.
QC_SHELL	Each test	Type of shell in which the test ran.

QC_EXE_CLEANUP*n*	Each test	Cleanup commands numbered in the order the test executed them starting from 1.
QC_TEST_COUNT	Each test	Number of test procedures in the class that ran.
QC_EXE_NCLEANUPS	Each test	Number of cleanup commands that executed in the test.
QC_EXE_NPASSFAILS	Each test	Number of pass/fail commands that executed in this test.
QC_EXE_NSETUPS	Each test	Number of setup commands that executed in this test.
QC_EXE_PASSFAIL*n*	Each test	Pass/fail commands numbered in the order the test executed them starting from 1.
QC_EXE_SETUP*n*	Each test	Setup commands numbered in the order the test executed them starting from 1.
QC_FAIL_DESC	Procedure	Description of the failure of the test procedure, if it failed.
QC_JOB_ID	Job	ID number of the job.
QC_JOB_MACHINES	Job	List of machines designated in the job description.
QC_JOB_NAME	Job	Name of the job.
QC_JOB_TIME	Job	Time the job started.
QC_LICENSE_HOST	Job	Name of the machine on which the *QA*Director license was running.
QC_LICENSE_USER	Job	Name of the user of the *QA*Director license.
QC_MACHINE	Procedure	Machine on which the test ran.
QC_OS_NAME	Job	Operating system of the machine on which the job ran.
QC_RESULT_TOP	Job	Absolute path to the result directory.
QC_START_TIME	Each test	Date and time the test started running.
QC_START_TIME_GMT	Each test	Date and time the test started in Greenwich mean time.
QC_RESULT_DIR	Each test	Result directory of the test.
QC_TIMERNAMES	Each test	Space-separated list of names of all timers used in the test

QC_TIMER_*name*	Each test	For each timer of name, the average number of seconds elapsed over each use of the timer in the test
QC_TIMER_*name*_MIN	Each test	For each timer of name, the elapsed time of the shortest use of the timer in the test
QC_TIMER_*name*_MAX	Each test	For each timer of name, the elapsed time of the longest use of the time in the job or test
QC_UNEXP_PCT_FAIL	Each class	Percent of test procedures in the class that were not expected to fail, but did fail.
QC_UNEXP_PCT_PASS	Each class	Percent of test procedures in the class that were not expected to pass, but did pass.
QC_UNEXP_PCT_UNRUN	Each class	Percent of test procedures in the class that were not expected to be Not Executed, but were Not Executed.
QC_UNEXPECTED	Procedure	Set if the outcome of the test procedure was unexpected.
QC_VERSION	Job	Version of *QA*Director that ran the job.
QC_VISIBLE_IDS	Procedure	ID numbers of test procedures listed to run.

Previewing the JCL

Use the compare log as a tool for analyzing tests.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To request a compare log and preview JCL:

1. Open the **Job Information Center.**
2. Double click a job. The Job Results screen appears.
3. Double click on a job result. The Test Procedure dialog box appears.
4. Click on the **Preview** tab.

Viewing Job Results

After running the job from the **Suite Information Center**, see the job results in the **Job Information Center**. The following process explains how to review the job results and details of the job. For a detailed discussion of the job results see Analyzing Job Results.

To review the job results:

1. After running the job, go into the **Job Information Center**.

2. Click on the **Results** tab.
3. Choose the test and click **Actions>View Results**, or right click and choose **View Results**. The Job Results screen appears.

On the **Job Results screen** right click on the different levels to view pertinent information about the job:

View Details: Depending on the level, the Properties or Procedure dialog boxes appear.

Change Results: The Change Result dialog box appears.

View History: The Result Change History dialog box appears.

Submit Defect: Click this to submit the defect. A message appears stating that the defect was successfully submitted with a defect number.

Rerun All: The Job Description dialog box appears for rerunning the job.

Rerun Failed: The Job Description dialog box appears for rerunning the job.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Viewing Summary Results

The results for the current test suite are stored in folders in the **Job Information Center**. Folders are located at the top of the **Jobs** panel. As soon as a job starts running, *QA*Director continuously tracks the job's progress in the **Results** tab. To access the **Results** tab, right-click a job in the **Execution** tab of the **Job** panel, and select **View Results**. By default, the **Results** tab displays the following information about each job:

Job name

Status

Total number of tests

Number of completed tests

Number of passed, failed, and Not Executed tests

Number or remaining tests

Tip: To change the type of information displayed in the **Job Information Center**, right-click on the selected tab and select **Customize** from the menu.
Note: The results in the **Results** tab are always displayed in the order that they were run.

Viewing Result Change History

After a job has been run, you can add to the results by typing a description or comments associated with the test result. These changed job results can be viewed and analyzed in the **Job Information Center** and used to track tests.

To view a history of result changes:

1. Go to the **Job Information Center.**
2. Click on the **Results** tab and select a result.
3. Click **Actions>View History**. The Result Change History dialog box appears.
4. If it is necessary to change a result, select a result and click **Edit**. The Change Result dialog box appears.

Viewing Detailed Job Results From the Job Information Center

After a job has completed running, go to the Job Information to view detailed information for the job.

To view job result details from the Job Information Center:

1. From the **Job Information Center**, click on the **Results** tab.
2. Double click on the job, or right-click and choose **View Results**. The results appear.
3. Click **Actions>View Details,** or right-click within the **Results** and select **View** Details from the shortcut menu. The Test Procedure dialog box appears.
4. Depending on which testing tool you are using and certain other factors, you may see icons for the following result files:

   run.out: For test procedures whose scripts executed, this file contains the input and output of the scripts.

   transcript.log: For each test that ran, *QA*Director creates a file to contain the chronological record of all the commands that the test issued, including the setting of each variable that prepared the environment for the test.

   ***type_of_rule_n*.out**: For each setup, pass/fail, and cleanup rule that executed, *QA*Director creates a type_of_rule_n.out file to contain the input and output of the rule as it executed. The *type_of_rule* is setup, pass-fail, or cleanup and *n* counts the number of rules of the type_of_rule that executed within a single test. For example, *QA*Director names a setup rule setup_3.out if it is the third setup rule that a test runs during test execution.

   ***scriptname*.TP-view**: Links to TestPartner's log view.

   ***scriptname*.QARun-view**: Links to *QA*Run's log view.

   ***scriptname*.QALoad-view**: Links to *QA*Load's Analyze component.

   ***scriptname*.FACmp-view**: Links to File-AID/CS Compare log view.

   ***scriptname*.MT-view**: Links to detailed information about the manual test.

   ***scriptname*.log**: Links to the Compare Log in *QA*Hiperstation+.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Viewing Job Result Details From the Defect Information Center

After a job has been run, and defects have been entered, go the Defect Information Center to look at details of these defects.

To view job result details from the Defect Information Center:

1. Go to the **Defect Information Center**.
2. Choose a suite or job from the **Jobs** panel
3. A list of tests with defects associated with these jobs appears in the **Defects** panel.
4. Right-click a job in the **Defects** panel, and select **View Test Result Detail**. The Test Procedure dialog box appears.
5. Double-click on any of the attached logs and files to view detailed results.

   Note: This feature is not available if using *QA*Director with CARS Workbench.

Result Folders

About Result Folders

The test results for the current suite are stored in folders at the top of the **Job Information Center.** By default, four folders appear:

User folder: This folder has the same name as the current user name. All job results are initially copied to this folder, but can be moved to different folders later. Also, an administrator or the suite owner can create a public or private default folder in which all results are routed to it instead of the user's folder.

BaseLine: This folder is typically used to store a baseline set of results against which subsequent results can be compared.

Personal: This folder contains any results that can't be shared with others.

Trash: Use this folder to store obsolete results. Right-click the folders panel and choose **Empty Trash** to delete the results from the database. Note that when moving results from *QA*Run tests to the Trash folder and emptying the trash, the result information is also deleted from the *QA*Run database.

Changing Result Folders Pane View

Change the **Result Folders** pane view to a list view or an icon view. You can also resize the **Results Folder** pane by clicking on the bottom pane border and dragging it to the preferred size.

To change the result folder pane view:

1. Open the **Job Information Center**.
2. Right-click in the **Result Folders** pane.
3. Click either **View>List** or **View>Icons** from the right-click menu.

Arranging Result Folders

It is possible to arrange icons by date or name. By default, the result folders are organized by date entered. It is also possible to drag and drop folders into a preferred custom order. Click the folder to move and drag it to the new location.

To change the result folder organization:

1. Open the **Job Information Center**.
2. Right-click in the **Result Folders** pane.
3. Click either **Sort>Name** or **Sort>Date Entered** from the right-click menu to arrange the folders alphabetically by name or chronologically by date entered.

Sorting the folders retains the update until the next time you select a sort type from the menu. If you add folders, you must select the sort option again to refresh the view to your sort preference.

Adding Results Folders

Add new, custom folders to the **Folder** panel. It is also possible to display other users' custom folders if those folders have been made public. This is helpful for archiving results to use at a later time.

To add a result folder:

1. Click the **Job Information** icon from the **Centers** toolbar.
2. Right-click the folders portion of the **Jobs** panel and select **Manage Job Folders**. The Manage Job Folders dialog box appears.

This dialog box displays any custom folders that have been added and any public folders added by other users. To add another user's folder to the Jobs panel, select the check box next to the folder. If the folder doesn't appear in the list, the owner has not made the folder public. Administrators can select the **Administrator's View** check box to view all user result folders.

Note: Without administrative privileges, the **Administrator's View** check box is disabled.

3. Click **New** to add a new, custom folder. The **Add New Results Folder** dialog box appears.
4. In the **Folder Name** field, type the name of the new folder.
5. Select the **Public** option to allow others to view this folder, or select **Private** to prevent others from using this folder.
6. Click **OK**.
7. Administrators and suite owners can create default folders. To create a default folder:

   a. Select the suite from the **Suite Name** field if the folder exists in another suite.

   b. Select the folder from the **Public Folders** field.
8. On the **Results Folder** dialog box, select the check box next to the new folder.
9. Click **OK**. The new folder appears in the Result Folder pane.

All folders appear in the order they were added. Use the right-click menu to change the order or the view.

Editing or Deleting Results Folders

It is possible for a folder owner to delete a folder that is no longer used or to change a folder's public/private status. If the deleted folder contains results, the results move to the **Admin** folder.

To edit or delete results folders:

1. Go the **Job Information Center**.
2. Right-click in the top of the center where the folders are located and select **Manage Job Folders**. The Manage Job Folders dialog box appears.
3. Select the folder to edit or delete.
4. Click **Edit** or **Delete** .
5. Click **OK**.

Moving Jobs and Results to Folders

Do the following from the **Results** tab to move a job or result to a folder or drag and drop the job or result into the destination folder.

To move jobs or results to another results folder:

1. Go to the **Job Information Center**.
2. Click the **Results** tab.
3. Right-click the desired result or job and choose **Move to** from the menu. The Move to Folder dialog box appears.
4. Select a destination folder from the **Move to folder list** field.
5. Click **OK**. *QA*Director moves the job or result to the destination folder.

Tracking Defects in the Defect Information Center

About Tracking Defects

Submit a defect using *QA*Director in conjunction with a defect tracking application to track failed scripts in a test application. Use the Defect Information Center to view and edit defects associated with tests in the project. These defects are submitted through the Job Information Center.

By default, the **Defect Information Center** opens to the first suite and its defects.

Note: This feature is not available if using *QA*Director with CARS Workbench.

About Tracking Defects

Submit a defect using *QA*Director in conjunction with a defect tracking application to track failed scripts in a test application. Use the Defect Information Center to view and edit defects associated with tests in the project. These defects are submitted through the Job Information Center.

By default, the **Defect Information Center** opens to the first suite and its defects.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Submitting a Defect

Submit a defect to track failed scripts in a test application.

To submit a defect for a particular test procedure, or a script:

1. From the **Centers** toolbar, click the **Job Information Center** icon.
2. Click the **Results** tab or from the **Execution** tab, right-click the job and select **View Results**.
3. Select and right-click the test procedure, or script that failed and select **Submit Defect**.

Note: When submitting a defect from *QA*Director to TrackRecord, the *QA*Director fields "Entered By" and "Priority" do not appear in TrackRecord.

Note: If the Defect Management option is not selected and saved in the **Project Information Center**, the Submit Defect function is not available.

Editing and Closing Defects

Customize the information in a defect or close it if it has been resolved.

To edit or close a defect:

1. From the **Centers** toolbar, click the **Defect Information Center** icon.
2. Select a job that failed from the **Job** list.
3. Right-click the associated defect in the **Defect** list and select **Edit Defect**. The application used to create the defect opens and displays the defect.
4. Modify or close the defect if appropriate.
5. Exit the application and return to *QA*Director.

Deleting a Defect Association

Delete a defect association to remove the ID from the test and database.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To delete a defect association:

1. Choose a job from the **Job** list.
2. Right-click a defect from the **Defect** list and select **Delete Defect Association**. A message displays to confirm the deletion.
3. Click **OK**. *QA*Director removes the defect ID from the specified test.

Using TrackRecord

TrackRecord is Compuware's defect-tracking tool for distributed applications. Use *QA*Director in conjunction with TrackRecord to easily track defects in a test application. For example, if a failed *QA*Director test reveals a defect in the application, submit the defect right from *QA*Director. *QA*Director will automatically enter relevant information about the failed test in the new TrackRecord defect item, reducing the amount of required data entry. The failed test is linked to the TrackRecord defect by means of a defect ID. After the test passes, edit the defect in TrackRecord.

*QA*Director projects that are integrated with TrackRecord and are migrated to CARS Workbench 5.1 can still be used with TrackRecord. When the 'Saved As' functionality is used on the migrated *QA*Director projects containing TrackRecord integrations, the project will cease to support TrackRecord. In this case, you can only select Request Management as the defect management integration.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Integrating *QA*Director with a Customized TrackRecord Database

When *QA*Director submits a defect into TrackRecord, it will attempt to import data into two types that are shipped with TrackRecord's default schema: *QA*Center Reported Defect and *QA*Director Test.

Note: This feature is not available if using *QA*Director with CARS Workbench.

In *QA*Director's Custom\win32 directory, there are two integration files for TrackRecord:

trqacenter_item.txt contains the mapping of *QA*Director attributes to tags in the TrackRecord *QA*Center Reported Defect type.

trqadirector_item.txt contains the mapping of *QA*Director attributes to tags in the TrackRecord *QA*Director Test type.

To set up *QA*Director's custom integration with a customized TrackRecord database:

1. Customize the TrackRecord database. Complete all editing before adding data.
2. Associate the QC_Defect tag with the *QA*Center Reported Defect type or its equivalent. This will identify this type to *QA*Director as the defect type to which it should send its information.
3. Associate the QC_Test tag with the *QA*Director Test type or its equivalent. This will identify this type to *QA*Director as the test information type contained in TrackRecord.
4. Open the trqacenter_item.txt and trqadirector_item.txt files.
5. In TrackRecord, apply any field tags listed in the files that you wish to use in the types.
6. If there are any unused tags that are listed in the *QA*Director text files, delete the attribute and tag lines. Each attribute must be mapped to a tag. If there are any tags that appear in these files that are not used, the integration will not work.

7. Save and close the trqacenter_item.txt and trqadirector_item.txt files.
8. In TrackRecord's *QA*Center Reported Defect type, ensure that the QC_Defect_Test field tag is applied to a single-item combination box and that field is pointed to the *QA*Director Test type.

   In the default schema shipped with TrackRecord, this field tag is associated with the Test single-item combination box. This will create a link between the *QA*Center Reported Defect and the *QA*Director Test types and allow TrackRecord to pass its internal database identifier to *QA*Director. When the integration is finished, there will be a link to the corresponding *QA*Center Reported Defect in each test submitted from *QA*Director.
9. Test the integration by submitting a defect from *QA*Director to TrackRecord.

Tips For Troubleshooting the TrackRecord Integration

Use the following tips to help with integrating with TrackRecord, Compuware's defect tracking tool:

Note: This feature is not available if using *QA*Director with CARS Workbench.

If something is wrong with the *QA*Director test integration file, a message appears stating there has been an error creating *QA*Center items. Ensure that all tags and attributes are matched up and all TrackRecord tags in the trqadirector_item.txt file are present in the TrackRecord *QA*Director Test type

If there is an error in the trqacenter_item.txt file, a message appears stating that there is a Path/File access error. Ensure that all tags listed in trqacenter_item.txt are present in the TrackRecord *QA*Center Reported Defect type. Also, verify that the field containing the QC_Defect_Test field tag is referencing the *QA*Director Test type.

If there is an error with the temporary files that are created each time a defect is submitted, a message similar to the following is received:

```
Trackrecord ProcessItemChanges fail.
Error in process file: C:\DOCUME~1\ADMINI~1.PAU\LOCALS~1\Temp\QATRitemfile.2
```

*QA*Director creates two temporary files, qatritemfile.1 and qatritemfile.2, each time a defect is submitted into TrackRecord. These files are stored in a temp directory. If an error occurs, immediately copy these files to a different directory to prevent them from being overwritten. They contain the text that *QA*Director is attempting to import into TrackRecord. Comparing the information that is imported into TrackRecord with the text in these two files should indicate at what point the error occurred .

About Using Changepoint Request Management

*QA*Director integrates with Changepoint Request Management to submit, edit and delete defects.

To use Changepoint Request Management, create an integration between *QA*Director and Changepoint Request Management. The Changepoint Request Management application opens and the Changepoint Request Management Submit Defect dialog box appears. Job information is automatically entered on the tabs for Request Details, Details, History, and Other Information. This information can be edited.

To integrate with Changepoint Request Management, install and configure *QA*Center Portal to work with the *QA*Center database.

Prior to submitting a defect using Changepoint Request Management, it is recommended that you map projects between *QA*Center Portal and Changepoint Request Management. For further information refer to the Changepoint Request Management documentation.

When you save the defect, a message appears to verify that the defect was successfully submitted. The defect appears in the **Defect Information Center** where you can edit it.

Note: If installing *QA*Director from the CARS CD, TrackRecord is not supported. However, if projects are migrated from 05.00.01 to 05.01.00, and already contain TrackRecord integrations, these **are** supported.

If installing *QA*Director from the *QA*Center CD, both TrackRecord and Request Management are supported.

Managing Changepoint Request Management Integrations

*QA*Director integrates with Changepoint Request Management to submit, edit and delete defects for systems and services. Only one Changepoint Request Management integration can be created. After a Request Management integration has been established, the **New Product Integration** option is not available until the integration is deleted.

Use this process to integrate with Request Management.

To set up a Changepoint Request Management integration with *QA*Director:

1. Open the **Project Information Center.**
2. Select **Defect Management** in the Project panel.
3. Select **Request Management**, and click **Browse**, or click **Tools>Manage Product Integrations**. The Manage Product Integrations dialog box appears.
4. Choose the **Request Management** folder from the **Integrations** list.
5. Click **New Product Integration**.
6. Type an integration name in the **Name** field.
7. Choose **Request Management** from the **Type** list.
8. Type the URL address where **Request Management** resides.
9. Type the following information in the appropriate fields:
    - **Server** field: *Changepoint server path*
    - **Database** field: *Changepoint database name*
    - **User** field: *User name for the Changepoint database.*
    - **Password** field: *Password for the Changepoint database.*
- Click the **Save** icon or **Save**.
- Click **OK**.

About Viewing Defects

View defects associated with a test from the **Defect Information Center.** The **View Defects** option is not available if Requirements Management was selected in the Project Management Center:

To view defects associated with a test:

1. From the **Centers** toolbar, click the **Defect Information Center** icon.
2. Choose the job that failed from the **Job** list.
3. Right-click the associated defect in the **Defect** list and choose **View Defect**. The View Defect dialog box appears.
4. Click **Close** to close the View Defect dialog box.

Customizing the View in the Defect Information Center

Change the columns in the **Defect Information Center** to customize views for individual projects. You can add or delete columns, and change the column order. You can also group the defects to see related items together, similar to an outline.

To change a view in the Defect Information Center:

1. From the **Centers** toolbar, click the **Defect Information Center** icon.

2. From the **Defect Tracking** panel **Action** menu, select **Customize View**. The Customize View dialog box appears.
3. Click **Columns**. The Show Columns dialog box appears.
4. To add a column: select a column from the **Available Columns** list and click **Add**.
5. To Remove a column: select a column from the **Show these columns in this order** list and click **Remove**.
6. To change the column order: select a column from the **Show these columns in this order** list and click either **Move Up** or **Move Down** until the column is in the desired position.
7. Click **OK** to save the changes and return to the Customize View dialog box.
8. Click **OK** to close the Customize View dialog box.

Sorting Defects

Sorting is a way of arranging defects in ascending or descending order. For example, sort defects by job name, then by test name to view all defects related to specific jobs and tests.

To sort defects:

1. From the **Centers** toolbar, click the **Defect Information Center** icon.
2. Click **Action>Customize View**. The Customize View dialog box appears.
3. Click **Sort**. The Sort dialog box appears.
4. In the **Sort items by** list, select a field to sort by.
5. Select **Ascending** or **Descending** for the sort order.
6. To sort by an additional field, select a field from the **Then by** list.
7. Click **OK** to save the changes and return to the Customize View dialog box.
8. Click **OK** to close the Customize View dialog box.

Note: Sorting by more than one field sorts all items by the first field and then, within that sort, sorts again by the second field. For example, if you choose to sort by job name and then by defect number, the sort structure is "Job A 1 2 3, Job B 1 2 3."

Grouping Defects

Grouping is a way of arranging defects to see related items together, similar to an outline. For example, group defects by job name to separate items according to their associated job. Expand or collapse the group headings to display or hide the items they contain.

To group the Defect Information Center view:

1. From the **Centers** toolbar, click the **Defect Information Center** icon.
2. From the **Defect Tracking** panel **Action** menu, select **Customize View**. The Customize View dialog box appears.
3. Click **Group By**. The Group By dialog box appears.
4. In the **Group items by** list, select a field to group by.
5. Select **Ascending** or **Descending** for the sort order of the group headings.
6. To group by subgroups, click a field in the **Then by** list.
7. Click **OK** to save the changes and return to the Customize View dialog box.
8. Click **OK** to close the Customize View dialog box.

Testing Servers

About Testing Servers

The Test Management Server (TMS) starts automatically when booting the computer and normally runs as a service or as a console application in the startup group. The Test Management Server reads job submissions from the database, manages jobs, tracks the machines available for test execution, and manages the licenses used for manual test Web execution. It also reports on the status of jobs in progress.

A Test Execution Server (TES) must be running on each machine where the tests run. In addition, a user must be logged onto the workstation in order for the TES to be active. In most cases, the TES should already be running. If it is not, click the **Start** button and choose **Programs>Compuware>*QA*Director>Test Execution Server**. In Windows XP, click the **Start** button and choose **All Programs>Compuware>*QA*Director>Test Execution Server**.

About Testing Servers

The Test Management Server (TMS) starts automatically when booting the computer and normally runs as a service or as a console application in the startup group. The Test Management Server reads job submissions from the database, manages jobs, tracks the machines available for test execution, and manages the licenses used for manual test Web execution. It also reports on the status of jobs in progress.

A Test Execution Server (TES) must be running on each machine where the tests run. In addition, a user must be logged onto the workstation in order for the TES to be active. In most cases, the TES should already be running. If it is not, click the **Start** button and choose **Programs>Compuware>*QA*Director>Test Execution Server**. In Windows XP, click the **Start** button and choose **All Programs>Compuware>*QA*Director>Test Execution Server**.

Setting Default Ports

*QA*Director executables require the use of ports. By default, *QA*Director randomly allocates ports to executables. However, some environments use firewalls that limit port usage.

To confine *QA*Director's port usage:

1. Open the System Administration Center.
2. From either the TM (Test Management) List panel or the Online TE (Test Execution) Servers panel choose **Action> Manage Ports**. The Test Execution Administration dialog box appears.
3. Enter port numbers in the appropriate files or check **Allocate Dynamically** by the appropriate port.
4. Click **OK**.

Test Execution Server

The Test Execution Server (TES) allows jobs to run on a computer. When requested by the Test Management Server (TMS), the TES starts the test driver, which subsequently starts the test engine. The TES also stores information about who is permitted to execute jobs on the computer, and whether the execution is remote or local.

In a typical team environment, install and start a TES on each machine where jobs will run.

Install the TES as a single component. If installed as a single component, configure a database by clicking **Start>Programs>Compuware>QADirector>DB Configuration Utility.** The Select a Database dialog box appears.

If connecting to an SQL Server, the TES is dependent on the SQL Server. The SQL Server must be started first.

To start the Test Execution Server:

Click **Start >Programs>Compuware>QADirector>Test Execution Server**. If using Microsoft Windows XP, choose **All Programs>Compuware>QADirector>Test Execution Server**. The TES appears as a satellite dish icon in the system tray.

The TES stores the test execution permissions, timeout settings, and ping interval settings for contacting the TMS.

Configuring the Test Execution Server

Configure the Test Execution Server (TES) to test locally or remotely. Via local testing, *QA*Director runs a job locally without binding scripts. Via remote testing, *QA*Director runs a job remotely using a WAN. If a job is scheduled on a remote machine, *QA*Director gets data from the local TES to run remote jobs, and binds each script to the selected machine.

To configure the TES:

1. Ensure that the **TES** is running by clicking **Start >Programs>Compuware>*QA*Director>Test Execution Server**. If using Microsoft Windows XP, click **All Programs>Compuware>*QA*Director>Test Execution Server**. The **TES** appears as a hard drive icon in the system tray.
2. Right-click the icon that appears and choose **Change TE Type** from the menu. The Test Execution Server Configuration dialog box appears.
3. Choose the **Local TE** option to run jobs locally or the **Remote TE** option to run remote jobs across a WAN.
4. If testing remotely, click the up or down arrow on the **Test Execution Port** box to choose a Test Execution port. Clear the **Dynamic** check box if specifying a port in the **Test Execution Port** box.

   Select the **Dynamic** check box to have Windows assign an available Test Execution port.
5. Click **OK** to configure the TES. Any changes will take effect after the TES is restarted.

Editing a Test Execution Server

Use the Test Execution Server Administration dialog box to adjust the common configurations for the Test Execution Server. From this dialog box select the following for the machine: file, directory paths, permissions, and ports. In addition, determine ping test intervals between the TES and TMS.

To edit a Test Execution Server:

1. Open the **System Administration Center**.
2. Select the TES from the **TE Servers** panel.
3. Click the **Action** menu and select **Properties**.

4. The Test Execution Machine Properties dialog box appears.
5. Make the appropriate changes and click **OK**.

Stopping the Test Execution Server

When finished using a Test Execution Server (TES) for testing, shut it down automatically from the System Administration Center.

Shutting down a TES will remove it from the TES list. The TES cannot be restarted after this occurs. A better option is to choose Restart. This will shut down the TES and bring it up again.

To stop the Test Execution Server:

1. Open the **System Administration Center**.
2. Select a Test Execution Server from the Online TES panel.
3. From the Online TES panel **Actions** menu, choose **Shutdown**.

Restarting the Test Execution Server

It may be necessary to restart the Test Execution Server (TES) if there is a performance problem.

Shutting down a TES will remove the TES from the TES list. The TES cannot be restarted after this occurs. A better option is to choose Restart. This will shut down the TES and bring it up again.

To restart the Test Execution Server:

1. Open the **System Administration Center**.
2. Select a TES from the **Online TE Servers** panel.
3. From the **Online TE Servers** panel choose **Actions>Restart**.

Test Management Server

The Test Management Server (TMS) reads job submissions from the database, manages jobs, tracks the machines available for test execution, and manages the licenses used for manual test Web execution. It also reports on the status of jobs in progress.

If installed as a single component, configure a database by clicking **Start>Programs>Compuware>QADirector>DB Configuration Utility.** The Select a Database dialog box appears.

If connecting to an SQL Server, the TMS is dependent on the SQL Server. The SQL Server must be started first.

If the TMS is not running, start it before attempting to run a job. It is possible to start it on your computer or on another team member's computer.

Starting the Test Management Server

The TMS starts automatically when booting the computer and normally runs as a service or as a console application in the startup group. In a typical team environment, the TMS is active on one computer and manages multiple jobs. In the System Administration Center, it is possible to set up several TMS with different priorities. The prioritized TMS list appears in the TMS List panel. These Test Management Servers will start according to priority and only run on one machine at a time. For example, if a Test Management Server with the priority of 1 fails to start, the TMS with the priority of 2 will start in its place.

To start the TMS only, click **Start>Programs>Compuware>*QA*Director>Test Management Server**.

To start the TMS when it is installed as a Windows 2000 service:

1. Click **Start>Settings>Control Panel**.
2. Double-click **Administrative Services.**
3. Double-click **Services** to open the Services dialog box.
4. Select **Test Management Server** and click the **Start** button on the **Tool** bar.

To start the TMS when it is installed as an XP service:

1. Click **Start>Control Panel>Administrative Tools>Services**.
2. Select **Test Management Server** and click the **Start** button on the **Tool** bar.

To manually start the TMS from Services in Windows:

1. Navigate to the Control Panel.
2. Click **Administrative Tools>Settings>Services**.
3. Right click on the TMS in the menu and choose **Properties**.
4. Choose the following options on the **Recovery Tab** and type the following information:

   First Failure list: Choose Restart the Service

   Second Failure list: Choose Restart the Service

   Subsequent Failures list: Choose Take No Action

   Reset Fail Count After: Type 0

   Restart Service After: Type 3

Note: To add, delete, and manage Test Management Servers, click the **Action** menu from the **TM Server** panel and select the appropriate action.

Configuring the Test Management Server

When testing remotely, configure the Test Management Server (TMS) with the appropriate machine names and port numbers. Add, modify, or delete machines as necessary.

To configure the TMS:

1. Ensure that the **TES** is running by clicking **Start>Programs>Compuware>QADirector>Test Execution Server**. If using Microsoft Windows XP, choose **All Programs>Compuware>QADirector>Test Execution Server**. The **Test Execution Server** appears as a hard drive icon in the system tray.
2. Right-click the icon and choose **TM List** from the menu. The Test Management Configuration dialog box appears.
3. Add, update or delete a machine:

   To **add** a machine, type a machine name in the **Machine Name** field, and type a port number in the **Port field**, or select the port number using the arrows, and click **Add**. Make sure that the machine entered is accessible.

   To **update** an existing machine, select it from the **TM List**, modify it as necessary, and click **Update**.

   To **delete** an existing machine, select it from the **TM List**, and click **Delete**.

4. Click **OK**.

Adding a Test Management Server

The Test Management Server (TMS) reads job submissions from the database, manages jobs, tracks the machines available for test execution, and manages the licenses used for manual test Web execution. It also reports on the status of jobs in progress.

To add a Test Management Server to the TM Server List:

1. Open the **System Administration Center**.
2. From the TM Server List panel click **Action>Add**. The Add Test Management Server dialog box appears.
3. Type a machine name in the **Machine Name** field.
4. Select a priority from the **Priority** list.
5. Do one of the following:

   Enter a port number in the **Port** field. The TMS will always execute tests through this port.

   Check the **Allocate Dynamically** check box. *QA*Director chooses a random port on which to execute tests.

   Check the **Use Default** check box. *QA*Director uses the default port set in the Test Execution Administration dialog box to execute tests.
6. Click **OK**.

Editing a Test Management Server

The TMS starts automatically when booting the computer and normally runs as a service or as a console application in the startup group. In a typical team environment, the TMS is active on one computer and manages multiple jobs. In the System Administration Center, it is possible to set up several TMS with different priorities.

Edit information such as the machine name, its priority in starting, and the port it is using.

To edit a Test Management Server:

1. Open the **System Administration Center**.
2. Select the TMS from the TM Servers panel.
3. Click **Action > Properties**.
4. The Test Management Server Properties dialog box appears.
5. Make the appropriate changes and click **OK**.

Deleting a Test Management Server

It is possible to delete a Test Management Server (TMS) from the list of Test Management Servers to change a configuration if necessary.

To delete a Test Management Server:

1. Open the **System Administration Center.**
2. Select the TMS from the **TM Server** List panel
3. Click **Action>Delete**.

4. *QA*Director removes the TMS from the list and reorders the remaining Test Management Servers according to their priorities.

Activating the Toggle Fallback

When the listed Test Management Servers are not available, the Toggle Fallback option allows *QA*Director to select any available Test Management Server.

To activate the Toggle Fallback option:

1. Open the **System Administration Center**.
2. From the TM Server List panel click **Action>Toggle Fallback**. *QA*Director places a check mark next to the option to indicate that it is selected.

To deactivate the Toggle Fallback option, repeat the above steps. *QA*Director removes the check mark indicating that the option is no longer active.

Reports

*QA*Center Portal provides a comprehensive reporting facility that is consistent among the *QA*Center point products. It enables users, with various roles and responsibilities, to create and to view detail, summary, and cross-product reports.

In addition to the report features, *QA*Center Portal provides the framework for web-based views and functionality to the *QA*Center products. *QA*Center Portal supports defect tracking, as well as manual test execution and test planning. *QA*Center Portal ties together requirements planning, test planning, test execution, and defect tracking to provide users with a centralized view of application quality.

To access the *QA*Center Portal click **Reports>Launch *QA*Center Portal**

Refer to *QA*Center Portal's documentation set for additional information about specific features and functionality.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Reports

*QA*Center Portal provides a comprehensive reporting facility that is consistent among the *QA*Center point products. It enables users, with various roles and responsibilities, to create and to view detail, summary, and cross-product reports.

In addition to the report features, *QA*Center Portal provides the framework for web-based views and functionality to the *QA*Center products. *QA*Center Portal supports defect tracking, as well as manual test execution and test planning. *QA*Center Portal ties together requirements planning, test planning, test execution, and defect tracking to provide users with a centralized view of application quality.

To access the *QA*Center Portal click **Reports>Launch *QA*Center Portal**

Refer to *QA*Center Portal's documentation set for additional information about specific features and functionality.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Filters

About Filters

A filter searches the database and displays the information requested. To display specific items in a tree view or grid, use filters to customize the view. There are two types of filters. User filters are used in all of the following centers and components:

Suite Information Center

Script Information Center

Job Information Center

Test Library

The **User filter** is applied in the center or library by a user who logged in. The user filter can be applied to all subsequent suites in the **Suite Information Center**, scripts in the **Script Information Center**, jobs in the **Job Information Center**, and test assets in the **Test Library**. This filter will not affect the suites, scripts, jobs, and test assets in their respective Centers, and the **Test Library**, when other users access them.

In the **Suite Information Center**, a filter can be applied using the **Suite Level filter** as well. The **Suite Level filter** is applied to suites from the General tab of the **Properties dialog box**. This filter affects the particular suite for **all** users who use this suite.

About Filters

A filter searches the database and displays the information requested. To display specific items in a tree view or grid, use filters to customize the view. There are two types of filters. User filters are used in all of the following centers and components:

Suite Information Center

Script Information Center

Job Information Center

Test Library

The **User filter** is applied in the center or library by a user who logged in. The user filter can be applied to all subsequent suites in the **Suite Information Center**, scripts in the **Script Information Center**, jobs in the **Job Information Center**, and test assets in the **Test Library**. This filter will not affect the suites, scripts, jobs, and test assets in their respective Centers, and the **Test Library**, when other users access them.

In the **Suite Information Center**, a filter can be applied using the **Suite Level filter** as well. The **Suite Level filter** is applied to suites from the General tab of the **Properties dialog box**. This filter affects the particular suite for **all** users who use this suite.

Applying a Filter

From within a center or component, select the filter from the **Filter** icon, **Filter** list, or the **Filter** item on the **Action** or right-click menus.

Creating a Filter

Create a simple filter to search the database and display the information requested. To display specific items in a tree view or grid, create a filter to customize the view.

To create a new filter:

1. Click **Tools>Manage Filters** or click the **Filter** icon on the toolbar. The Manage Filters dialog box appears.
2. Click **New**. The Filter dialog box appears.

Tip: To open the Filter dialog box directly, click **File>New>Filter** or, within each center, select **Create New** from the **Filter** list.

On the **General** tab:

1. Type the filter name in the **Name** field. The specified criteria is saved with this name.
2. Type a filter description in the **Description** field.
3. In the **Applies To** section, select the appropriate centers or components where the filter is available.
4. Select an access privilege from the **Access Privilege** section.

Use the **Simple** tab to create basic filters. On the **Simple** tab:

1. Click the **Field** field and select an attribute/property from the menu. The menu is broken down into center and component categories. The attributes/properties are organized by the center or component in which they are used.

   Common Fields refers to the common attributes/properties of the chosen centers.

   Each Center has a sub-menu which lists the attributes/properties.

2. Click on a **Condition** field and choose a condition.
3. In the **Value** field, type a value.

   Depending on what was selected in the **Field** field, the **Value** field's functionality will change. For example, if `date` is selected in the **Field** field, the **Value** field will open a Calendar box when clicked. Conversely, if `name` is selected in the **Field** field, the **Value** field will accept text

4. If adding additional criteria to the filter query, select **AND** or **OR** from the **Operator** field, and on the next line, repeat steps 1 through 3.
5. After specifications for the filter are complete -- whether editing or creating a new filter -- returning to the General tab will show that the filter is locked. Delete the information and start again.

Filters appear in the **Filter List** on the General tab of the **Test Class** dialog box when this is accessed from the Suite level.

Deleting a Filter

When a filter is obsolete and no longer necessary delete it.

To delete a filter:

1. Click **Tools>Manage Filters** or click the **Filter** icon on the toolbar. The Manage Filter dialog box displays.
2. From the **Filters** list, select a filter.
3. Click **Delete**.
4. Click **Close** to close the Filter Manager dialog box.

Editing a Filter

Use an existing filter and tailor it for other tests.

To edit a filter:

1. Click **Tools>Manage Filters** or click the **Filter** icon on the toolbar. The Manage Filters dialog box appears.

2. Choose a filter name and description.
3. Click **Edit**. The Filter dialog box appears.
4. Make changes on the appropriate tabs and in the appropriate fields, and click **OK**.
5. Optional: On the Advanced tab, manually edit a query string in the **Query** field.
6. Click **Save** to save edits and close the Manage Filters dialog box.

Creating an Advanced Filter

Use the **Advanced** tab to create complex filters that include attributes and conditions which add more detailed information when filtering tests.

To create and advanced filter:

1. Click **Tools>Manage Filters** or click the **Filter** icon on the toolbar. The Filter dialog box appears.
2. Click **New**. The Filter dialog box appears.

Tip: To open the Filter dialog box directly, click **File**>**New**>**Filter** or, within each center, select **Create New** from the **Filter** list.

On the **General** tab:

1. Type the filter name in the **Name** field. The specified criteria is saved with this name.
2. Type a filter description in the **Description** field.
3. In the **Applies To** section, select the appropriate centers or components where the filter is available.
4. Select an access privilege from the **Access Privilege** section.

On the **Advanced** tab:

1. Select an attribute from the **Field** list.
2. Select a condition from the **Condition** list.
3. Type in any alphanumeric value in the Value field. The value is applied to the attribute according to the selected condition.
4. Click **Add**. If creating a new filter, the query string will appear in the **Query** field. If editing a filter, the existing query string located in the **Query** field will reflect the changes.

   A query statement appears in the following format:

   $FieldName$ [condition] "value"

   The components of a query statement are:

   $: precedes the field name. The dollar ($) is a key symbol. Do not use it when naming attributes.

   FieldName: the field name

   $: finds the field.

   [condition]: one of the comparison statements (==, !=, <,>, etc.)

   value: the user supplied alphanumeric value contained within quotes

   One space must be included between the field name and [condition] and another space must be placed between the [condition] and the value.

5. To add to the query string, click one of the following buttons:

AND: Adds the word "AND" to the string at the cursor position in the query field. Add the second statement. The two statements are connected with **AND**. The tests matching both statements display in the tree. For example, the following query searches for all tests without the name of "`myclass`" and containing the word "`proc`":

```
$NAME$ != "myclass" AND $KEYWORDS$ contains "proc"
```

OR: Adds the word "**OR**" to the string at the cursor position in the query field. Add the second statement. The two statements are connected with OR. The tests matching both statements display in the tree. For example, the following query searches for all tests without the name of "`myclass`" and containing the word "`proc`":

```
$NAME$ != "myclass" OR $KEYWORDS$ contains "proc"
```

(: Adds an open parenthesis to the query statement.

): Adds a close parenthesis to the query statement.

If creating a more complex query, use parenthesis to group queries. For example, the following query searches for all tests named "`myclass`" or all tests named "`myproc`" and containing a completed status:

```
$NAME$ == "myclass" OR ($NAME$ == "myproc" AND $STATE$ == "completed")
```

Note: If clicking **Add** after modifying a statement, the additional query statement will be placed after the cursor. Do not position the cursor in the middle of a query statement, otherwise the additional query string is placed in the middle of the existing query string.

6. To filter the value as case sensitive, select **Match Case**.

7. If specifying a wildcard in the value statement, select **Expand Wildcard**. The wildcard character is an asterisk (*).

8. Click **Validate** to test the stability of the query.

9. To clear the query string, click **Clear Query**.

10. Click **Save** to save the filter to the database and remain in the Filter dialog box or click **OK** to update the database and close the Filter dialog box.

11. Click **Cancel** to close the Filter dialog box. *QA*Director will prompt to save.

Operators

The following is a list of available filter operators.

Operator	Function
==	Items containing the exact attribute will appear in the tree view if the specified value is identical to the test attribute.
!=	All items not containing the exact attribute will appear in the tree view if the specified value is not the same as the attribute.
>	All items greater than the attribute will appear in the tree view if the specified value is greater than the attribute.
<	All tests less than the attribute will appear in the tree view if the specified value is less than the attribute.
>=	All tests greater than and equal to the attribute will appear in the tree view if the specified value is greater than of equal to the attribute.

<=	All tests less than and equal to the attribute will appear in the tree view if the specified value is less than or equal to the attribute.
Contains	The tests containing the attribute will appear in the tree view if the specified value is included in the entity.
Does not Contain	The tests that do not include the specified attribute will appear in the tree view if the specified value is not included in the entity.

Commands

Unlike standard commands, which are accessed and executed from the command line, global commands can be invoked within *QA*Director rules and scripts. You cannot invoke global commands outside of *QA*Director.

Global commands encompass jobs. However, a number of limitations exist when using these commands. All tests must be in the same job. If using the **run in parallel** option, the test must be part of the same class or suite. Tests cannot be dragged from different classes to the Job Description window in order to create a single job.

Some commands are used in combination with others. For more information about dependencies, refer to the dependency section of each command description.

Some uses of global commands are:

- Finding the elapsed time between events
- Imposing dependencies between tests
- Locking common resources
- Finding the state of events
- Synchronizing processes

Tip: You can invoke *QA*Director commands from *QA*Run scripts. Note that you must execute the *QA*Run scripts from *QA*Director in order for the commands to work. Refer to each command for more information.

It is important to have a full understanding of these commands before adding them to your tests.

Global Commands

Unlike standard commands, which are accessed and executed from the command line, global commands can be invoked within *QA*Director rules and scripts. You cannot invoke global commands outside of *QA*Director.

Global commands encompass jobs. However, a number of limitations exist when using these commands. All tests must be in the same job. If using the **run in parallel** option, the test must be part of the same class or suite. Tests cannot be dragged from different classes to the Job Description window in order to create a single job.

Some commands are used in combination with others. For more information about dependencies, refer to the dependency section of each command description.

Some uses of global commands are:

- Finding the elapsed time between events
- Imposing dependencies between tests
- Locking common resources
- Finding the state of events
- Synchronizing processes

Tip: You can invoke *QA*Director commands from *QA*Run scripts. Note that you must execute the *QA*Run scripts from *QA*Director in order for the commands to work. Refer to each command for more information.

It is important to have a full understanding of these commands before adding them to your tests.

*QA*Director Command

Starts the *QA*Director GUI.

Syntax

```
QADirector
```

Parameters

None available

Test Execution Commands

The default **Test Execution Server** settings are used on machines where specific settings are not configured. To change these default settings, use the **qc_tesrv_config** utility supplied with *QA*Director. The qc_tesrv_config utility is located in `\Program Files\Compuware\QADirector\x86-win32\bin`. The qc_tesrv_config utility has three switches:

-list: Displays the current default settings.

-add user: Adds a new permission record for the user. With the addition of a permission record, the user can run tests as different user.

-runas user1, user2, user3: Identifies which user names can be run by a single user. This can only be used with the **-add user** switch.

-times [mon|tue|wed|thu|fri|sat|sun] :hr_start:hr_end: Defines the day and time when the user can run tests. Times must be defined in a 24-hour format. For example, `-times mon, tues, wed:8:14` where 8 is 8:00 a.m. and 14 is 2:00 p.m. This can only be used with the **-add user** switch.

-remove user: Removes user permissions.

-cfgfile <filename>: Reads the filename and loads it into the *QA*Director database. To retrieve the file format, use **-dump** option and edit the file. This is especially useful when making small changes to the existing configuration.

-dump <file>: Creates a file named <*file*> that you can use to edit the default settings.

-cfg default: Uses the *QA*Director default settings for connection timout (**ConnTimeout**), request timeout (**RequestTimeout**), Test Management Server ping connection (**tmsrvPingConn**), and Test Management Server ping disconnect (**tmsrvPingDisconn**). *QA*Director will use these default settings until a user manually changes them.

-cfg tmpdir:ConnTimeout:RequestTimeout:tmsrvPingConn:tmsrevPingDisconn: Creates manual settings for the following:

> **tmsrvPingConn** = TMServer Ping Interval (connected)
>
> **tmsrvPingDisconn** = TMServer Ping Interval (disconnected)
>
> **RequestTimeout** = CLIPC Request Timeout
>
> **ConnTimeout** = CLIPC Connect Timeout

After using -cfg,-add,-cfgfile, or -remove, use the `qc_admin reinit` command to re-initialize the **Test Execution Server**.

qc_exec event_done

Determines whether an event is finished. Returns 1 if *name* is finished or 0 otherwise.

Syntax

```
qc_exec event_done <name>
```

Operation

Returns 1 if *name* is finished or 0 otherwise.

Use this command in a rule or in a batch file added to the procedure as a user-defined script. View the results in an output file (.out) in the results sub-directory. The following is a sample of a successful result:

```
Running local rule 'setup_1'
Shell: C:\WINNT\system32\cmd.exe
Command: qc_exec event_done abc
-----Output Start-----
1
-----Output End-----
Command executed with status: 1
Command Succeeded
```

As an example, place `qc_exec event_done` in a setup rule in a procedure `P1`. After running the test, view the result in a file named setup_1.out.

Dependencies

A qc_exec event_start command must be issued first.

Parameters

Parameter	Description
name	Specifies the name of the event.

qc_exec event_end

Marks the end of an event.

Syntax

```
qc_exec event_end <name>
```

Operation

Place this command in a setup rule or in a batch file added to the procedure as a user-defined script. The following example uses a batch file. After executing the test, view the results in the run.out file.

The batch file contains the following:

```
QC_EXEC event_start ABC
Notepad
QC_EXEC event_end ABC
```

The test result appears as follows:

```
Running local rule 'run'
Shell: C:\WINNT\system32\cmd.exe
Command: qc_run_scripts
-----Output Start-----
***Running Script EVENT1***
C:\Program
Files\Compuware\QADirector\SuiteDir\100.results\Admin\165\101>QC_EXEC
event_start ABC
C:\Program
Files\Compuware\QADirector\SuiteDir\100.results\Admin\165\101>NOTEPAD
C:\Program
Files\Compuware\QADirector\SuiteDir\100.results\Admin\165\101>QC_EXEC
event_end ABC
-----Output End-----
Command exited with status: 0
Command Succeeded (set via qc_exec commands)
```

Dependencies

A qc_exec event_start command must be issued first.

Parameters

Parameter	Description
name	Specifies the event you are ending.

qc_exec event_in_progress

Finds whether an event is in progress. Returns 1 if the event is in progress; otherwise 0.

Syntax

```
qc_exec event_in_progress <name>
```

Operation

Returns 1 if the event is in progress; otherwise 0.

Use this command in a rule or in a batch file added to the procedure as a user-defined script. View the results in an output file (.out) in the results sub-directory. The following is a sample of a successful result:

```
Running local rule 'run'
Shell: C:\WINNT40\system32\cmd.exe
Command: qc_run_scripts
-----Output Start-----
***Running Script event_in_progress***
```

```
C:\PROGRAM
FILES\COMPUWARE\QADirector\SUITEDIR\103.RESULTS\Admin\148\101>qc_exec
event_in_progress
1
-----Output End-----
Command exited with status: 0
Command Succeeded (set via qc_exec commands)
```

Dependencies

A qc_exec event_start command must be issued first.

Parameters

Parameter	Description
name	Specifies the name of the event.

qc_exec event_occurred

Determines whether an event is running or finished. Returns 1 if the event is either running or finished. Returns 0 otherwise

Syntax

```
qc_exec event_occurred <name>
```

Operation

Returns 1 if the event is either running or finished. Returns 0 otherwise.

Place this command in a rule or in a batch file added to the procedure as a user-defined script. View the result in the file named run.out. If you place this command in a *QA*Director user-defined script, this file is located in the results sub-directory. If you invoke this command from a rule, the results are in the .log file for that rule. The following is an example of a successful result:

```
Running local rule 'setup_1'
Shell: C:\WINNT\system32\cmd.exe
Command: qc_exec event_occurred ABC
-----Output Start-----
1
-----Output End-----
Command exited with status: 0
Command Succeeded
```

Dependencies

A qc_exec event_start command must be issued first.

Parameters

Parameter	Description
name	Specifies the name of the event.

qc_exec event_rendezvous

Pauses until the command is called the specified number of times.

Syntax

```
qc_exec event_rendezvous [-timeout <timeout>] <name> <count>
```

Operation

Use this command to synchronize a set of processes before proceeding.

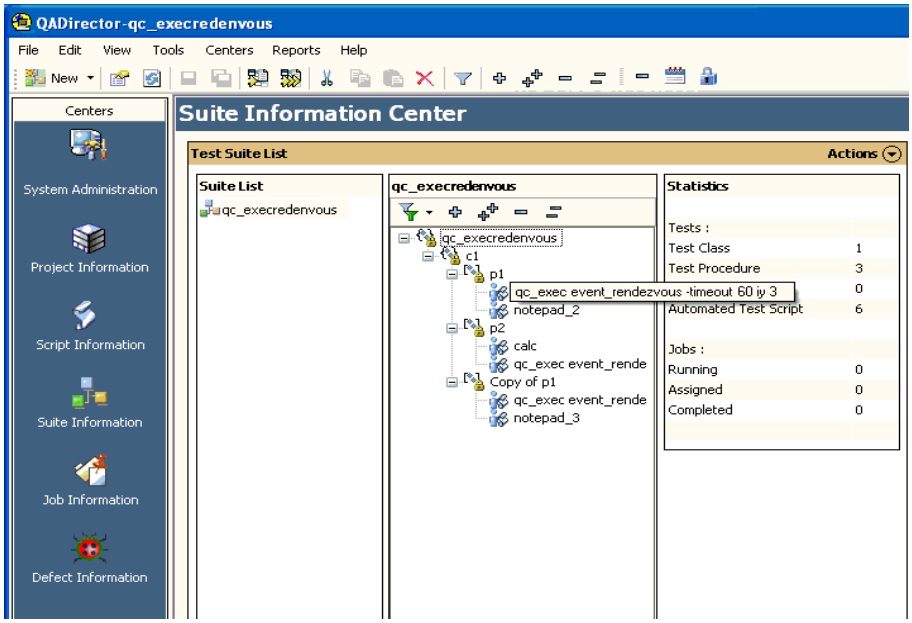
Error checking ensures that all processes use the same count.

Do not use a name used with any other command, like `event_start` or `event_end`. Use the specified name only with `event_rendezvous`.

If one of the processes that called `event_rendezvous` aborts, all `event_rendezvous` calls return with a failure to synchronize message. The *timeout* of event_rendezvous or the *timeout* of the process test could abort the process.

Dependencies

This example shows a simple suite with three procedures using the qc_exec event rendezvous command. The first script of each procedure contains the qc_exec event rendezvous command. The event name, the number of events, and the timeout value have been supplied. When you submit this suite, none of the tests run until all three reach the same point in their execution. In this example, all tests execute on the same machine.



Parameters

Parameter	Description
count	The number of times you want the rendezvous call invoked.
name	Specifies the name of the event.
-timeout <timeout>	Specifies the maximum number of seconds to wait.

qc_exec event_start

Marks the start of an event.

Syntax

```
qc_exec event_start <name>
```

Operation

Defines the start of an event during a job. Place this command in a setup rule or in a batch file added to the procedure as a user-defined script. Add a qc_exec event_end command to mark the end of the event.

Dependencies

None

Parameters

Parameter	Description
name	Specifies the name of the event you are marking.

qc_exec event_wait_for

Waits until an event is finished.

Syntax

```
qc_exec event_wait_for [-timeout <timeout>] <name>
```

Operation

Use this command to arrange the order of events. Place this command in a setup rule or in a batch file added to the procedure as a user-defined script. View the results in files stored in the results sub-directory. The following is a sample result:

```
Running local rule 'setup_1'
Shell: C:\WINNT\system32\cmd.exe
Command: qc_exec event_wait_for -timeout 60 abc
-----Output Start-----
-----Output End-----
```

```
Command exited with status: 0
Command Succeeded
```

In this example with two procedures P1 and P2, place `qc_exec event_wait_for` in the setup rule of the procedure P2 to ensure that P2 runs after P1. Execute the tests with the **Run in Parallel** option turned on. The above example displays the results of P2.

Dependencies

The qc_exec event_start and qc_exec event_end commands must be issued first.

Parameters

Parameter	Description
name	Specifies the name of the event.
-timeout <timeout>	Identifies the maximum number of seconds to wait as specified by *timeout*.

qc_exec fail

Marks the test as failed and sets a result attribute to contain the failure description.

Syntax

```
qc_exec fail failure_description [-file <file>]
```

Operation

If you issue this command from a *QA*Director script, the script is marked as failed. If you issue this command from a pass/fail rule, the procedure is marked as failed. You cannot use this command in a setup or cleanup rule.

Dependencies

None

Example

test.bat contains the following:

```
set > c:\test.txt
qc_exec fail 'test failed'
```

Add the test.bat file to a procedure as a user-defined script. Use a fully qualified path for test.bat. After the procedure containing test.bat executes, you should see a failed result with the status of test failed.

Parameters

Parameter	Description
failure_description	Describes the reason the command failed.

-file <file>	Defines the file the UNIX GUI opens when you double-click the test in the Results Browser as specified by *file*. On Windows, defines the file that the GUI opens when you double-click the test in the **Result** tab in the Tree View and press **More Info**.

Example

This example marks the test as failed with a descriptive failure message:

```
qc_exec fail 'Test failed. Check log for data.'
```

qc_exec get_attr

Returns the value of the attribute name for the specified test ID.

Syntax

```
qc_exec get_attr <test_id> <attribute_name>
```

Operation

Returns the value of the specified attribute for the given test id. If you give this command as a setup rule, *QA*Director creates a setup_<n>.out file in the result directory of the test procedure with the value of the attribute.

Parameters

Parameter	Description
id	Specifies the test ID.
attribute_name	Specifies the name of the attribute.

Dependencies

None

Example

This example retrieves the **QC_STATE** attribute value for test 101:

```
qc_exec get_attr 101 QC_STATE
```

The following are the results for `qc_exec get_attr`*:*

```
Running local rule 'setup_1'
Shell: C:\WINNT\system32\cmd.exe
Command: qc_exec get_attr 101 QC_STATE
------ Output Start -------
Unwritten
------- Output End --------
Command exited with status: 0
Command Succeeded
```

qc_exec lock_get

Acquires a lock. See Locking Resources.

Syntax

```
qc_exec lock_get <lock_name>
```

Operation

Once *QA*Director acquires a lock, it holds the lock until it is released or the test ends. Only one test within a job can hold a lock. Another test within the same job receives an error if it tries to acquire a lock that a test already holds.

The request fails if another test already has the lock.

Dependencies

None

Examples

In this example, the following command is added to a test as a setup rule:

```
qc_exec lock_get walter
```

This acquires a lock named "`walter`". If another test within the same job attempts to acquire a lock with the same name, that test fails with a status of `Not Executed`. An error message appears stating:

```
“Error: Lock ‘walter’ unavailable (held by test ID 101).”
```

The test ID differs in each instance.

Parameters

Parameter	Description
lock_name	Specifies the name of the lock you are acquiring.

qc_exec lock_get_wait

Acquires a lock or waits for a lock. See Locking Resources.

Syntax

```
qc_exec lock_get_wait <lock_name> [-timeout <timeout>]
```

Operation

This command causes a test to wait if the lock is unavailable because another test already holds it. This command fails if waiting would cause a deadlock situation. If multiple tests are waiting for the same lock, the first test to issue this command gets the lock when it becomes available. Once a test acquires a lock, it holds the lock until the test ends or is released. Only one test within a job can hold a lock.

Parameters

Parameter	Description

lock_name	The name of the lock.
-timeout timeout	Specifies the maximum number of seconds to wait for the lock.

Dependencies

None

Examples

In this example, add the following command to a test as a setup rule:

```
qc_exec lock_get walter –timeout 5
```

This acquires a lock named "`walter`". If another test within the same job already holds this lock, then this test will wait five seconds for the lock to become available.

qc_exec lock_owner

Returns the ID number or name of the test holding the lock. See Locking Resources.

Syntax

```
qc_exec lock_owner <lock_name> -name
```

Operation

The name or the ID of the test holding the lock is returned. An error is returned if the lock does not exist.

Dependencies

None

Examples

In this example, add the following command to a test as a setup rule:

```
qc_exec lock_owner walter
```

In this case, *QA*Director returns the ID of the test holding the lock in the setup output file. If the lock does not exist, the test fails with a status of `Not Executed`. An error message appears stating:

```
“Error: Lock ‘walter’ does not exist.”
```

Parameters

Parameter	Description
lock_name	Specifies the name of the lock.
-name	Returns output in the form of the test name.

qc_exec lock_release

Releases a lock. See Locking Resources.

Syntax

```
qc_exec lock_release <lock_name>
```

Operation

Releases the named lock. Returns an error if the lock does not exist or the test releasing the lock does not own the lock.

Parameters

Parameter	Description
lock_name	Specifies the name of the lock.

Dependencies

The lock must be acquired by the test issuing the release.

Examples

```
qc_exec lock_release walter
```

If the test issuing this command owns the lock, then *QA*Director releases the lock. If the issuing test does not own the lock or the lock does not exist, *QA*Director returns an error. The test fails with a status of `Not Executed`. An error message appears stating:

```
"Error: You are not holding lock 'walter'."
```

qc_exec lock_waitlist

Returns the ID number or names of the tests waiting for a lock. See Locking Resources.

Syntax

```
qc_exec lock_waitlist <lock_name> [-name]
```

Operation

Returns the test ID numbers of the tests waiting for a lock. If you specify the **-name** parameter, then *QA*Director returns the test names. Returns an error message if the lock does not exist.

Without the **-name** parameter, returns the ID numbers of the tests holding the lock.

Parameters

Parameter	Description
<lock_name>	The name of lock you are acquiring. The *lock_name* is case sensitive. For example, a *lock_name* of `walter` is different from a lock_name of `WALTER`.
-name	Indicates that the names of the tests are returned rather than the test ID

	numbers.

Examples

Consider the following suite structure:

```
ste1
c1(Run in parallel = procedures)
    p1 (setup rule:qc_exec lock_get testlock)
        (cleanup rule:qc_exec lock_release testlock)
      <Script: test.txt>
    p2
       <Script:notepad>
    p3   (setup rule: qc_exec lock_get_wait testlock -timeout 20)
       <Script:test.txt>
    p4
       <Script:notepad>
    p5   (setup rule: qc_exec lock_waitlist testlock)
       <Script:test.txt>
```

The test procedure's scripts and rules are bound to computers in the following manner:

PC1 – *p1*

PC2 – *p2,p3*

PC3 – *p4,p5*

The tests, *p1, p2* and *p4*, run in parallel on all 3 machines.

Test *p1* acquires a lock named `testlock` and executes test.txt. *QA*Director releases the lock when the test ends. Test *p2* executes notepad. Test *p3* executes after the notepad window closes. The setup rule for test *p3* waits for the lock, `testlock`, to become available. Test *p4* executes notepad. Test *p5* executes after the notepad window closes. The setup rule for test *p5* gets a list of the tests that are waiting on the lock named `testlock`. The following is the output of that command.

Results:

```
Running local rule 'setup_1'
Shell: C:\WINNT\system32\cmd.exe
Command: qc_exec lock_waitlist testlock
------ Output Start -------
109
------- Output End --------
Command exited with status: 0
Command Succeeded
```

qc_exec setenv

Sets the value of an environment variable.

Syntax

```
qc_exec setenv <name> <value>
```

Operation

Allows setting environment variables dynamically inside the test suite. These environment variables are visible only inside the suite. Modify the environment variable using the qc_exec unsetenv command.

Parameters

Parameter	Description
name	Names the environment variable you are setting.
value	Sets the environment variable you are setting to *value*.

Dependencies

None

Example

This example sets the **DEBUG** environment variable to 1:

```
qc_exec setenv DEBUG 1
```

This example adds c:\RW to the path:

```
qc_exec setenv PATH "c:\RW:$PATH"
```

qc_exec setres

Sets the value of a result attribute.

Syntax

qc_exec setres <name> <value>

Parameters

Parameter	Description
name	Name of the result attribute
value	Sets the environment variable

Operation

Allows setting a result attribute dynamically inside of the test suite.

Dependencies

None

Example

This example sets ***QC_EXP_PASS*** to 1.

```
qc_exec setres QC_EXP_PASS  1
```

Add a column named *Expected Pass* using the following command: `qc_exec add_col "Expected Pass" QC_EXP_PASS`. Issue the command in the example to set the value of the column for each test procedure. Opening the result in the **Result** pane will show the added column *Expected Pass* with a value of 1 at the node level where this command was issued.

qc_exec test_has_outcome

Pauses until a test finishes and reports when it finishes.

Syntax

```
qc_exec test_has_outcome -id <id> <results>
```

Operation

Use this command in a setup rule to verify that another test procedure performed as expected. For example, if a test procedure that loads a database fails, it might not make sense for any other tests to run.

If the test procedure has not yet finished, the qc_exec test_has_outcome command waits for the test to finish before reporting the outcome. Limit the wait time using qc_exec test_wait_for as a setup rule. That rule should have a reasonable timeout value.

If the outcome of the test appears in <results>, it returns with an exit status of 0. Otherwise, it returns with an exit status of 1.

Parameters

Parameter	Description
-id <id>	Identifies the test ID number as specified by *id*.
<results>	Specifies the outcome you want **qc_exec test_has_outcome** to check for. The value of *results* must be one or a combination of the following: **pass**, **fail**, **unrun**, **ux-pass**, **ux-fail**, or **ux-unrun**.

Dependencies

None.

Example

This example checks that the test with an ID of 101 executed with the expected outcome (space-separated results):

```
qc_exec test_has_outcome –id 101 pass fail unrun
```

This example checks that the test with an ID of 202 executed with an outcome of ux-unrun:

```
qc_exec test_has_outcome –id 202 ux-unrun
```

This example checks that the test with an ID of 7 passed:

```
qc_exec test_has_outcome -id 7 pass
```

qc_exec test_wait_for

Waits until a test finishes, ignoring its outcome.

Syntax

```
qc_exec test_wait_for -id <id>
```

Operation

This command is typically used in a setup rule and instructs *QA*Director to wait until the specified test procedure finishes before proceeding. The outcome of the waited-on test procedure is ignored.

Parameters

Parameter	Description
-id <id>	Identifies the test ID number as specified by id. Waits until this test finishes.

Dependencies

None

Example

Consider the following suite:

TestSuite

TestClass1

TestProc1 (id=102)

TestScript1

TestClass2

TestProc2 (setup rule qc_exec test_wait_for –id 102)

TestScript2

Set **Bind Scripts and Rules** of TestProc1 to Machine01.

Set **Bind Scripts and Rules** of TestProc2 to Machine02.

Set **Run in Parallel** property of TestSuite root class to “Child Classes and Procedures.”

Run the TestSuite. TestProc1’s TestScript1 executes first on Machine 01, but TestProc2’s TestScritp2 did not execute in parallel since it is waiting for TestProc1 to complete before executing.

Note: For this command to work, run a parent class holding both the test “waiting for” and the test “waited for.” In this example, run the root class TestSuite since it holds both TestProc1 and TestProc2, the conditional procedures.

For more information on how to run a job, see Running a Job.

qc_exec timer_assert

Determines if the elapsed time of the timer was greater than the specified number of seconds.

Syntax

```
qc_exec timer_assert <name> <time_value>
```

Operation

If the *time_value* specified is less than the elapsed time of the operation, the test fails. For example, the message accompanying a failure might say that the Timer "`walter`" value 20.828 exceeded 2.000. In this case, 2 is given as the *time_value* and the timing operation ran for 20.828 seconds. The test passes if the

time_value is greater than the elapsed time of the operation. You can not use this command on the cleanup rule

If there is more than one timer for the specified name, the average of those timers is used in the comparison.

Parameters

Parameter	Description
name	Specifies the name of an existing timer.
time_value	Specifies the number of seconds used in the comparison.

Dependencies

A valid timer must be specified.

Example

The following sample batch file shows starting a timer, a call opening notepad, ending the timer, and issuing a timer assert.

```
qc_exec timer_start walter
Call notepad.exe
qc_exec timer_end walter
qc_exec timer_assert walter 2
```

If the elapsed time between the start and end of the timer is greater than the specified time, 2 seconds, the test fails.

qc_exec timer_end

Indicates the end of a timing operation.

Syntax

```
qc_exec timer_end <name>
```

Parameters

Parameter	Description
name	Specifies the name of timer given in the qc_exec timer_start command.

Operation

Timing operations for the named timer end. The results of the timing operations are available in the timer.log file. See Finding the Elapsed Time between Events for more information.

Dependencies

A `qc_exec timer_start` command must be issued prior to this test.

qc_exec timer_getval

Returns the number of seconds elapsed during a timing operation.

Syntax

```
qc_exec timer_getval [-min | -max] [-sec] <name>
```

Operation

When used as a cleanup rule, *QA*Director returns the value in the file that gives the result of the rule's execution. If used within a script, *QA*Director returns the value in the file that gives the result of the script's execution. The value returned can be an integer when -sec is used. Otherwise, the value returned has three decimal places of precision.

*QA*Director returns a zero value if a `qc_exec timer_end` command was not issued for the specified timer or if a non-existing timer is given.

See Finding the Elapsed Time between Events for more information.

Parameters

Parameter	Description
-max	Returns the maximum value of a number of timing operations.
-min	Returns the minimum value of a number of timing operations.
name	Specifies the timer name.
-sec	Returns the value as an integer.

Dependencies

You must issue valid `qc_exec timer_start` and `qc_exec timer_end` commands for the specified timer.

Example

Create a simple user-defined script that executes notepad as follows:

1. Define a setup rule to start the timer.

```
qc_exec timer_start walter
```

2. Define a cleanup rule to end the timer.

```
qc_exec timer_end walter
```

3. Define another cleanup rule to get the value of the timer.

```
qc_exec timer_getval -sec walter
```

In the following example, the value of the timer is returned as an integer in a file named `clean_2.out` in the results subdirectory. The contents of that file look similar to the following:

```
Running local rule 'clean_2'
Shell: C:\WINNT\system32\cmd.exe
Command: qc_exec timer_getval -sec walter
```

```
------ Output Start -------
9
------- Output End --------
Command exited with status: 0
Command Succeeded
```

qc_exec timer_start

Indicates the beginning of a timing operation.

Syntax

```
qc_exec timer_start <name>
```

Parameters

Parameter	Description
name	Specifies the timer name.

Operation

Starts timing operations for the named timer. See Finding the Elapsed Time between Events for more information.

Dependencies

None

qc_exec unsetenv

Removes the value of an environment variable for the current test and all of its descendants.

Syntax

```
qc_exec unsetenv <name>
```

Parameters

Parameter	Description
name	Specifies the name of the environment variable.

Operation

This command works only for environment variables set in the test suite. It does not change environment variables set outside the test suite, such as those in the shell resource file.

Delete an environment variable that was set using qc_exec setenv or an environment variable created as part of rules inside of the test suite.

Dependencies

None

Example

This example releases the DEBUG environment variable that was set using the qc_exec setenv described above:

```
qc_exec unsetenv DEBUG
```

qc_exec subtest_fail

Ends a subtest and marks the subtest as failed with a specified reason.

Syntax

```
qc_exec subtest_fail [-name <name>] failure_description
```

Operation

Invoke the qc_exec subtest commands inside a script.

Use the qc_exec subtest_start command to break a test procedure's script into smaller units or subtests. Then, use the qc_exec subtest_fail command to fail the subtest depending upon a certain condition. The –**name** option causes the command to fail the specified subtest. Otherwise, the command fails the last subtest that was started. This eventually fails the test procedure.

See the example for qc_exec subtest_start. In this example, use the qc_exec subtest commands inside a *QA*Run script. The subtest passes or fails if it satisfies specified criteria

Parameters

Parameter	Description
<failure_description>	Describes the reason the subtest failed.
-name <name>	Name of the subtest to fail (optional).

Dependencies

qc_exec subtest_start

Example

This example ends the current subtest and marks it as failed with a descriptive message:

```
qc_exec subtest_fail 'Test failed. Check log for data.'
```

qc_exec subtest_pass

Ends a subtest of a script and marks it as passed.

Syntax

```
qc_exec subtest_pass [-name <name>]
```

Operation

Invoke the qc_exec subtest commands from inside of a script.

Use qc_exec subtest_start command to break a test procedure's script into smaller units or subtests. Then, use the qc_exec subtest_pass command to pass the subtest depending upon a certain condition. If you use the **–name** option, the command passes the specified subtest. Otherwise, the command passes the last subtest that was started.

See the given example for qc_exec subtest_start command. The subtest passes or fails if it satisfies specified criteria.

Parameters

Parameter	Description
-name <name>	Name of the subtest to pass (optional)

Dependencies

qc_exec subtest_start

Example

This example ends the current subtest and marks it as passed:

```
qc_exec subtest_pass
```

qc_exec subtest_start

Marks the beginning of a subtest with the specified name.

Syntax

```
qc_exec subtest_start <subtest_name>
```

Operation

Use this command to break up a test procedure's script into smaller units. Use the qc_exec subtest_pass command to pass a subtest or the qc_exc subtest_fail to fail a subtest.

Parameters

Parameter	Description
subtest_name	Specifies the name of the subtest you are starting.

Dependencies

None

Example

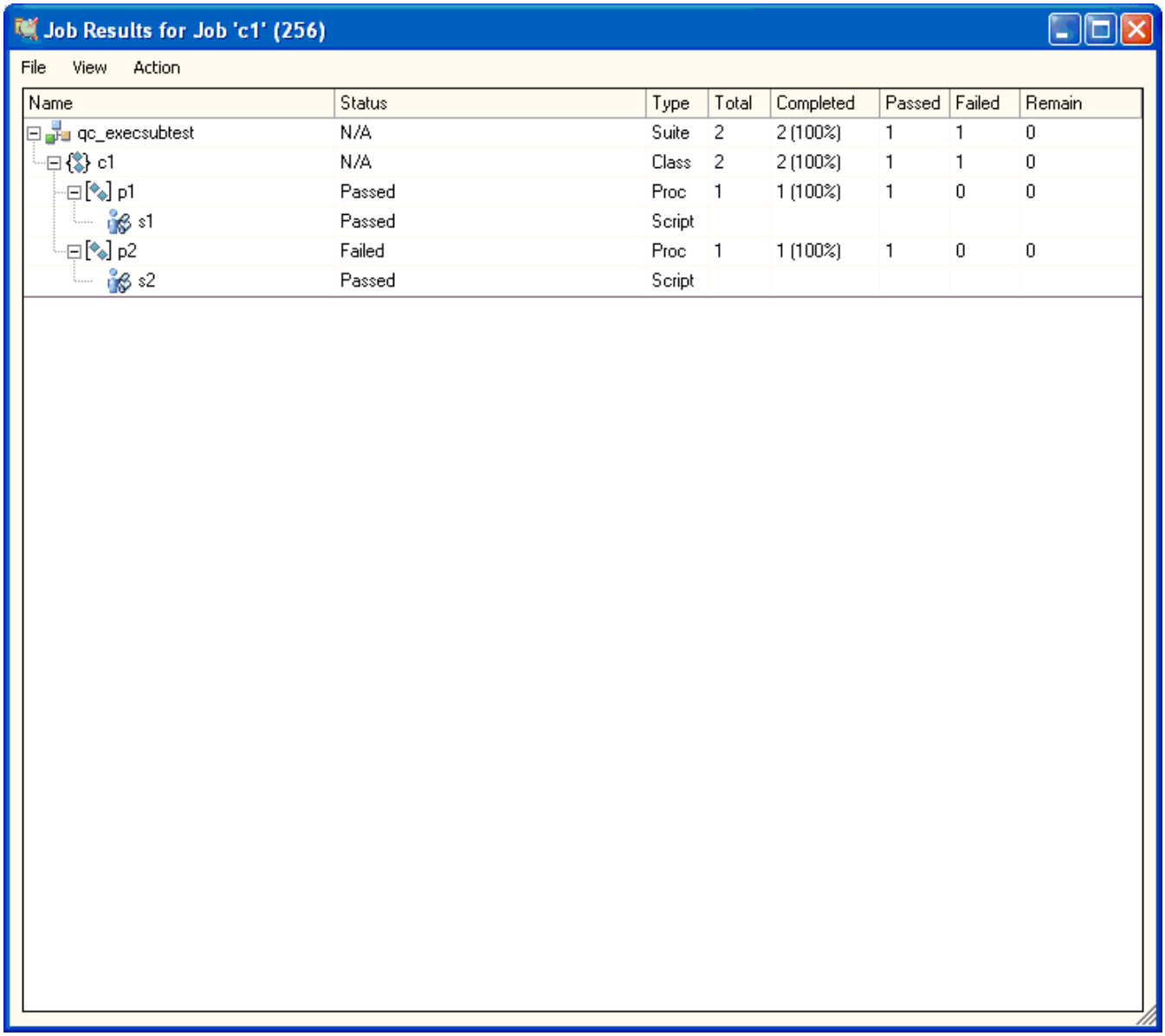
This example uses a VB6 application to test a *QA*Run script. This application prompts the user to enter a *name* and an *SSN*. The script uses qc_exec subtest commands to break the script into 2 subtests. A `qc_exec subtest_start` command marks the starting point of each subtest. One subtest validates the **Name** field;

the other validates the **SSN** field. If the test gives the wrong input (the right input displays in the application main window), *QA*Director invokes the `qc_exec subtest_fail` command to fail the subtest or that unit of the script. This eventually fails the test procedure. If the correct input is given, the `qc_exec subtest_pass` command is invoked to pass the subtest.

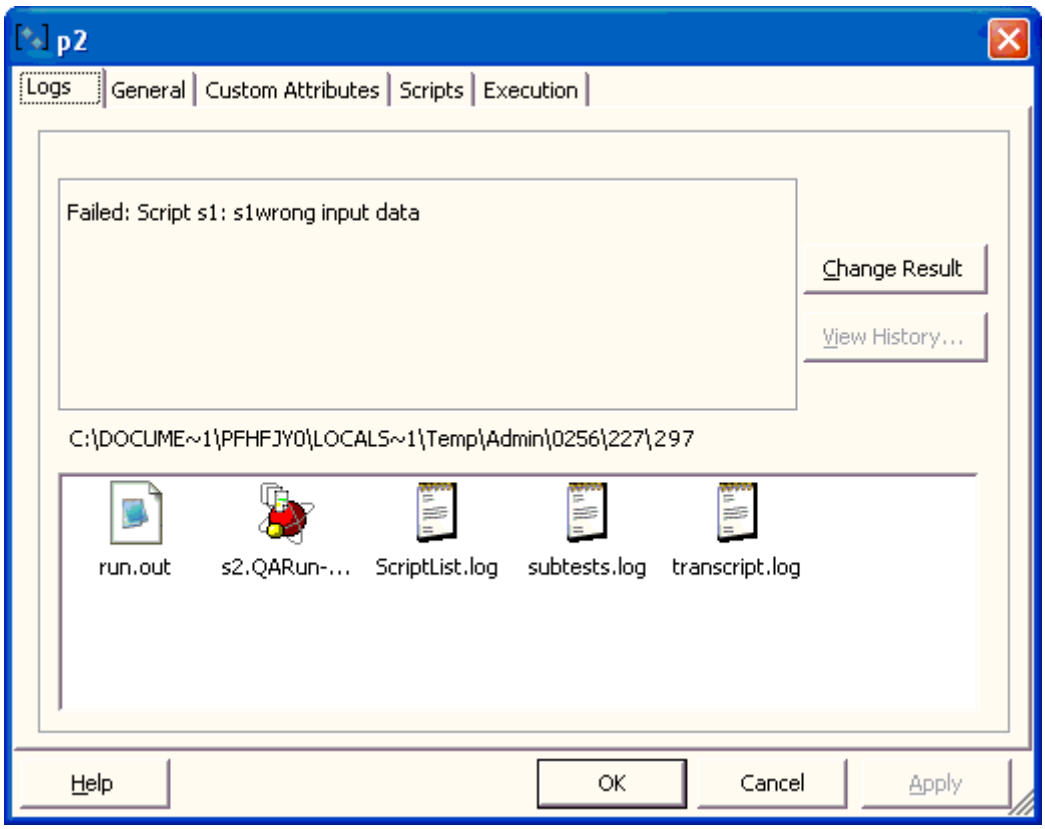
Note: The script is an automated script. Only inputs for the **Name** and **SSN** fields are expected from the user.

The following is a graphical representation of the sample application in the sequence that it is invoked.

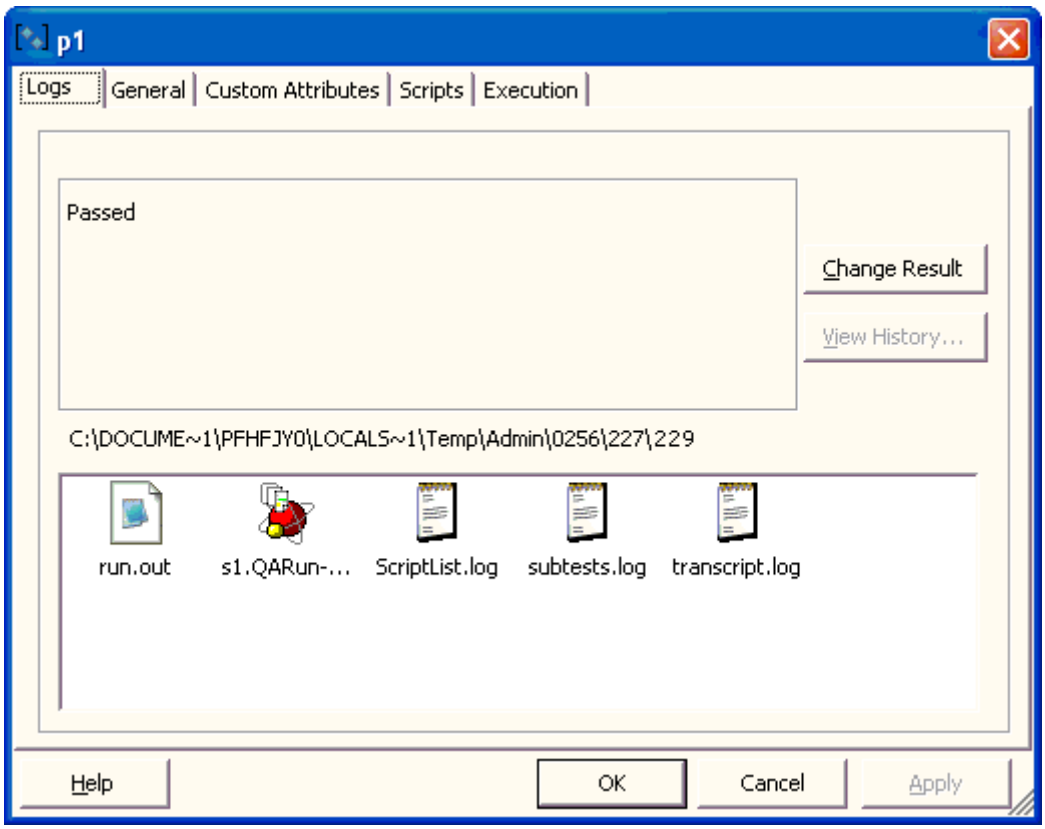
Test application main window



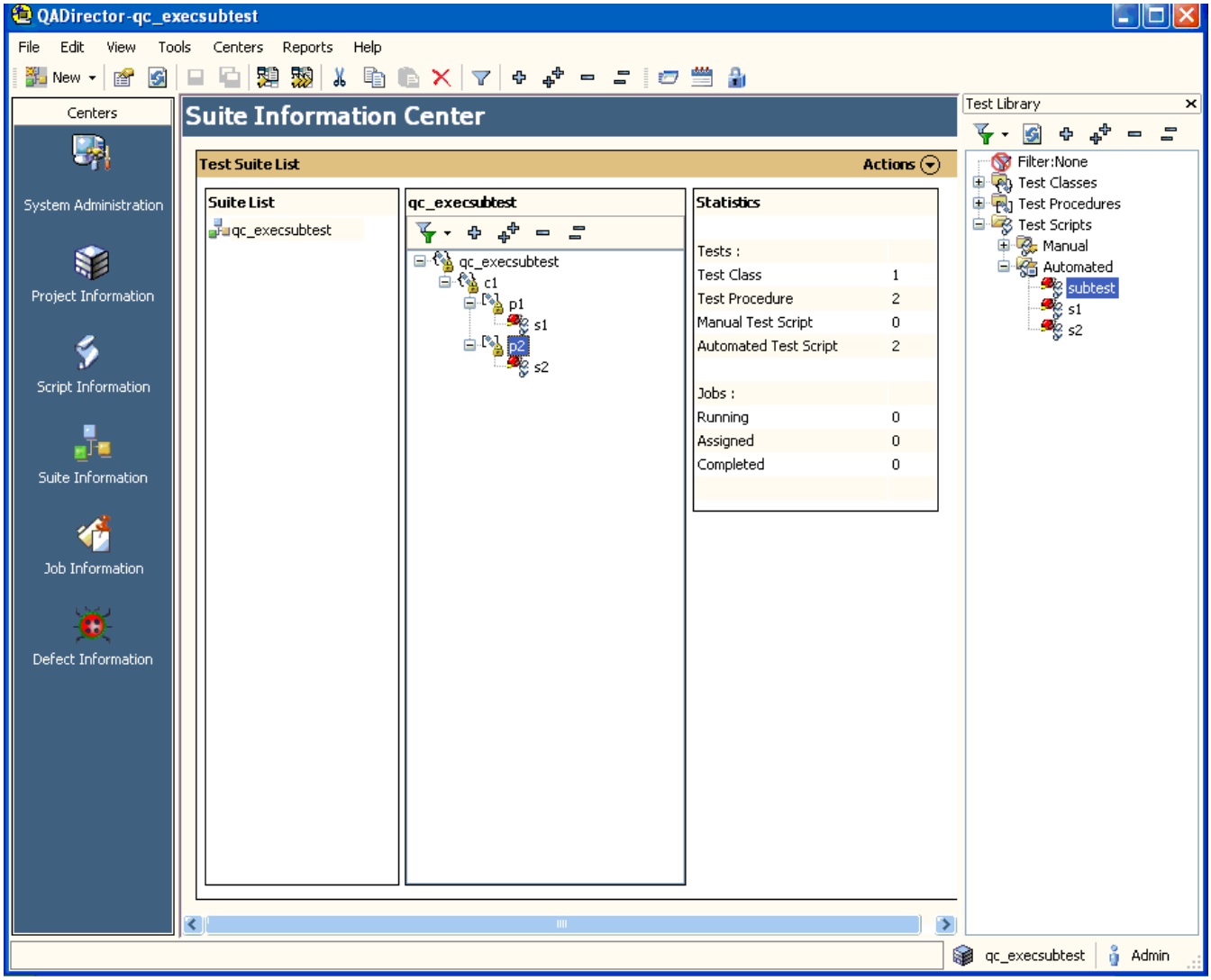
Test application main window validates **Name** input



Test application main window validates **SSN** input



Various combinations of the subtest results



The following is the *QA*Run script that was tested against the above sample VB application:

Note: The commented code highlights the sections where the global command is called.

```
Function Main
; Remove the comment below to "Enable" error handling
; On Error Call OnErrorHandler
ret = PromptBox("Test","Enter Name",value)


Attach "Test MainWindow"
exec("c:\program files\compuware\QADirector\x86-win32\bin\qc_exec.exe subtest_start s1");
GLOBAL COMMAND
EditClick "~1", 'Left SingleClick', 44, 9
EditText "~1", value
Button "Name OK", 'Left SingleClick'
ret =ActiveName()
if ret ="Name Not OK PopupWindow"
```

```
     exec("c:\program files\compuware\QADirector\x86-win32\bin\qc_exec.exe
     subtest_fail””Check log for data””")
else
     exec("c:\program files\compuware\QADirector\x86-win32\bin\qc_exec.exe
     subtest_pass")
endif
    SetFocus(ret)
    Button "OK", 'Left SingleClick'


ret = PromptBox("Test","Enter SSN",value)
Attach "Test MainWindow"
exec("c:\program files\compuware\QADirector\x86-win32\bin\qc_exec.exe subtest_start s2");
GLOBAL COMMAND
EditClick "~2", 'Left SingleClick', 49, 14
EditText "~2", value
Button "SSN OK", 'Left SingleClick'
ret =ActiveName()
if ret ="SSN Not OK PopupWindow"
     exec("c:\program files\compuware\QADirector\x86-win32\bin\qc_exec.exe
     subtest_fail ““Check log for data””")
else
     exec("c:\program files\compuware\QADirector\x86-win32\bin\qc_exec.exe
     subtest_pass")
endif
    SetFocus(ret)
    Button "OK", 'Left SingleClick'
End Function ; Main
```

Shell Commands

Shell Commands

*QA*Director shell commands are located at \path-to\Compuware*QA*Director\bin directory. The path_to is the path to your Compuware installation.

The command usage can be displayed at the DOS prompt. To display the command usage, type the command name in the shell with the –help parameter. For example, `qc_del_suite -help`.

qc_abort_jobs

Terminates one or more running jobs

Syntax

```
qc_abort_jobs [-help] [-v]
[-login <QC_QADLOGIN>] [-passwd <QC_QADPASSWD>]
[-graceful] {-ids <job ids>}
```

Parameters

Parameter	Description
help	Provides online help for command usage and syntax.
v	Specifies that the command be verbose.
login	Specifies a *QA*Director login name to use.
passwd	Specifies a password for the *QA*Director login user.
graceful	Runs cleanup rules when aborting. Default is not to run cleanup rules.
ids	Lists job IDs to abort. Use “qc_admin list_jobs –all” to get a list of job IDs.

qc_admin

Administrative tools include several commands that govern test administration in general or on individual servers.

Syntax

```
qc_admin [–help] [-v]
[-login <QC_QADLOGIN>] [-passwd <QC_QADPASSWD]
conn
kill_tmsrv
kill_tesrvs
list_jobs [-all]  [-projID <ID>]
list_suites  [-projID <ID>]
list_tesrvs [-ip]
reinit
shutdown
remove_tesrv <mach>
mach_info <mach>
start_prog <mach>  [-dir <dir>]  [-timeout <secs>| -wait] -e "cmd [arg(s)]"
kill_prog <mach> [-pid PID]
list_prog <mach> [-pid PID]
mkdir <mach> {-tmp | -dir dirname}
rmdir <mach> -dir dirname   .
send <mach> <local_file> <remote_dir>
get <mach> <remote_file> <local_dir>
```

Parameters

Parameter	Description
help	Provides online help for command usage and syntax
v	Specifies that the command be verbose
login	Sets the *QA*Director login name.
password	Sets the *QA*Director password.

Commands

Command	Description
conn	Attempts to connect to the Test Management Server to confirm that the server is running. If a server is running, the command returns OK. If the server is not running, the command returns an error message.
kill_tmsrv	Shuts down the test management server. Only users with administrator privileges, QA Manager roles, or Team Leader roles can execute this option.
kill_tesrvs	Shuts down available test execution servers. Only users with administrator privileges, QA Manager roles, or Team Leader roles can execute this option.
list_jobs [-all] [-projID <ID>]	Lists scheduled or running jobs: [-all] switch lists scheduled, running and completed jobs. [-projID] is optional and lists only jobs for the given

	project.
list_suites [-projID <ID>]	Lists all suites. [-projID] is optional and lists only suites for given project by <ID>.
list_tesrvs [-ip]	Lists all machines available for execution. '-ip' displays the IP addresses for each TE Server.
reinit	Reinitializes the test management server and the test execution servers. Only users with administrator privileges, QA Manager roles, or Team Leader roles can execute this command.
shutdown	Shuts down the available test execution servers and the test management server. Is equivalent to issuing both qc_admin kill_tesrvs and qc_admin kill_tmsrv.
remove_tesrv <mach>	Removes the test execution server on the machine specified by <mach> from the list of available test execution servers. Only users with administrator privileges, QA Manager roles, or Team Leader roles can execute this command.
mach_info <mach>	Lists information about the machine specified by <mach>.
mkdir <mach> {-tmp | -dir <dirname>}	Creates a directory on the machine specified by <mach>. If -tmp is specified, a new temporary directory is created. If –dir <dirname> is specified, a new directory specified by <dirname> is created.
rmdir <mach> -dir <dirname>	Removes the directory and its contents specified by <dirname> on a machine specified by <mach>.

send <mach> <local_file> <remote_dir>	Sends a file specified by <local_file> to a directory specified by <remote_dir> on a remote machine specified by <mach>.
get <mach> <remote_file> <local_dir>	Gets a file specified by remote_file from a remote machine specified by <mach> and places it in a local directory specified by <local_dir>.
start_prog <mach> [-dir <dir>] [-timeout <secs> | -wait] -e "cmd [arg(s)]"	Starts a process specified by the command 'cmd,' and arguments specified by 'arg(s),' using a test execution server on a machine specified by <mach>. This returns after the number of seconds specified by –timeout <secs>, or waits until the command completes if the [-wait] option is used. Please note that [-timeout] option and [-wait] option cannot be used together. If the [-wait] option is used, [-timeout] will be ignored. The default is [-wait]. The command runs with the current environment.
kill_prog <mach> [-pid PID]	Tells the test execution server on the machine specified by <mach> to terminate the process with a PID number specified by PID
list_prog <mach> [-pid PID]	Tells the test execution server on the machine specified by <mach> to return the status of the process started by start_prog and specified by PID. If you do not specify a PID, the test execution server returns all PIDs started by start_prog.

qc_ch_class

Modifies the test class according to a specification.

Syntax

```
qc_ch_class [-help] [-v]
[-login <QC_QADLOGIN>] [-passwd <QC_QADPASSWD>]
[-project_name <project name>]
{-suite_name <suite name>}
-id <class suite node id for change>
[-setup "command[:timeout][:local|classes|procs|all]"]
[-cleanup "command[:timeout][:local|classes|procs|all]"]
[-passfail "command[:timeout]"]
[-env NAME=VALUE]
[-attr NAME=VALUE]
[-summary summary]
[-parallel classes|procs|all|none]
[-bind_to <bind machine>]
[-mode online|offline]
```

Parameters

Parameter	Description
help	Provides online help for command usage and syntax.
v	Specifies that the command be verbose.
project_name <project name>	Specifies the project by <project name>.
suite_name <suite name>	Identifies the test suite specified by <suite name> as the suite in which you are creating the test class.
id <id>	Identifies the test class you want to modify by the suite node id specified by <id>.
setup “command command[:timeout][:local|classes|procs|all]"	Defines a setup command in the test class specified by command, with timeout specified by timeout, in minutes, and applies the command to local (this class), Classes (descendant classes) procedures (descendant procedures) or all (this class, descendant classes, and descendant procedures). The default is local.
cleanup “command	Defines a cleanup command in the test class

command[:timeout][:local|classes|procs|all]"	specified by command, with timeout specified by timeout, in minutes, and applies the command to local (this class), Classes (descendant classes) procedures (descendant procedures) or all (this class, descendant classes, and descendant procedures). The default is local.
passfail “command[:timeout]”	Defines a passfail command in the test class specified by command, with timeout specified by timeout, in minutes.
env name=value	Defines an environment variable rule in the test class with the name and the value.
attr name=value	Defines a test attribute for the test class with the name and value.
summary <summary>	Defines a summary for the test procedure specified by <summary>.
parallel class|procedure|all|none	Specifies which type of child can run in parallel: class (child test classes), procedure (child test procedures), all (child test classes and procedures), none (all children run serially). The default is none.
bind_to machines	Defines the machine on which the test must run as specified by the machine.
login	Sets the *QA*Director login name.
passwd	Sets the *QA*Director password.

qc_ch_proc

Modifies the test procedure according to specifications.

Syntax

```
qc_ch_proc [-help] [-v]
[-login <QC_QADLOGIN>] [-passwd <QC_QADPASSWD>]
[-project_name <project name>
{-suite_name <suite name>
{-id < test procedure suite node ID to change>}
[-type automated|manual]
[-setup "command[:timeout]"]
[-cleanup "command[:timeout]"]
```

```
[-passfail "command[:timeout]"]
[-env NAME=VALUE]
[-attr NAME=VALUE]
[-summary summary]
[-script "name^tool^descr^exit^options[^cmdline]"]
[-timeout <timeout in mins for the scripts>]
[-expected [pass|fail|unrunnable]
[-bind_to <bind_machine>]
[-abort_on_first_fail 0|1]
[-mode online|offline]
```

Parameters

Parameter	Description
help	Provides online help for command usage and syntax.
v	Specifies that the command be verbose.
login	Sets the *QA*Director login name.
passwd	Sets the QADirector password.
project_name <project name>	Identifies the project specified by <project name>.
suite_name <suite name>	Identifies the test suite specified by <suite name> as the suite in which you are creating the test procedure.
id <id>	Identifies the test procedure to modify by the suite node ID specified by <id>.
type automated|manual	Identifies the new test procedure as manual or automated. The default is automated.
setup *"command[:timeout]]"*	Defines a setup command in the test procedure specified by *command*, with timeout specified by *timeout*, in minutes.
cleanup *"command [:timeout]"*	Defines a cleanup command in the test procedure specified by

	command, with timeout specified by *timeout*, in minutes.
passfail "*command[:timeout]*"	Defines a pass/fail command in the test procedure specified by *command*, with timeout specified by *timeout*, in minutes.
env name=value	Defines an environment variable rule in the test procedure with the name and the value.
attr name=value	Defines a test attribute for the test procedure with the name and value.
summary <summary>	Defines a summary for the test procedure specified by <summary>
script "script name^tool name^descrption^exit status^options[^command line]"	Defines script information, which is a set of the following information, separated by a caret(^): script name tool name description exit status options command line.
timeout <timeout>	Defines the timeout in minutes for the script of the test procedure specified by <timeout>. The default value is 60.
expected pass|fail|unrunnable	Defines the expected outcome of the test procedure specified by pass, fail, or unrunnable. The default value is pass.
bind_to <machine>	Defines the machine on which the test must run as specified by machine.
abort_on_first_fail 0|1	Specifies if the job will stop if one of the scripts has failed. "0" is set that job will continue even if one test script is failed. "1" is set that job will stop once a test script is failed. The default is "0"

mode online|offline	Defines whether the test procedure is online or offline.

Example

Insert a User Defined script into Proc1 with Test ID 101:

```
qc_ch_proc -login admin -passwd admin –project_name project1 -suite_name suite1 -id 101
-script "Notepad^User Defined^Test^n/a^c:\temp\new.txt"
```

qc_ch_result

Changes the job result.

Syntax

```
qc_ch_result [-help] [-v]
[-login <QC_QADLOGIN>] [-passwd <QC_QADPASSWD>]
{-project_name <project name>}
{-job_id <job_id>} {-id <id>} [-c <comment>]
{<result> [failure_desc]}
```

Parameters

Parameter	Description
help	Provides online help for command usage and syntax.
v	Specifies that the command be verbose.
login <QC_QADLOGIN>	Specifies the *QA*Director login name.
passwd <QC_QADPASSWD>	Specifies the *QA*Director password.
project_name <project name>	Specifies project name for the job.
job_id <job id>	Specifies the ID of the job that is to be changed.
id <suite node id>	Specifies the suite node ID of the test procedure that is to be changed.
c <comment>	Displays a comment for this change. This is optional.
<result>	Defines the changed result. It must be a result of of 'pass', 'fail',or 'unrunnable' .

<failure_desc>	Defines the failure, if applicable. Ignored if <result> is 'pass.'

qc_del_class

Removes a test class from the test suite.

Syntax

```
qc_del_class [-help] [-v]
[-login <QC_QADLOGIN>] [-passwd <QC_QADPASSWD>]
[-project_name <project name>]
{-suite_name <suite name>}
-id <test class id to delete>
```

Parameters

Parameter	Description
help	Provides online help for command usage and syntax
v	Specifies that the command be verbose
login	Sets the *QA*Director Login Name
passwd	Sets the *QA*Director Password
project_name <project name>	Specifies the project by <project name>. You must specify the project name option, if it is not same as suite name.
suite_name <suite name>	Identifies the test suite specified by <suite name> as the suite in which you are creating the test class.
id <id>	Identifies the test class you want to delete by the suite node id specified by <id>.

qc_del_proc

Removes a test procedure from the test suite.

Syntax

```
qc_del_proc [-help] [-v]
 -login <QC_QADLOGIN>] [-passwd <QC_QADPASSWD>]
[-project_name <project name>]
{-suite_name <suite name>}
{-id <id>}
```

Parameters

Parameter	Description
help	Provides online help for command usage and syntax.
v	Specifies that the command be verbose.
login	Sets the *QA*Director login name.
passwd	Sets the *QA*Director password.
project_name <project name>	Specifies the project by <project name>. Specifies a project name option, if it is not the same as suite name.
suite_name <suite name>	Identifies the test suite specified by <suite name> as the suite in which you are creating the test class.
id <id>	Identifies the test procedure to delete by the suite node id specified by <id>.

qc_del_results

Deletes one or more specified job results.

Syntax

```
qc_del_results [-help] [-v]
[-login <QC_QADLOGIN>] [-passwd <QC_QADPASSWD>]
{-project_name <project name>
{-ids <id-list>}\n\n")   ;
```

Parameters

Parameter	Description
help	Provides online help for command usage and syntax.
v	Specifies that the command be verbose.
login <QC_QADLOGIN>	Specifies *QA*Director login Name.
passwd <QC_QADPASSWD>	Specifies *QA*Director password.
project_name	Specifies the project name for the job.

ids <id-list>	Deletes the job results specified by id-list. You can use “qc_admin list_jobs –all” to get a list of job IDs.

qc_del_suite

Removes a test suite from the list known to QADirector

Syntax

```
qc_del_suite [-help] [-v]
[-login <QC_QADLOGIN>] [-passwd <QC_QADPASSWD>]
[-projectname <project name>]
{-suite_name <suite name>
```

Parameters

Parameter	Description
help	Provides online help for command usage and syntax.
v	Specifies that the command be verbose.
projectname <project name>	Specifies the project by <project name>.
suite_name <suite name>	Identifies the test suite specified by <suite name> as the suite in which you are creating the test procedure.

Example

This example removes a test suite name suite1 from the list known to *QA*Director:

```
qc_del_suite -login admin –passwd admin –projectname project1 -suite_name suite1
```

qc_get_attr

Returns the name and values of the attributes or environment variables you specify.

Syntax

```
qc_get_attr: [-help]
[-login <QC_QADLOGIN>] [-passwd <QC_QADPASSWD>]
[-projectname <project name>]
{-suite_name <suite> | -job_id <job id> }
{-id <suite node id>}
[<attr>|-list_test|-list_result|-list_envvs| -list_builtins|-list_all]
```

Parameters

Parameter	Description
login	Sets the QADirector login name.
passwd	Sets the QADirector password.
help	Provides online help for command usage and syntax.
projectname <project name>	Specifies the name of the project by <project_name>. If -job_id option is used, -projectname option must be used.
suite_name <suite_name>	Specifies that the command return the current values of the attributes in the test suit specified <suite_name>. Note that –suite_name and –job_id are mutually exclusive.
id <suite node id>	Specifies the test by <suite node id> in the suite or job you specified with –suite_name or –job_id.
job_id <job id>	Specifies that the command return the current values of the job as specified by <job id>. Note that –suite_name and –job_id are mutually exclusive.
<attr>	Specifies the attribute whose value you want returned.
list_test	Returns the test attributes of the test specified by –id.
list_envvs	Returns the environment variables of the test specified by –id.
list_builtins	Returns the built-in attributes of the test specified by –id.
list_all	Returns all attributes of the test specified by –id option.

qc_new_class

Creates a test class defined by the switches you specify

Syntax

```
qc_new_class [-help] [-v]
[-login <QC_QADLOGIN>] [-passwd <QC_QADPASSWD>]
[-project_name <project name>]
{-suite_name <suite name>}
-p_id <parent id for new class>
[-setup "command[:timeout][:local|classes|procs|all]"]
[-cleanup "command[:timeout][:local|classes|procs|all]"]
```

```
[-passfail "command[:timeout]"]
[-env NAME=VALUE]
[-attr NAME=VALUE]
[-summary <summary>]
[-parallel classes|procs|all|none]
[-bind_to <bind machine>]
[-mode online|offline]
<test class name>
```

Parameters

Parameter	Description
help	Provides online help for command usage and syntax.
v	Specifies that the command be verbose.
project_name <project name>	Specifies the project by <project name>.
suite_name <suite name>	Identifies the test suite specified by <suite name> as the suite in which you are creating the test class.
p_id <id>	Identifies the selected test class as the parent by its suite node id.
setup “command [:timeout][:local|classes|procs|all]"	Defines a setup command in the test class specified by command, with timeout specified by timeout, in minutes, and applies the command to local (this class), Classes (descendant classes), procedures (descendant procedures) or all (this class, descendant classes, and descendant procedures). The default is local.
cleanup “command [:timeout][:local|classes|procs|all]"	Defines a cleanup command in the test class specified by command, with timeout specified by timeout, in minutes, and applies the command to local (this class), Classes (descendant classes) procedures (descendant procedures) or all (this class, descendant classes, and descendant procedures). The default is local.
passfail “command[:timeout]”	Defines a passfail command in the test class specified by command, with timeout specified by timeout, in minutes.
env NAME=VALUE	Defines an environment variable rule in the test class with the NAME and the VALUE.

attr NAME=VALUE	Defines a test attribute for the test class with the NAME and VALUE.
summary <summary>	Defines a summary for the test procedure specified by <summary>.
parallel class|procedure|all|none	Specifies which type of child can run in parallel: class(child test classes), procedure (child test procedures), all (child test classes and procedures), none (all children run serially). The default is none.
bind_to machines	Defines the machine on which the test must run as specified by machine.
<test class name>	Specifies the name of the new test class.
login	Sets the *QA*Director login name.
passwd	Sets the *QA*Director password.

qc_new_proc

Creates a new test procedure defined by the switches you specify.

Syntax

```
qc_new_proc [-help] [-v]
[-login <QC_QADLOGIN>] [-passwd <QC_QADPASSWD>]
[-project_name <project name>
{-suite_name <suite name>
{-p_id <parent Id for new procedure>}
[-type automated|manual]
[-setup "command[:timeout]"]
[-cleanup "command[:timeout]"]
[-passfail "command[:timeout]"]
[-env NAME=VALUE]
[-attr NAME=VALUE]
[-summary summary]
[-script "name^tool^descr^exit^options[^cmdline]"]
[-timeout <timeout in mins for the scripts>]
[-expected [pass|fail|unrunnable]
[-bind_to <bind_machine>]
[-abort_on_first_fail 0|1]
[-mode online|offline]
<test procedure name>
```

Parameters

Parameter	Description
help	Provides online help for command usage and syntax.
v	Specifies that the command be verbose.
project_name <project name>	Specifies the project by <project name>.
suite_name <suite name>	Identifies the test suite specified by <suite name> as the suite in which you are creating the test procedure.
p_id <id>	Identifies the selected test procedure as the parent by its suite node id.
type automated|manual	Identifies the new test procedure as manual or automated. The default is automated.
setup “command[:timeout]]"	Defines a setup command in the test procedure specified by command, with timeout specified by timeout, in minutes.
cleanup “command [:timeout]"	Defines a cleanup command in the test procedure specified by command, with timeout specified by timeout, in minutes.
passfail “command[:timeout]”	Defines a passfail command in the test procedure specified by command, with timeout specified by timeout, in minutes.
env name=value	Defines an environment variable rule in the test procedure with the name and the value.
attr name=value	Defines a test attribute for the test procedure with the name and value.
summary <summary>	Defines a summary for the test

	procedure specified by <summary>.
script "script name^tool name^descrption^exit status^options[^command line]"	Defines script information which is a set of the following information, each separated by a caret(^):script name, tool name, description, exit status, options, command line.
timeout <timeout>	Defines the timeout in minutes for the script of the test procedure specified by <timeout>. The default value is 60.
expected pass|fail|unrunnable	Defines the expected outcome of the test procedure specified by pass, fail, or unrunnable. The default value is pass.
bind_to <machine>	Defines the machine on which the test must run as specified by machine.
abort_on_first_fail 0|1	Specifies if the job will stop to run while one of script is failed. “0” is set so the job will continue even if one test script is failed. “1” is set so the job will stop once a test script is failed. The default is “0”.
mode online|offline	Defines the test procedure is online or offline.
<test procedure name>	Specifies the name of the new test procedure.

Example

Create a new procedure Proc1 with a User Defined script:

```
qc_new_proc -login admin -passwd admin -suite_name suite1 -p_id 100   -script "Notepad^User Defined^Test^n/a^c:\temp\new.txt” Proc1
```

qc_new_suite

Creates a *QA*Director test suite

Syntax

```
qc_new_suite [-help] [-v]
[-login<QC_QADLOGIN>] [-passwd <QC_QADPASSWD>]
[-projectname <project name>]
{-suite_name <suite name>}
[-descr <suite description>]
```

Parameters

Parameter	Description
help	Provides online help for command usage and syntax.
v	Specifies that the command be verbose.
projectname <project name>	Specifies the project by <project name>.
suite_name	Specifies the name of the new test suite.
descr <suite description>	Displays a description for the new suite specified by <suite description>.
login	Sets the *QA*Director Login Name.
passwd	Sets the *QA*Director Password.

qc_run_job

Schedules a job to run.

Syntax

```
qc_run_job [-help] [-v]
[-login <QC_QADLOGIN>] [-passwd <QC_QADPASSWD>]
[-project_name <project name>]
-suite_name <suite name> -name <job name>
-platform <platform> -ids <id list> -machines <m/c list>)
[option]
-time
-freq
-env
-type automated|manual|all
-s
-use_default_env 0|1
-run_on_web
-assign_to <USER>
```

Parameters

Parameter	Description

project_name	Specifies the name of the project. Is only omitted when the project name is same as suite name.
suite_name	Specifies the name of the suite.
name	Specifies the job name.
platform	Specifies the platform, for example, on windows, x86/winNT.
machines	Displays a list of machines on which to run the job.
ids <suite node ids>	Sets the tests specified by <suite node ids> you want to run. Set the tests by a space separated list of <suite node ids> to run. If '-' is used the suite node ids will be read from stdin.
login	Sets the *QA*Director Login Name.
passwd	Sets the *QA*Director Password.
time hh:mm [am|pm]	Sets the time you want the job to run by using the format hh:mm [am|pm] .
freq	Specifies the frequency: daily, weekly, monthly or once. The default is once.
env	Specifies the environment variable rules.
type automated|manual|all	Sets the type of tests to schedule specified by automated, manual, or all. The default is automated.
s	Stops on an error (such as currently unavailable machines). Default: unavailable machines for scheduled jobs in future, are ignored.
use_default_env 0|1	Runs the job including the user's default environment. The default is '1.'
run_on_web	Runs the job 'on the web.'
assign_to <USER>	Assigns the job to the USER. Only valid with -run_on_web

Example:

```
qc_run_job –login admin –passwd admin –project_name project1 –suite_name suite1 –name
command_line –platform x86\winNT –machines London –ids 49
```

qc_tesrv

Starts a **Test Execution Server**.

Syntax

```
qc_tesrv [switches...]
```

Operation

A **Test Execution Server** provides information about machine availability and starts processes when requested by the test management server. To run tests with *QA*Director, start at least one **Test Execution Server**. Start a **Test Execution Server** on each machine on which you want tests to run.

Parameters

Parameter	Description
c <config_file>	Identifies the tesrv.cfg file the **Test Execution Server** should read, specified by *config_file*. The **Test Execution Server** reads the specified file. The server does not read either the global nor machine tesrv.cfg file.
help	Provides online help for command usage and syntax.
v	Specifies that the command be verbose.
domainname	Specifies a domain name. Use a domain-qualified, not local, hostname. By default, servers identify themselves with local hostnames. For example, if the local hostname is *nobska*, the servers use *nobska*. Supplying the **-domainname** switch causes the servers to use a domain-qualified name. For example, if the fully qualified hostname is *nobska.eng.company.com,* the servers use *nobska.eng*, the domain-qualified name.
name <name>	Identifies the name the servers should use to refer to the local machine, specified by *name*. In most cases, **-name** is unnecessary, because the **-domainname** switch supports *QA*Director processes communicating across subnets. If **-domainname** does not work correctly (it relies on the NIS domainname), use the **-name** switch to set the hostname of the local machine. For example, if the fully qualified hostname is *nobska.group.eng.company.com*, use `-name nobska.group` to function within the *eng* subnet.
tmpdir <directory>	Identifies the directory the test execution server should use as a temporary directory as specified by *directory*. The **-connect** switch is applicable only in remote execution.

qc_testselect

Prints to standard output the test_ids of the tests that match a query using the values of test attributes, result attributes, or both

Syntax

```
qc_testselect [-help] [-v]
[-login <QC_QADLOGIN>] [-passwd <QC_QADPASSWD>]
[-project_name <project name>]
{-suite_name <suite_name> | -job_id <job_id>}
[-ids test_ids | -] [-select_tps_only] [-collapse]
[-print_names]
```

```
{-query <query>}
```

Parameters

Parameter	Description
help	Provides online help for command usage and syntax.
v	Specifies that the command be verbose.
suite_name <suite name>	Specifies that the attributes used as search criteria are test attributes in the suite specified by suite name. You must use either –suite_name or –job_id
job_id <job id>	Specifies that the attributes used as search criteria are test attributes, result attributes, or both, in a job specified by job id. You must user either –suite_name or –job_id.
project_name <project name>	Specifies a project by the project name in which resides the suite or job. If -job_id option is used, -project_name option must be used. If suite name is same as project name, -project_name option can be omitted.
login	Specifies a *QA*Director login name to use.
passwd	Specifies a password for the *QA*Director login user.
ids	Applies a query to the tests in the list and their descendants. **Format:** Space separated list of test ids to search. If '-' is used, read from stdin. **Default:** Apply the query to the entire suite.
collapse	Prints only the parent test class if all the descendants satisfy the query.
select_tps_only	Prints only the test procedures that satisfy the query.
print_names	Prints the names of matching tests. **Default:** Print their ids.
query <query>	Specifies the query to run.

Query Operators

==	Equals
!=	Is not equal to
	Is greater than

<	Is less than
>=	Is equal to or greater than
<=	Is equal to or less than
contains	Contains
not_contains	Does not contain

Examples

1. Print all the tests has keywords "A" with a state of “Unwritten:”

```
-query "$QC_STATE$ == \"Unwritten\" && $QC_KEYWORDS$ == \"A\""
```

2. Print all the tests has keywords "test" with a state of “In Progress”:

```
-query "$QC_STATE$ == \"In Progress\" && $QC_KEYWORDS$ == \"test\""
```

qc_tmsrv

Starts the **Test Management Server**.

Syntax

```
qc_tmsrv [switches...]
```

Operation

The **Test Management Server** assigns unique IDs to test suites and jobs, schedules jobs, and requests that **Test Execution Servers** start processes. To use *QA*Director after you install it, start the **Test Management Server**.

Parameters

Parameter	Description
install	Installs the service
remove	Removes the service
start	Starts the service
stop	Stops the service
console	Runs as a console application
nojoblimit	Removes limits from number of concurrent jobs running on win32
debug <params>	Runs as a console application for debugging

Scenarios

Alternative Command Execution

The global commands can be invoked in *QA*Run scripts. *QA*Director must execute those scripts. The following *QA*Run script demonstrates how to accomplish this:

Note: The commented code indicates where a *QA*Director global command is called.

```
Function Main
    ; Remove the comment below to "Enable" error handling
    ; On Error Call OnErrorHandler
ret = PromptBox("Test","Enter Name",value)
Attach "Test MainWindow"
exec("c:\program files\compuware\qadirector\x86-win32\bin\qc_exec.exe subtest_start s1") ;
GLOBAL COMMAND
    EditClick "~1", 'Left SingleClick', 44, 9
    EditText "~1", value
    Button "Name OK", 'Left SingleClick'
ret =ActiveName()
if ret ="Name Not OK PopupWindow"
    exec("c:\program files\compuware\qadirector\x86-win32\bin\qc_exec.exe subtest_fail s1
””Check log for data””")
else
    exec("c:\program files\compuware\qadirector\x86-win32\bin\qc_exec.exe subtest_pass s1")
endif
    SetFocus(ret)
    Button "OK", 'Left SingleClick'
ret = PromptBox("Test","Enter SSN",value)
Attach "Test MainWindow"
     exec("c:\program files\compuware\qadirector\x86-win32\bin\qc_exec.exe subtest_start
s2") ; GLOBAL COMMAND
    EditClick "~2", 'Left SingleClick', 49, 14
    EditText "~2", value
    Button "SSN OK", 'Left SingleClick'
ret =ActiveName()
if ret ="SSN Not OK PopupWindow"
exec("c:\program files\compuware\qadirector\x86-win32\bin\qc_exec.exe subtest_fail s2
“”Check log for data””")
else
    exec("c:\program files\compuware\qadirector\x86-win32\bin\qc_exec.exe subtest_pass s2")
endif
    SetFocus(ret)
    Button "OK", 'Left SingleClick'
End Function ; Main
```

The commented lines show how you can invoke the global commands from a QARun script. You must specify the fully qualified path to the qc_exec.exe program. In this case, the path is `c:\program files\compuware\qadirector\x86-win32\bin\`.

The QARun `exec` function executes the qc_exec.exe program.

Command Log

*QA*Director creates a file named events.log in the results sub-directory for tests that use lock or event global commands. This file contains an entry for every lock and event command as they are executed. Samples of this log follows.

Note: when running tests in parallel across multiple machines, the events.log file appears in the results sub-directory of one of the tests.

```
          Test
Time          ID    Operation        Result
--------    ----   ---------        ------
08:36:23     103  lock_get walter    Fail
```

Reason: Lock 'walter' unavailable (held by test ID 101).

```
          Test
Time          ID     Operation                        Result
--------    ----    ---------                        ------
16:05:31    101    lock_get walter                     Ok
16:05:35    101    lock_release walter (test completed) Ok
```

```
          Test
Time        ID    Operation            Result
--------   ----   ---------            ------
10:38:22   104   event_occurred walter   Ok
```

Finding Elapsed Time Between Events

Determine how well a test performs by using commands that measure timings. With these commands, the following can be accomplished:

- Find the elapsed time between the start of an operation (qc_exec timer_start) and its conclusion (qc_exec timer_end).
- Find the longest and shortest elapsed times of an operation that a job performs multiple times (qc_exec timer_getval).
- Notify *QA*Director to fail a test if its operation took more than a specified number of seconds (qc_exec timer_assert).

The scope of a timer is a test. One test cannot access the timer of another test.

Finding the State of Events

Find information about the state of events that occur during a job by defining the start (qc_exec event_start) and end (qc_exec event_end) of events. You can find out whether the event is:

- In progress (qc_exec event_in_progress)

Finished (qc_exec event_done)

Running or finished (qc_exec event_occurred)

Imposing a Dependency

It is possible to prevent a test from executing if a preceding test procedure does not pass. The test whose execution is to be prevented is named **export_db_data**, which exports data to a database. The procedure that must pass is **create_db**, which creates the database into which you want to export data.

To make test procedure **export_db_data** dependent on **create_db**, create the following setup rule in **export_db_data**:

```
qc_exec test_has_outcome -id 7 pass
```

Note that the test ID number for **create_db** is 7. *QA*Director makes the status of **export_db_data** `Not Executed` if **create_db** did not pass.

Imposing Dependencies Between Tests

While a test is running, it can find information about the outcome of a test procedure in the same job and use that information during its execution. A test can:

Find out whether a specified test procedure had an unexpected outcome or if the test procedure passed, failed, or was `Not Executed` (qc_exec test_has_outcome). If the test procedure has not yet finished, the qc_exec test_has_outcome command waits for the test to finish before reporting the outcome. You can limit the wait time, for example, by using qc_exec test_has_outcome in a setup rule. That rule can have a reasonable timeout value.

Wait until a specified test procedure finishes before proceeding (qc_exec test_wait_for). This command ignores the actual outcome of the test procedure.

Locking Resources

Use the lock commands to ensure that only one test at a time accesses a common resource. The tests must be part of the same job. Use these commands to:

Acquire a lock on a resource (qc_exec lock_get) and release it (qc_exec lock_release).

Wait for a resource to become available (qc_exec lock_get_wait).

Fail a test if it caused a deadlock on a common resource (qc_exec lock_get_wait).

Debug lock commands by obtaining information about the locks used during test execution: the name or ID of the test currently holding a lock (qc_exec lock_owner) or a list of tests currently waiting on a lock (qc_exec lock_waitlist).

To work properly, all the tests must use the same lock name for a common resource. For that reason, the test designer should assign lock names and ensure tests acquire the appropriate lock before accessing a common resource.

Locks are valid only within a job. A lock persists at most as long as the test that requested it is running, so tests in different jobs can hold the same lock without contention. If the test exits without releasing the lock, *QA*Director automatically releases the lock.

Using the Lock Capability

To make use of the locking capability, the tests of a job must execute at more or less the same time. Users must create a job where tests run on multiple machines in parallel. One way this can be accomplished is by binding tests to specific machines and having the tests run in parallel.

For example, suppose the tests of a job access the same file and that only one test at a time can access the file. The resource representing that file is named file_lock1. Each test must:

1. Acquire file_lock1 using the qc_exec lock_get command.

2. Access the file.

3. Release file_lock1 using the qc_exec lock_release so that other tests can acquire the lock and access the file.

Results of Timing

*QA*Director creates a file named timers.log in the results sub-directory of the test. The file has the following format:

```
Timer name          Count Avg [ms] Min [ms] Max [ms] Total
----------------------- ----- -------- -------- -------- --
sample_timer            1  7.886     7.886    7.886
```

In this example, one instance of the timer, *sample_timer*, was invoked. The elapsed time was 7.886 milliseconds.

Synchronizing Processes

Synchronize multiple processes in a job by pausing each one until they all reach specified points before allowing them to proceed. Synchronizing processes uses the qc_exec event_rendezvous command.

To synchronize processes:

1. Choose the processes to synchronize. The processes can be rules, scripts, or commands called by rules or scripts.

2. Pinpoint where in each process you want the process to synchronize with the others you have chosen.

3. At each point, type the following command: `qc_exec event_rendezvous rendezvous_name count`

The `rendezvous_name` is a name you choose for the rendezvous. The `count` is the number of processes to rendezvous.

Timing an Operation

Suppose you want to time three operations, each of which is represented by a test procedure: apply1, apply2, and apply3. The procedures are in a root class named global.

To time the execution of the procedures:

1. In each test procedure, define a local setup rule to start a timer. For example: `qc_exec timer_start sample_timer`. This command starts a timer named *sample_timer*.

2. In each test procedure, define a local cleanup rule to end the timer. For example: `qc_exec timer_end sample_timer`.

3. Run a job that includes the three test procedures.

Note: Instead of defining the same setup and cleanup rules three times, once in each test, define them once in the parent class. The descendant procedures inherit the rules from the parent procedure.

Integrations

About Integrating QACenter Products

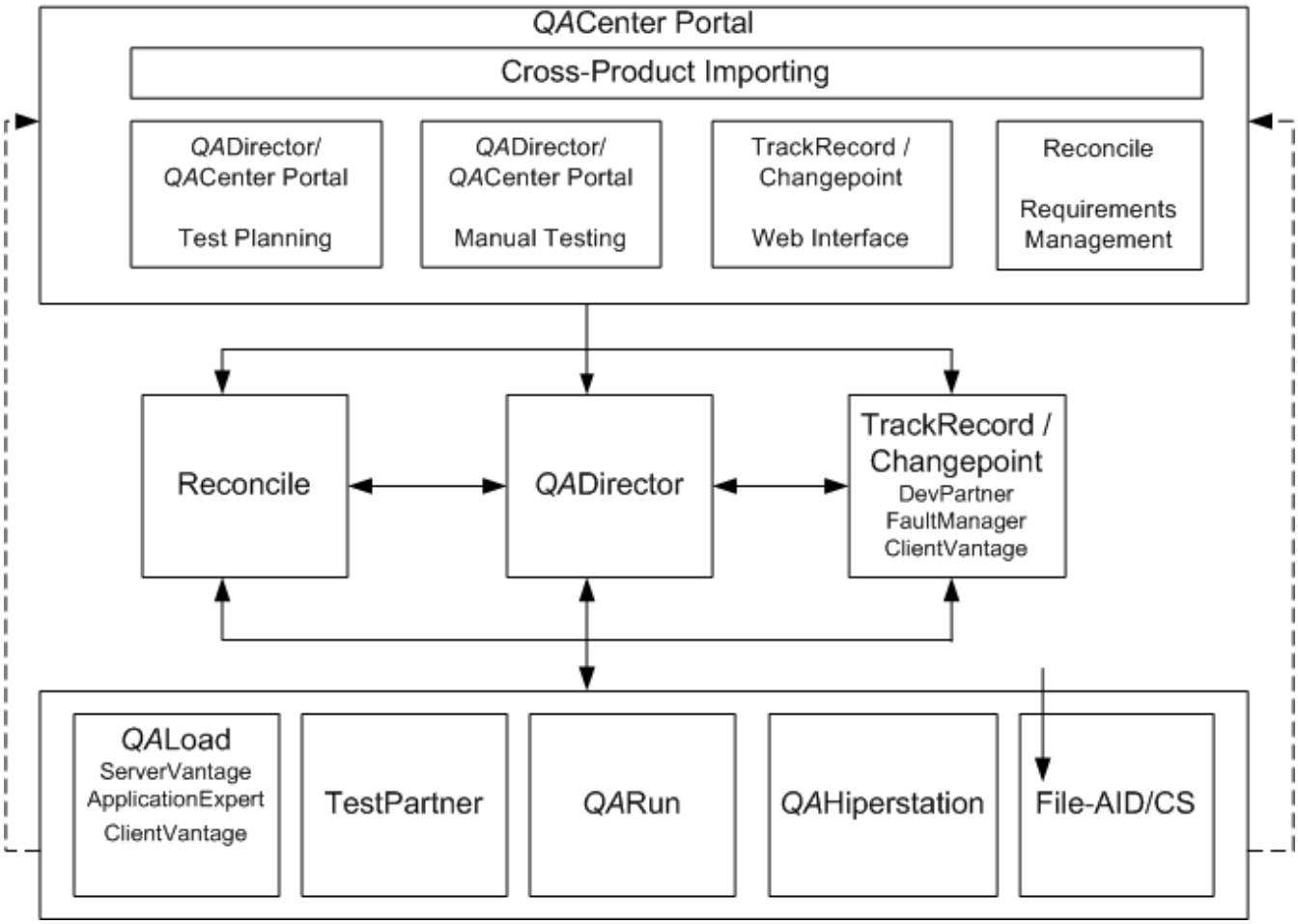
Integrating *QA*Center Products

*QA*Center tools help companies achieve consistent, dependable application performance.

*QA*Center delivers automated testing products and solutions designed to validate applications running in the full spectrum of environments, to isolate and correct problems, and to ensure that systems can handle anticipated load before applications go live. These include:

- Requirements Management
- Test Planning
- Test Execution Management
- Functional Test Automation
- Load Testing
- Mainframe Test Automation
- Defect/Request Management
- Test Data Management

As shown in the following figure, *QA*Center products work together to successfully meet all enterprise needs:



File-Aid CS

File-AID/CS is a comprehensive data management tool designed to help developers, testers and quality assurance teams work efficiently with data as they develop, test and support their applications. With File-AID/ CS, it is possible to extract, to load, to copy, to convert, to transform, to compare, and to edit data without being an expert in each database environment. File-AID/CS supports the major distributed databases (Oracle, DB2 UDB, Microsoft SQL Server and Sybase), in addition to a wide range of data types including mainframe data.

File-Aid/CS integrates with the following *QA*Center applications:

*QA*Director: Launch File-AID/CS from within *QA*Director, and avoid searching through the Windows Explorer Start menu. Also from *QA*Director, access the File-AID/CS Convert and Compare facilities. Use the File-AID/CS Convert facility to create specifications that contain all the data necessary to convert an existing data file to a new target data file. The File-AID/CS **Compare** facility creates specifications that compare the expected and actual contents of output files. From *QA*Director, create File-AID/CS tool domains for processing File-AID/CS test scripts within a test suite.

*QA*Run: *QA*Run works with File-AID/CS to execute compare specifications. Within *QA*Run, create Checks containing File-AID/CS compare specifications.

*QA*Director

Testing is a major undertaking that requires the coordination of many tasks, many testers, large suites of test scripts, and multiple versions of different applications. *QA*Director provides a framework for managing the entire testing process—from design, to execution, to analysis.

*QA*Director works with testing, requirement, and defect tracking tools, organizing tests and results in one location. Specifically, *QA*Director integrates with the following *QA*Center applications:

QACenter Portal: Data from *QA*Director reports is published in QACenter Portal. View these reports from any workstation with Microsoft Internet Explorer 5.5 or 6.0. *QA*Director does not have to be installed on the computer. Also, perform manual Web tests through the QACenter Portal.

File-AID/CS: Launch File-AID/CS from within *QA*Director and avoid searching through the Windows Explorer Start menu. Also from a *QA*Director menu, access the File-AID/CS Convert and Compare facilities. Use the File-AID/CS Convert facility to create specifications that contain data necessary to convert an existing data file to a new target data file. Use the File-AID/CS Compare facility to create specifications that compare the expected and actual contents of output files. From *QA*Director, create File-AID/CS tool domains for processing File-Aid/CS test scripts within a test suite. In addition, analyze test data by comparing test logs that are accessible through *QA*Director.

*QA*Hiperstation: From within *QA*Director, browse *QA*Hiperstation scripts on the mainframe, add scripts to test procedures, launch *QA*Hiperstation to play back the scripts, and view details about failed *QA*Hiperstation scripts.

*QA*Hiperstation+: With *QA*Hiperstation+ installed on a computer, edit and create *QA*Hiperstation scripts from *QA*Director.

*QA*Load: Launch *QA*Load Conductor, Analyze, and Player modules from a convenient *QA*Director menu. Add, find, execute, and abort scripts that can stress a distributed system by simulating thousands of users simultaneously performing different operations. Also, view results of *QA*Load tests from within *QA*Director.

*QA*Run: Launch *QA*Run from within *QA*Director to add, to create, to edit, to find, to execute, and to abort scripts that test native Windows applications, distributed applications with a Windows client, or host-based (mainframe) applications accessed via a Windows terminal emulation program. Additionally, view results and logs to analyze *QA*Run test data.

Reconcile: Use *QA*Director with Reconcile to generate test plans in accordance with changing requirements. Integration between *QA*Director and Reconcile accelerates test planning and ensures end-user needs are met. Plan tests between Reconcile and *QA*Director through QACenter Portal.

TestPartner: Launch TestPartner from within *QA*Director to add, to create, to edit, to find, to execute, and to abort scripts that test Java, Visual Basic, and Visual C++ applications, as well as browser-based Web applications. View results and logs from within *QA*Director.

TrackRecord: Launch TrackRecord, submit defects, and view defects from a center within *QA*Director and avoid searching through the Windows Explorer Start menu. Load failed test results into TrackRecord defect reports. Store the defect ID with the test procedure for easy follow-up.

Changepoint: *QA*Director integrates with Changepoint Request Management to submit, edit and delete defects.

*QA*Center Mainframe Solutions

*QA*Center mainframe solutions are a suite of products that automates mainframe server testing. *QA*Center helps measure, manage, and minimize the risk in mainframe servers by simplifying frequent mainframe server application testing. Solutions include:

*QA*Hiperstation to streamline the testing of traditional VTAM applications.

*QA*Center for IBM Enterprise Servers to test IBM WebSphere applications using HTTP, HTTP/S and APPC communications.

*QA*Center for IBM WebSphere MQ to test MQSeries applications.

*QA*Load

Today's distributed systems must perform reliably under loads ranging from hundreds to thousands of simultaneous users. Organizations must perform repeatable load testing and determine the ultimate

performance and potential limits of a system. Using loads that mimic realistic business usage, *QA*Load validates that the system meets acceptable service levels.

*QA*Load integrates with the following *QA*Center applications:

*QA*Director: Launch *QA*Load Conductor, Analyze, and Player modules from a convenient *QA*Director menu. Add, find, execute, and abort scripts as well as view results of *QA*Load tests from within *QA*Director.

*QA*Run

The time consuming and labor intensive task of testing e-commerce, ERP or distributed applications is difficult to perform accurately and thoroughly using only hit-or-miss, manual methods. *QA*Run provides automation capabilities necessary to quickly and productively create and execute test scripts, verify tests, and analyze test results.

Programmers create scripts that imitate the actions of a human tester. This keeps testing in sync with accelerated release cycles, improves the return on testing investment, and delivers the quality applications end users and management expect.

*QA*Run integrates with the following applications:

File-Aid/CS: Create, verify and restore data using the integration between *QA*Run and File-AID/CS. Use File-AID/CS to create datapools for data entry during testing. Invoke File-AID/CS's Compare feature directly from *QA*Run's built-in file check function. View the comparison results from the *QA*Run log files, the central point for verifying test results. Restore test databases using File-AID/CS, ensuring repeated tests are run without data modification.

*QA*Director: Launch *QA*Run from within *QA*Director. Add, create, edit, find, execute, and abort scripts to test native Windows applications, and distributed applications. Access these applications with a Windows client, or host based (mainframe) applications accessed via a Windows terminal emulation program. Additionally, view results and logs to analyze *QA*Run test data.

Reconcile

Reconcile helps accurately assess a project's status by keeping planning, development, and testing activities in sync. With Reconcile, test all requirements, and include development or testing changes, within the confines of a tight schedule.

Reconcile shares information with other tools that support the project lifecycle. This eliminates having to document similar information in multiple sources. Specifically, Reconcile integrates with the following *QA*Center applications:

*QA*Director: From within Reconcile, launch *QA*Director and import test results. Use *QA*Director with Reconcile to generate test plans in accordance with changing requirements.

TrackRecord: From within Reconcile, launch TrackRecord to synchronize requirements.

TestPartner

TestPartner is an automated functional testing tool specially designed for testing complex applications based on Microsoft, Java, and Web-based technologies. TestPartner's unique features allow testers and developers to create repeatable tests through visual scripting and automatic wizards. Users also have access to the full capabilities of Microsoft's Visual Basic for Applications (VBA), allowing tests to be as high-level or as detailed as necessary. From within TestPartner, submit defects to TrackRecord.

TrackRecord/Changepoint

TrackRecord is a defect tracking application that records and reports information about developed and supported products. TrackRecord records project information including project team members and testers, schedulers and milestones, bug reports, and feature requests in an Object database. TrackRecord's query and reporting features retrieve and format the information necessary to keep a project on track.

Additionally, you can use Changepoint to submit, edit and delete defects for systems and services.

TrackRecord integrates with the following *QA*Center applications:

Reconcile: From within Reconcile, launch TrackRecord to synchronize requirements.

*QA*Center Portal

The initial and primary focus of *QA*Center Portal is to provide a comprehensive reporting facility that is consistent among the *QA*Center point products. This will enable users, with various roles and responsibilities, to create and to view detail, summary, and cross-product reports. The *QA*Center Portal reporting engine ties together requirements planning, test planning, test execution, and defect tracking to provide users with a centralized view of application quality.

In addition to the report features, *QA*Center Portal provides the framework for web-based views and functionality to the *QA*Center point products. *QA*Center Portal supports defect tracking, as well as manual test execution and test planning.

*QA*Center Portal integrates with the following *QA*Center applications:

*QA*Director: Publish *QA*Director report data in *QA*Center Portal.Users can view these reports from any workstation with supported Microsoft Internet Explorer and Netscape browsers. *QA*Center Portal also provides the functionality to execute manual tests.

Because *QA*Director and *QA*Center Portal share the same database, users may be created and maintained in either product.

TrackRecord: Launch the TrackRecord web interface from *QA*Center Portal to enter, edit, and view defects.

Reconcile: Import requirements from Reconcile into *QA*Center Portal to review details and generate reports.

Changepoint: Track requests in Changepoint Request Management by installing and configuring *QA*Center Portal to work with the *QA*Center database. Prior to submitting a defect using Changepoint Request Management, it is recommended that you map projects between *QA*Center Portal and Changepoint Request Management.

About Testing Tools

Use Compuware testing tools or third party testing tools with *QA*Director.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Compuware Testing Tools

If using a Compuware testing tool such as *QA*Run, *QA*Hiperstation, File-AID/CS, or *QA*Load, set up a tool domain for the testing tool. A tool domain is a set of information entered about the testing tool. This information is used to browse, select, create, and execute scripts from within *QA*Director.

Third-Party Testing Tools

If using a third-party testing tool with *QA*Director, create a new integration for the tool. With this integration, it is possible to:

create or select a script to include in a test procedure

run the tool with the selected script

analyze the result to determine if the script passed or failed

communicate the result back to *QA*Director.

Optionally, the integration can help with browsing results of failed tests and editing scripts that were previously inserted.

Add the testing tool to be used first by clicking **File>New>Testing Tool**. The Add Tool Dialog Box appears. Use this to enter information about the tool, such as executable files.

Changepoint

About Using Changepoint Request Management

*QA*Director integrates with Changepoint Request Management to submit, edit and delete defects.

To use Changepoint Request Management, create an integration between *QA*Director and Changepoint Request Management. The Changepoint Request Management application opens and the Changepoint Request Management Submit Defect dialog box appears. Job information is automatically entered on the tabs for Request Details, Details, History, and Other Information. This information can be edited.

To integrate with Changepoint Request Management, install and configure *QA*Center Portal to work with the *QA*Center database.

Prior to submitting a defect using Changepoint Request Management, it is recommended that you map projects between *QA*Center Portal and Changepoint Request Management. For further information refer to the Changepoint Request Management documentation.

When you save the defect, a message appears to verify that the defect was successfully submitted. The defect appears in the **Defect Information Center** where you can edit it.

Note: If installing *QA*Director from the CARS CD, TrackRecord is not supported. However, if projects are migrated from 05.00.01 to 05.01.00, and already contain TrackRecord integrations, these **are** supported.

If installing *QA*Director from the *QA*Center CD, both TrackRecord and Request Management are supported.

Managing Changepoint Request Management Integrations

*QA*Director integrates with Changepoint Request Management to submit, edit and delete defects for systems and services. Only one Changepoint Request Management integration can be created. After a Request Management integration has been established, the **New Product Integration** option is not available until the integration is deleted.

Use this process to integrate with Request Management.

To set up a Changepoint Request Management integration with *QA*Director:

1. Open the **Project Information Center.**
2. Select **Defect Management** in the Project panel.
3. Select **Request Management**, and click **Browse**, or click **Tools>Manage Product Integrations**. The Manage Product Integrations dialog box appears.
4. Choose the **Request Management** folder from the **Integrations** list.
5. Click **New Product Integration**.
6. Type an integration name in the **Name** field.
7. Choose **Request Management** from the **Type** list.
8. Type the URL address where **Request Management** resides.
9. Type the following information in the appropriate fields:
    - **Server** field: *Changepoint server path*
    - **Database** field: *Changepoint database name*

- **User** field: *User name for the Changepoint database.*
- **Password** field: *Password for the Changepoint database.*

- Click the **Save** icon or **Save**.
- Click **OK**.

FileAID

Using File-AID: CS, CS Compare and ConverterPro with *QA*Director

File-AID/CS is a client/server test management tool used to create and to modify test data, to validate test data results, and to restore data to its baseline state prior to running a test. The **File-AID/CS Convert** facility is used to create specifications that contain all the data needed to convert an existing data file to a new target data file. The **File-AID/CS Compare** facility is used to create specifications that compare the expected and actual contents of output files.

Using File-AID/CS ConverterPro

File-AID/CS ConverterPro executes simple one to one conversions with local databases and flat files as well as complex conversions using remote MVS data types or any combination thereof. When you select **ConverterPro**, scripts are loaded from the repository to the **Script Information Center.** *QA*Director only works with File-Aid ConverterPro 3.2 and above.

Before you can view these scripts, you must configure the following:

*QA*Center Portal Database: See *Configuring QACenter Portal for SQL Server* in the *QA*Director-*QA*Center Portal Installation and Configuration Guide.

File-AID/CS in QACP: See the *Configuring Applications* topic in the QACP online help.

Configure the Repository Management Utility in File-AID. Refer to the documentation in File-AID.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Creating a File-AID/CS Specification

It is possible to open File-AID/CS from *QA*Director in order to create a specification.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To create a specification:

1. Go to the **Script Information Center**.
2. Choose File-AID/CS from the from the **Testing Tool** list.
3. Click **Actions>New Script**.
4. Create the specification within File-AID/CS.
5. Save the specification and exit File-AID/CS when finished.

Editing a File-Aid/CS specification

It is possible to open File-Aid/CS from *QA*Director in order to edit a specification.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To edit a specification:

1. Go to the **Script Information Center.**
2. Choose the appropriate File-Aid tool from the **Testing Tool** list.
3. Choose the script to edit.
4. Right-click and choose **Edit** to open the specification in File-AID/CS.
5. After editing the specification, save it and exit File-AID/CS.

Running File-AID/CS Specifications

In order to run a File-AID/CS specification from *QA*Director, first add the specification to a test procedure, and then run the test:

Note: This feature is not available if using *QA*Director with CARS Workbench.

To run a test procedure with File-AID/CS specifications:

1. Open the **Suite Information Center**.
2. In the Tree View, choose the test procedure that contains the File-AID/CS specifications.
3. Right-click and choose **Run**. The Job Description dialog box appears.
4. Select options on the Job Description dialog box. See Running a job for details.
5. Click **Submit**.

Viewing the Compare Log for File-AID/CS

Use the compare log to analyze results.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To view the compare log:

1. Open the **Job Information Center.**
2. Double-click a job. The Job Results screen appears.
3. Double-click on a job result. The Test Procedure dialog box appears.
4. Click on the **Logs** tab.
5. Double-click the appropriate icon for the script.

QACenter Portal

*QA*Center Portal provides a comprehensive reporting facility that is consistent among the *QA*Center point products. It enables users, with various roles and responsibilities, to create and view detail, summary, and cross-product reports. *QA*Center Portal ties together requirements planning, test planning, test execution, and defect tracking to provide users with a centralized view of application quality.

In addition to the report features, *QA*Center Portal provides the framework for web-based views and functionality to the *QA*Center products. *QA*Center Portal supports defect tracking, as well as manual test execution and test planning.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Generating Suites from *QA*Center Portal

Using *QA*Center Portal, it is possible to create suites and move them over to *QA*Director for use with testing.

The suites generated in *QA*Center Portal can define filters in *QA*Director, if the filtering feature is enabled in QACenter Portal. In *QA*Director, this filter takes the name of the suite created in *QA*Center Portal and appears in the filter list on the General tab of the **Test Class** dialog box when accessed from the Suite level.

Risk and Cycle attributes from the *QA*Center Portal generated suite appear on the Custom Attributes tab of the **Test Class** dialog Box or the **Procedure** dialog box.

To access the *QA*Center Portal click **Reports>Launch *QA*Center Portal.** Refer to *QA*Center Portal's documentation set for additional information about specific features and functionality.

Note: This feature is not available if using *QA*Director with CARS Workbench.

QALoad

Using *QA*Load with *QA*Director

*QA*Load is Compuware's load testing solution for distributed applications. *QA*Load can stress the distributed system by simulating thousands of users simultaneously performing different operations. Manage and execute *QA*Load tests from within *QA*Director.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To use *QA*Load with *QA*Director:

1. In *QA*Load, create one or more session ID files.
2. In *QA*Director, set up a *QA*Load Tool Domain.
3. In *QA*Director, right-click the ***QA*Load Tool Domain** and select **Properties**.
4. Click the Default Options tab.
5. Type or browse to the timing files directory path.

Note: The timing files directory path must be identical, including the drive letter, on every machine in the testing environment. If they differ, some machines can not access the timing files.

6. In *QA*Director, add the session ID files to test procedures.
7. From *QA*Director, execute the test procedures to launch *QA*Load's Conductor, which manages the testing activity.
8. When the job is complete, view the timing file in *QA*Load's Analyze component. The creation of the timing file determines whether the session passes or fails in *QA*Director. If a timing file is created, the session passes. If the timing file is not created, the session fails.

Adding *QA*Load Session ID Files to Test Procedures

In *QA*Director, do not run a session file on its own. Instead, insert the session ID file into a *QA*Director test procedure and then run the test procedure. Include any number of session ID files in a single test procedure. These instructions assume that a test suite with one or more test procedures already exists.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To add *QA*Load session ID files to test procedures:

1. Open the **Suite Information Center**.
2. Select a suite from the **Suite List**.
3. In the Suite Organization tree, double-click the test procedure to add to the session ID file. The Test Procedure dialog box appears.
4. On the **Scripts** tab, click **Insert**. The Add Test Scripts from Library dialog box appears.
5. In the **Testing Tool** list, choose ***QA*Load**.
6. The available *QA*Load tool domains appear in the list. Double-click the desired tool domain to display the session ID files.
7. Select the session ID file to add to the test procedure.
8. In the **Timeout** field, type the number of minutes to wait for the timing file to be created. If typing a 0, *QA*Director will wait indefinitely. The creation of the timing file determines whether the session file passes or fails in *QA*Director. In other words, if a timing file is created, the session file passes. If the timing file is not created, the session file fails.
9. Click **Add** to add the session file to the **Run Scripts** list.

Running *QA*Load Session ID Files

In order to run a *QA*Load session from *QA*Director, first add the session to a test procedure. Then run the test procedure like any other.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To run a *QA*Load session ID file:

1. Open the **Suite Information Center**.
2. In the Tree View, select the test procedure that contains the *QA*Load session files.
3. Choose the test procedure to run and right-click.
4. From the shortcut menu, choose **Run**. The Job Description dialog box appears.
5. Set the options on the Job Description dialog box.
6. Click **Submit**.
7. View job progress in the Job Information center.

Note: The creation of the timing file determines whether the session passes or fails in *QA*Director. In other words, if a timing file is created, the session passes. If the timing file is not created, the session fails.

Configuring DCOM on the *QA*Load Computer

If *QA*Director and *QA*Load are not installed on the same computer, configure DCOM on the *QA*Load computer. If unfamiliar with DCOM, refer to the following MSDN knowledge base articles:

Windows 95/98: article 182248

Windows NT: article 179615 and article 183607

Note: This feature is not available if using *QA*Director with CARS Workbench.

To configure DCOM on the *QA*Load computer:

1. On the computer where *QA*Load is installed, click **Start**>**Run**.
2. In the **Open** field, type **dcomcnfg** and click **OK**. The Distributed COM Configuration Properties dialog box appears.
3. On the **Applications** tab, select **LoadFileServer**.
4. Click **Properties**. The LoadFileServer Properties dialog box appears.
5. Click the **Location** tab.
6. Select the **Run application on this computer** check box. Clear all other check boxes.
7. Click the **Security** tab. On this tab, specify which users have permission to access, to launch, and to configure *QA*Load.

   If adding user names, make sure the names are valid on the domain or on the local computer (where *QA*Load is installed).
8. Click the **Identity** tab.
9. Select the **The launching user** option. This means *QA*Load will run using the security context of the user who started *QA*Load from *QA*Director.
10. Click **OK** to close the Properties dialog box.
11. Click **OK** to close the Distributed COM dialog box.

Hint: If setting environment variables for *QA*Load, such as a license variable, it is best to set them as system variables, rather than as user variables. System variables are valid for all users who log onto the *QA*Load computer.

Reviewing timing files

After running a *QA*Load test, open the timing file.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To open the timing file:

1. Go to the **Job Information Center**.
2. Click on the **Results** tab.
3. Double-click a job. The Job Results screen appears.
4. Double-click a test procedure. The Test Procedure dialog box appears.
5. Click on the **Logs** tab.
6. Double-click the **.qaload-view** file to display the timing file in *QA*Load's Analyze component. The creation of the timing file determines whether the session passes or fails in *QA*Director. That is, if a timing file is created, the session passes. If the timing file is not created, the session fails.

QARun

Using *QA*Run With *QA*Director

*QA*Run can be used to test native Windows applications, distributed applications with a Windows client, or host-based (mainframe) applications accessed via a Windows terminal emulation program. *QA*Run also includes advanced Internet testing capabilities to support the latest Web technologies.

File Checks: If the *QA*Run scripts include file checks, which compare the expected and actual contents of an output file using Compuware's File-AID/CS Compare component, install File-AID/CS on a workstation or on an available network drive.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Analyzing Scripts in *QA*Run's Log View

If the *QA*Run run environment includes logging, analyze the script results in *QA*Run's log view.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To analyze scripts in the log view:

1. Go to the **Job Information Center**.
2. Click on the **Results** tab.
3. Double-click a job. The **Job Results** screen appears.
4. Double-click a test procedure. The Test Procedure dialog box appears.
5. Click on the **Logs** tab.
6. Double-click the ***scriptname*.qarun-view** icon. The Log View opens and displays the script's log.

Note: If moving *QA*Run script results to the Trash folder and emptying the trash, the result information is deleted from the *QA*Run database as well as the *QA*Director database.

Creating a *QA*Run Script

Open *QA*Run from *QA*Director to create a script.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To create a script:

1. Open the **Script Information Center**.
2. Select the tool domain where the new script will be stored.
3. Click **Actions>New Script**.
4. *QA*Run opens with a new script ready to be recorded. After creating the script, save it and exit *QA*Run.
5. In *QA*Director, click **Refresh** to display the new script.

It is possible to open *QA*Run from *QA*Director in order to edit a script.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To edit a script:

1. Go to the Script Information Center.
2. Choose the script to edit.
3. Right-click and choose **Edit** to open the script in *QA*Run.
4. After editing the script, save it and exit *QA*Run.

Running *QA*Run scripts

In order to run a *QA*Run script from *QA*Director, first add the script to a test procedure. Then run the test procedure like any other.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To run a script and view results:

1. Go to the **Suite Information Center**.
2. From the treeview, choose the test procedures that contains the *QA*Run scripts.
3. Choose the test procedure to run and right-click.
4. From the shortcut menu, choose **Run**. The Job Description dialog box appears.
5. Set options on the Job Description dialog box.
6. Click **Submit**.
7. View job progress in the **Job Information Center**.

Reconcile

Using Reconcile With *QA*Director

*QA*Director integrates with Reconcile, Compuware's requirements management planning solution. Reconcile helps organizations implement a disciplined approach to capturing, analyzing, and communicating product and testing requirements to eliminate costly schedule slips and application failures. With Reconcile, organizations can control and communicate requirements changes across the application lifecycle to help meet the ever-increasing time to market and quality demands of their end-users.

Note: This feature is not available if using *QA*Director with CARS Workbench.

The integration between *QA*Director and Reconcile helps ensure a quality process by validating tests against requirements, making sure original requirements were met.

To learn more about the integration between *QA*Director and Reconcile, refer to the Reconcile online help and user documentation.

Using Reconcile With *QA*Director

*QA*Director integrates with Reconcile, Compuware's requirements management planning solution. Reconcile helps organizations implement a disciplined approach to capturing, analyzing, and communicating product and testing requirements to eliminate costly schedule slips and application failures. With Reconcile, organizations can control and communicate requirements changes across the application lifecycle to help meet the ever-increasing time to market and quality demands of their end-users.

Note: This feature is not available if using *QA*Director with CARS Workbench.

The integration between *QA*Director and Reconcile helps ensure a quality process by validating tests against requirements, making sure original requirements were met.

To learn more about the integration between *QA*Director and Reconcile, refer to the Reconcile online help and user documentation.

Managing Reconcile Integrations

Use this process to integrate with Reconcile, Compuware's requirements management planning solution. Reconcile helps organizations implement a disciplined approach to capturing, analyzing, and communicating product and testing requirements to eliminate costly schedule slips and application failures. The integration between *QA*Director and Reconcile helps ensure a quality process by validating tests against requirements, making sure original requirements were met.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To set Reconcile integrations with *QA*Director:

1. Click **Tools>Manage Product Integrations**. The Manage Product Integrations dialog box appears.
2. Choose the Reconcile folder from the **Integrations** list.
3. Click **New**.
4. Type an integration name in the **Name** field.
5. Choose Reconcile from the **Type** list.
6. Type the appropriate product server and path in the **Server** field.
7. Type the appropriate product database name in the **Database** field.
8. Type the user name for the appropriate product in the **User** field.
9. Type the password for the appropriate product in the **Password** field.
10. Click the **Save** icon or **Save**.
11. Click **OK**.

TestPartner

Using TestPartner With *QA*Director

Compuware's TestPartner is an automated functional testing tool that is specifically designed for testing Java, Visual Basic, and Visual C++ applications, as well as browser-based Web applications.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Analyzing Results in TestPartner's Log View

If the TestPartner playback environment includes logging, analyze script results in TestPartner's log view:

Note: This feature is not available if using *QA*Director with CARS Workbench.

To analyze in log view:

1. Go to the **Job Information Center**.
2. Click on the **Results** tab.
3. Double-click a job. The **Job Results** screen appears.
4. Double-click a test procedure. The Test Procedure dialog box appears.
5. Click on the **Logs** tab.
6. Double-click the ***scriptname*.tp-view** icon. The TestPartner log view opens and displays the script's log.

Creating a TestPartner Script

Open TestPartner from *QA*Director to create a script.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To create a new script:

1. Open the **Script Information Center**.

2. Select the tool domain where the new script will be stored.
3. Click **Actions>New Script**. When prompted, type a name for the new script and click **OK**. (The script name cannot contain spaces or special characters such as the # or @ symbols.)
4. TestPartner opens with a new script ready to be recorded.
5. After creating the script, save it and exit TestPartner.
6. In *QA*Director, click **Refresh** to display the new script.

Editing a TestPartner Script

It is possible to open TestPartner from *QA*Director in order to edit scripts.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To edit a script:

1. Go to the Script Information Center.
2. Choose the script to edit.
3. Right-click and choose **Edit** to open the script in *TestPartner*.
4. After editing the script, save it and exit TestPartner.

Running TestPartner Scripts

In order to run a TestPartner script from *QA*Director, first add the script to a test procedure. Then run the test like any other.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To run a TestPartner script:

1. Select the test procedure that contains the TestPartner scripts.
2. Select the test procedure to run and right-click.
3. From the shortcut menu, choose **Run**. The Job Description dialog box appears.
4. Set the options on the Job Description dialog box.
5. Click **Submit**.

TrackRecord

Integrating *QA*Director with a Customized TrackRecord Database

When *QA*Director submits a defect into TrackRecord, it will attempt to import data into two types that are shipped with TrackRecord's default schema: *QA*Center Reported Defect and *QA*Director Test.

Note: This feature is not available if using *QA*Director with CARS Workbench.

In *QA*Director's Custom\win32 directory, there are two integration files for TrackRecord:

trqacenter_item.txt contains the mapping of *QA*Director attributes to tags in the TrackRecord *QA*Center Reported Defect type.

trqadirector_item.txt contains the mapping of *QA*Director attributes to tags in the TrackRecord *QA*Director Test type.

To set up *QA*Director's custom integration with a customized TrackRecord database:

1. Customize the TrackRecord database. Complete all editing before adding data.

2. Associate the QC_Defect tag with the *QA*Center Reported Defect type or its equivalent. This will identify this type to *QA*Director as the defect type to which it should send its information.
3. Associate the QC_Test tag with the *QA*Director Test type or its equivalent. This will identify this type to *QA*Director as the test information type contained in TrackRecord.
4. Open the trqacenter_item.txt and trqadirector_item.txt files.
5. In TrackRecord, apply any field tags listed in the files that you wish to use in the types.
6. If there are any unused tags that are listed in the *QA*Director text files, delete the attribute and tag lines. Each attribute must be mapped to a tag. If there are any tags that appear in these files that are not used, the integration will not work.
7. Save and close the trqacenter_item.txt and trqadirector_item.txt files.
8. In TrackRecord's *QA*Center Reported Defect type, ensure that the QC_Defect_Test field tag is applied to a single-item combination box and that field is pointed to the *QA*Director Test type.

   In the default schema shipped with TrackRecord, this field tag is associated with the Test single-item combination box. This will create a link between the *QA*Center Reported Defect and the *QA*Director Test types and allow TrackRecord to pass its internal database identifier to *QA*Director. When the integration is finished, there will be a link to the corresponding *QA*Center Reported Defect in each test submitted from *QA*Director.
9. Test the integration by submitting a defect from *QA*Director to TrackRecord.

Managing TrackRecord Integrations

Use this process to integrate with TrackRecord.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To set TrackRecord integrations with *QA*Director:

1. Open the **Project Information Center.**
2. Select **Defect Management** in the Project panel.
3. Select **TrackRecord** and click **Browse**, or, click **Tools>Manage Product Integrations**. The Manage Product Integrations dialog box appears.
4. Choose the TrackRecord folder from the **Integrations** list.
5. Click **New**.
6. Type an integration name in the **Name** field.
7. Choose TrackRecord from the **Type** list.
8. Type the appropriate product server and path in the **Server** field.
9. Type the appropriate product database name in the **Database** field.
10. To access QACP reports, type the QACP user name in the **User** field.
11. To access QACP reports, type the QACP password in the **Password** field.
12. Click the **Save** icon or **Save**.
13. Click **OK**.
14. From the User panel, double click your user name in the **User** panel. The User Properties dialog box appears.
15. On the Single Sign On tab locate the appropriate integration and enter the TrackRecord password and user name.
16. Click **OK**.

Mainframe

Using *QA*Hiperstation With *QA*Director

*QA*Hiperstation is Compuware's testing tool for VTAM-based applications. From within *QA*Director, browse *QA*Hiperstation scripts on the mainframe, add scripts to test procedures, launch *QA*Hiperstation to play back the scripts, and view details about failed *QA*Hiperstation scripts. In addition, if *QA*Hiperstation+ is installed on the computer, edit and even create *QA*Hiperstation scripts right from *QA*Director.

Note: This feature is not available if using *QA*Director with CARS Workbench.

QAHiperstation

Using *QA*Hiperstation With *QA*Director

*QA*Hiperstation is Compuware's testing tool for VTAM-based applications. From within *QA*Director, browse *QA*Hiperstation scripts on the mainframe, add scripts to test procedures, launch *QA*Hiperstation to play back the scripts, and view details about failed *QA*Hiperstation scripts. In addition, if *QA*Hiperstation+ is installed on the computer, edit and even create *QA*Hiperstation scripts right from *QA*Director.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Configuring the QA 3270 Emulator Server

The QA 3270 Emulator Server is a utility program that manages the communication between the 3270 mainframe emulator and Compuware products such as *QA*Director and *QA*Hiperstation+. After installation and setup, the QA 3270 Emulator Server runs silently and requires no direct interaction from the user.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Installing QA 3270 Emulator Server

To install the QA 3270 Emulator Server, select the **S/390 Support** optional component during the *QA*Director installation. The S/390 Support option installs the following items:

QA 3270 Emulator Server: This server manages the communications between the 3270 emulator and *QA*Director when accessing and executing *QA*Hiperstation scripts and MVS batch jobs on the mainframe.

TN3270 Mainframe Emulator: Use this free emulator for mainframe testing with *QA*Director and *QA*Hiperstation+.

If the *QA*Director administrator did not install S/390 Support during the initial *QA*Director installation, perform a modified installation now to add the S/390 Support component. Refer to the *QADirector Installation and Configuration Guide* for details.

Configuring QA3270 Emulator Server

Configure the QA 3270 Emulator Server to work with the mainframe emulator. The QA 3270 Emulator Server readme file details the supported emulators, version requirements, and configuration instructions. To view the readme file, click **Start>Program Files>Compuware>*QA*Director>QA 3270 Emulator**

Server>Read Me. In Windows XP, click **Start>All Program Files>Compuware>*QA*Director>QA 3270 Emulator Server>Read Me**.

To configure the QA 3270 Emulator Server, click **Start>Program Files>Compuware>*QA*Director>QA 3270 Emulator Server>Configuration Utility**. In Windows XP, click **Start>All Programs>Compuware>*QA*Director>QA 3270 Emulator Server>Configuration Utility**. Select an emulator and click **OK** to view the configuration settings for the emulator. Follow the instructions in the readme file and in the configuration utility's online help.

If not configuring the QA 3270 Emulator Server to use a different emulator, the TN3270 emulator is used by default. Enter a host name/IP address on the TN3270 configuration dialog box.

Creating a *QA*Hiperstation Script

Open *QA*Hiperstation+ from *QA*Director to create a script.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To create a new script:

Note*: QA*Hiperstation+ must be installed on the same computer as *QA*Director.

1. Open the **Script Information Center**.
2. Select the tool domain where the new script will be stored.
3. Click **Actions>New Script**. *QA*Hiperstation+ opens with a new script ready to be recorded.
4. After creating the script, save it and exit *QA*Hiperstation+.
5. In *QA*Director, click **Refresh** to display the new script.

Editing a *QA*Hiperstation Script

It is possible to open *QA*Hiperstation+ from *QA*Director in order to edit a script.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To edit a script:

Note: *QA*Hiperstation+ must be installed on the same computer as *QA*Director.

1. Choose the script to edit.
2. Right click and choose **Edit** to open the script in *QA*Hiperstation+.
3. After editing the script, save it and exit *QA*Hiperstation+.

Creating a *QA*Hiperstation Tool Domain

Before using *QA*Hiperstation with *QA*Director, create at least one *QA*Hiperstation tool domain. The tool domain provides *QA*Director with host connection information, job statement data, and *QA*Hiperstation-specific information such as wait time specifications.

Note: This feature is not available if using *QA*Director with CARS Workbench.

1. From the **Centers** toolbar, click the **System Administration Center** icon.
2. Click **Action>Add**. The Tool Domain Detail dialog box displays.

   Note: Tool Domain Permissions: If you receive a message that you do not have permission to view, to add, to modify, or to delete tool domains, contact the *QA*Director administrator. The *QA*Director administrator assigns permissions.

4. Type a tool domain name in the **Name** field.
5. Type a description of the tool domain in the **Description** field.

6. Select *QA*Hiperstation from the **Type** list.
7. Select the **Public** check box to make the tool domain available to other *QA*Director users.
8. On the **General** tab, enter all other requested information.

9. Click the **Connect Using** tab:

Emulator Session: Select the default emulator session to use when running scripts associated with this tool domain. It is possible to change this default before running the scripts.

Logon Macro: Click **Browse** and select a macro for logging onto the emulator session. If entering a logon macro but not setting up **Auto Logon**, *QA*Director will start the TSO logon and prompt the tester to enter their user ID and password. Thus, someone must be present during the test execution. If running an *unattended* mainframe job, complete the **Auto Logon** fields and select a logon macro that includes the keyword `[getuipw]` (get user ID and password) in order to properly obtain the user's TSO ID and password. For an example of the required syntax, see the sample logon macro provided with *QA*Director in `\Program Files\Compuware\QADirector\Custom\Win32\hson.mac`.

Auto Logon: Select this check box to set up automatic TSO logon, which is used when running unattended mainframe jobs. This check box is disabled if no macro is selected in the **Logon Macro** field.

Emulator Session Shortcut: Enter the information needed to start the emulator session. Type the session path and name in addition to the executable path and name. For example, for an EXTRA! Personal Client session, type the location and name of the EXTRA! executable as well as the location and name of the .EDP file. Separate the two paths with a space, as follows: `C:\Program Files\E!PC\Extra.Exe C:\Program Files\E!PC\Sessions\Session1.EDP`.

Note: If using the TN3270 emulator, type `TN3270` in this field. *QA*Director may have entered this automatically.

User ID: Enter the **&tsouserid** variable in this field. At execution time, *QA*Director will use the ID entered in the tester's User Preferences. It is possible to type a specific user ID in this field if executing all jobs under the same ID. This means that if several team members use this tool domain to run jobs, all the jobs will be executed under the same name. This field is mandatory in order to run unattended mainframe jobs.

Password: Enter the **&tsopassword** variable in this field. At execution time, *QA*Director will use the password entered on the tester's User Preferences. It is possible to type a specific password in this field if executing all jobs under the same password. This means that if several team members use this tool domain to run jobs, all the jobs will be executed under the same name. This field is mandatory in order to run unattended mainframe jobs.

Logoff Macro: Click **Browse** and select a logoff macro to automatically log off the emulator session.

10. Click the **Job Control** tab:

Job Statement: If entering user-specific MVS job control information in User Preferences, use the following variables in the tool domain to obtain the user-specific information during execution:

&jobname: Obtains job name from User Preferences.
&accounting: Obtains accounting information from User Preferences.
&name: Obtains user name or run name from User Preferences.
&class: Obtains job class from User Preferences.
&msgclass: Obtains job scheduler message output class from User Preferences.

PGM: Type the JCL statement specifying the program name.

STEPLIB: Type the PDS that contains the *QA*Hiperstation program.

Other JCL: Type any other JCL required, such as files used by REXX statements.

DSN: Type the RunLog dataset name. This is a required field.

Unit: Type a unit. This is a required field with a maximum length of 8 characters. The default is SYSDA.

Volume: Type a volume. This field is optional and has a maximum length of 6 characters.

TPF: Type the domain destination, which is the VTAM name that the script will be run under. When choosing this option, the *QA*Hiperstation *group* properties are applied to the group statement.

Attach user's high level dataset qualifier: Select this check box in order to attach the user's high-level dataset qualifier to datasets such as the RunLog and the compare log. The user's high-level dataset qualifier is defined in

User Preferences. Or, if everyone uses the same high level dataset qualifier, type it on the **Global Job Information** tab on this dialog box.

Retention period: Enter the number of days between 0 and 9999 that the data set should be retained. The data set cannot be deleted during the retention period. This field is optional.

11. Click the ***QA*Hiperstation** tab:

Wait Times: Specify the time interval that *QA*Hiperstation will wait for various responses or processes to complete.

Other Options: Select the check boxes next to the desired options.

APPLID Node Data: Type the prefix and suffix used for *QA*Hiperstation APPLIDS defined during the *QA*Hiperstation installation.

Dynamic Reports Datasets: Specify the unit and volume on which the Journal or Log datasets will be dynamically allocated.

Format of Date Fields: Select the format to use for date fields.

12. Click the **Global Job Information** tab.

This tab is useful for entering *QA*Hiperstation, load library, and high level dataset qualifier information in **one location**, rather than in each person's User Preferences. Thus, the information is common for everyone who uses this tool domain. If preferred that *QA*Director obtain this information from the tester's User Preferences, enter variables in these fields instead:

&qahsrelease: Obtains *QA*Hiperstation release from User Preferences.
&qahscommand: Obtains command line from User Preferences.
&loadlib: Obtains load library from User Preferences.
&hldq: Obtains high level dataset qualifier from User Preferences.

Do **not** leave these fields blank. Either enter the specific information or enter variables. If specific information is typed in both the tool domain **and** in User Preferences, the information in the tool domain is used at run time.

13. Click **OK** to create the new tool domain.

Creating *QA*Hiperstation Groups

Occasionally, it may be necessary to group *QA*Hiperstation test scripts together, so that they are executed as one job on the mainframe. It is possible to then add the test script group to a test procedure. Remember, the test procedure can contain a combination of test scripts and test script groups.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To create a *QA*Hiperstation test script group:

1. Open the **Script Information Center**.
2. From the Testing Tool list choose MVS Batch. If this is the first time the user is accessing this, the **Filter dialog** appears. Enter an MVS Batch filter. Click Enter.
3. The **Select 3270 Emulator Server dialog Box** appears. Verify that the emulator is positioned at the ISPF Primary Menu with the terminal cursor on the command line.
4. Choose the 3270 session to use to connect to the mainframe and click Ready.
5. Click **Actions> New Group.** Rename *New Group* to an appropriate name and double-click on it. The *QA*Hiperstation dialog box appears.
6. On the **General** tab, type a description for the test script group.
7. Click the **Scripts** tab. On this tab, select the scripts to include in the group.
8. Under **Group Options**, select the appropriate option:

   One group statement for all test scripts: Select this option to make *QA*Director execute the test scripts in a series. Type the Domain Destination (TPF), which is the VTAM name that the script will

be run under. When choosing this option, the *QA*Hiperstation *group* properties are applied to the group statement.

One group statement for each test script: Select this option to make *QA*Director execute the test scripts in parallel. When choosing this option, the *QA*Hiperstation *script* properties are applied to the group statement.

9. Review the **Preview** tab. This tab includes a complete view of all selected test elements to be executed, along with the job control statements (for *QA*Hiperstation and MVS batch job test scripts) and tool domain data associated with them.

10. Click **OK** to return to the **Script Information Center**.

11. In the **Script Information Center**, the new group is listed under **MVSBatch Groups** in the tool domain.

12. Right click the group and click **Add to Library**.

Entering *QA*Hiperstation Setup Information in User Options

In User Options, it is possible to specify which *QA*Hiperstation release and MVS load library is in use. At the same time, it is possible to enter MVS job control statement information.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Hint: If everyone on the team uses the same *QA*Hiperstation release, MVS load library, and/or high level dataset qualifier, consider entering this information on the **Global Job Information** tab in the *QA*Hiperstation tool domain. This makes it possible to enter the information in **one location** only, rather than in each person's User Options.

1. Start *QA*Director.
2. Click **Tools>User Options**.
3. Click the ***QA*Hiperstation** tab:

   ***QA*Hiperstation release Information**: Select a *QA*Hiperstation release.

   ***QA*Hiperstation Command Line**: Type the command to start *QA*Hiperstation from the ISPF Primary Panel load library. Obtain this information from your *QA*Hiperstation mainframe administrator. This field is applicable only if submitting File Manager items with scripts.

   MVS Load Library: Type the MVS load library to use. This field is mandatory.

   Attach High Level Dataset Qualifier in front: Select this check box for *QA*Director to attach a high level dataset qualifier in front of all new dataset names. Type a valid dataset qualifier on the **MVS Job Control** tab on this dialog box.

   TSO Logon Information: In this section, enter user ID and password for logging onto TSO. This makes it possible to enter variables in the *QA*Hiperstation tool domain when setting up automatic TSO logon.

4. Click the **MVS Job Control** tab:

   Job Name: Type the name to assign to *QA*Hiperstation jobs on the mainframe. This make it possible to use the **&jobname** variable in the job statement field on the tool domain.

   Accounting Data: Type any established accounting information, such as an account number to which to charge the job. In the job statement field on the tool domain, use the **&accounting** variable for this information.

   Enclose Accounting Data within parentheses: Select this check box to enclose accounting data in parentheses.

   Name: Type a name to identify the user or the run. In the job statement field on the tool domain, use the **&name** variable for this information.

   Job Class: Type the job class. In the job statement field on the tool domain, use the **&class** variable for this information.

   Message Class: Type the job scheduler message output class. In the job statement, use the **&msgclass** variable for this information.

High Level Dataset Qualifier: If necessary, specify a high-level dataset qualifier to be placed at the beginning of any dataset name that is generated by *QA*Director. Be sure to select the **Attach High Level Dataset Qualifier in front** check box on the ***QA*Hiperstation** tab.

5. Click **OK**.

Running *QA*Hiperstation Scripts

To run a *QA*Hiperstation script from *QA*Director, first add the script to a test procedure.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To run *QA*Hiperstation scripts:

1. Open the **Suite Information Center**.
2. Select the test procedure that contains the *QA*Hiperstation scripts and right-click.
3. Choose **Run**. The Job Description dialog box appears.
4. Set the options on the Job Description dialog box. See Running a job for general information about the job description.
5. Click **Submit**.

Viewing a *QA*Hiperstation Compare Log

When adding a script to a *QA*Director test procedure, it is possible to request a Compare Log for the script. If requested, open the script's Compare Log in *QA*Hiperstation+ after the job is complete.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To view the compare log:

1. Open the **Job Information Center.**
2. Double click a job. The Job Results screen appears.
3. Double click on a job result. The Test Procedure dialog box appears.
4. Click on the **Logs** tab.
5. Double-click the **Compare Log** icon for the script. *QA*Hiperstation+ opens and displays the compare log for the script.

MVSBatch

Using MVS Batch Jobs with *QA*Director

If you have *QA*Hiperstation, use *QA*Director to run MVS batch jobs. When running the job, *QA*Director will submit the batch job to the mainframe for execution.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Creating MVS Batch Groups

Group MVS batch jobs together, so that they are executed as one job on the mainframe. An MVS batch group can contain both MVS batch jobs and *QA*Hiperstation scripts.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To create a MVS batch job group:

1. Open the **Script Information Center**.

2. From the Testing Tool list choose MVS Batch. If this is the first time the user is accessing this, the **Filter dialog** appears. Enter an MVS Batch filter. Click Enter.
3. The **Select 3270 Emulator Server dialog Box** appears. Verify that the emulator is positioned at the ISPF Primary Menu with the terminal cursor on the command line.
4. Choose the 3270 session to use to connect to the mainframe and click Ready.
5. Click **Actions> New Group.**
6. Rename *New Group* to an appropriate name and double click on it. The MVS Batch dialog Box appears.
7. Click on the **General** tab. Type a description.
8. Click on the **MVS Batch Options** tab. Type a return code.
9. Click the **Scripts** tab. On this tab, choose the MVS batch scripts and *QA*Hiperstation scripts to include in the group.
10. Double-click the appropriate tool domain to display the test scripts.
11. Click **Add** to add scripts to the Scripts in Group list. You can add both MVS batch jobs and *QA*Hiperstation scripts to an MVS batch group.
12. Review the **Preview** tab. This tab includes a complete view of all selected test elements to be executed, along with the job control statements (for *QA*Hiperstation and MVS batch job test scripts) and tool domain data associated with them.
13. Click **OK** to return to the **Script Information Center**.
14. In the **Script Information Center**, the new group is listed under **MVSBatch Groups** in the tool domain.
15. Right click the group and click **Add to Library**.

Entering MVS Batch Setup Information in User Options

In User Options, it is possible to specify which *QA*Hiperstation release and MVS load library is in use. At the same time, it is possible to enter MVS job control statement information.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Hint: If everyone on the team uses the same *QA*Hiperstation release, MVS load library, and/or high level dataset qualifier, consider entering this information on the **Global Job Information** tab in the *QA*Hiperstation tool domain. This makes it possible to enter the information in **one location** only, rather than in each person's User Options.

1. Start *QA*Director.
2. Click **Tools>User Options**.
3. Click the ***QA*Hiperstation** tab:

   ***QA*Hiperstation release Information**: Select a *QA*Hiperstation release.

   ***QA*Hiperstation Command Line**: Type the command to start *QA*Hiperstation from the ISPF Primary Panel load library. Obtain this information from your *QA*Hiperstation mainframe administrator. This field is applicable only if submitting File Manager items with scripts.

   MVS Load Library: Type the MVS load library to use. This field is mandatory.

   Attach High Level Dataset Qualifier in front: Select this check box for *QA*Director to attach a high level dataset qualifier in front of all new dataset names. Type a valid dataset qualifier on the **MVS Job Control** tab on this dialog box.

   TSO Logon Information: In this section, enter user ID and password for logging onto TSO. This makes it possible to enter variables in the *QA*Hiperstation tool domain when setting up automatic TSO logon.

4. Click the **MVS Job Control** tab:

   Job Name: Type the name to assign to *QA*Hiperstation jobs on the mainframe. This make it possible to use the **&jobname** variable in the job statement field on the tool domain.

   Accounting Data: Type any established accounting information, such as an account number to which to charge the job. In the job statement field on the tool domain, use the **&accounting** variable for this information.

Enclose Accounting Data within parentheses: Select this check box to enclose accounting data in parentheses.

Name: Type a name to identify the user or the run. In the job statement field on the tool domain, use the **&name** variable for this information.

Job Class: Type the job class. In the job statement field on the tool domain, use the **&class** variable for this information.

Message Class: Type the job scheduler message output class. In the job statement, use the **&msgclass** variable for this information.

High Level Dataset Qualifier: If necessary, specify a high-level dataset qualifier to be placed at the beginning of any dataset name that is generated by *QA*Director. Be sure to select the **Attach High Level Dataset Qualifier in front** check box on the ***QA*Hiperstation** tab.

5. Click **OK**.

Running MVS Batch Jobs

Run a MVS Batch job which is submitted to the mainframe for execution.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To run an MVS batch job:

1. In the Treeview, select the test procedure that contains the MVS Batch Jobs.
2. Click **Run>Run <name of test procedure>**.
3. Set options on the Job Description dialog box. See Running a job for more information.
4. Click **Submit**.

TSO Logon and Logoff Macros

About Logon and Logoff Macros

Create macros that automate the logon and logoff to TSO. This is helpful for executing unattended jobs on the mainframe.

To get started, *QA*Director includes sample macros named **HSON.MAC** and **HSOFF.MAC** in `\program files\compuware\QADirector\custom\win32`. Customize these macros to reflect the site's system configuration, or write your own macros.

To customize or write the macros, use any text editor and any valid macro language such as the navigation macro language whose commands and control keys are described in this topic.

After creating the logon and logoff macros, run unattended mainframe jobs. See Running an unattended job for details.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Rules for Writing Macros

Note: This feature is not available if using *QA*Director with CARS Workbench.

Consider the following rules when writing macros:

Macro files must have the extension .mac.

All commands and 3270 control keys must be enclosed in brackets "[ ]".

All keywords can be entered in upper, lower, or mixed case.

All strings or text must be enclosed in single quotes (' ').

Multiple commands and 3270 keys may be entered on one line.

Navigation Macro Language 3270 Control Keys

The following is a list of the navigation macro language 3270 control keys.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Description	Control key
Attention Interrupt	[ATTN]
Backward or Reverse Tab	[BACKTAB]
Clear Key	[CLEAR]
Delete Key	[DELETE]
Down Arrow	[DOWNARROW]
Enter Key	[ENTER]
Forward Tab	[TAB]
Insert Key	[INSERT]
Left Arrow	[LEFTARROW]
PF1 to PF24	[PFnn]
Program Attention Keys	[PA1] [PA2]
Reset Key	[RESET]
Right Arrow	[RIGHTARROW]
Space	[SPACE]
Up Arrow	[UPARROW]

TSO Logon and Logoff Macro Examples

Use the following examples as a guide to creating TSO logon and logoff macros.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Example: TSO Logon Macro

This example is taken from the HSON.MAC file provided with *QA*Director. Remember, you can simply modify this file for your own site.

```
[WAIT] 30 0 0 'C O R P O R A T E D A T A C E N T E R'
[getuipw] 'Enter your TSO User ID & Password'
[TYPE] 'A B' [LEFTARROW]
[typeui] [ENTER]
[WAIT] 15 0 0 'TSO/E LOGON'
[typepw] [ENTER]
[WAIT] 30 0 0 'ISPF Primary Option Menu'
```

Example: TSO Logoff Macro

This example is taken from the HSOFF.MAC file provided with *QA*Director. Remember, you can simply modify this file for your own site.

```
[WAIT] 0 0 0 'ISPF Primary Option Menu'
[TYPE] '=X' [ENTER]
[WAIT] 15 0 0 'READY'
[TYPE] 'LOGOFF' [ENTER]
[WAIT] 15 0 0 'C O R P O R A T E D A T A C E N T E R'
```

Navigation Macro Language Commands

Navigation Macro Language Commands

Use the following navigation macro language commands when creating TSO macros.

End

Get User ID and Password

Input

Type

Type Input

Type Password

Type User ID

Wait

Navigation Macro Language Commands

Use the following navigation macro language commands when creating TSO macros.

End

Get User ID and Password

Input

Type

Type Input

Type Password

Type User ID

Wait

End Command

An optional command used to indicate the end of an automated script.

[End]

Get User ID and Password Command

Displays a pop-up window that prompts the user to enter a User ID and Password. The `[TypeUI]` and `[TypePW]` commands can be used to copy the **User ID** and **Password** fields to the current cursor position on the terminal emulation screen.

[GetUIPW] <caption>

where:

<caption> string that is placed in the title or heading of the pop-up window.

Input Command

Displays a pop-up window that prompts the user to enter a string. The entered string is then saved and can be written to the terminal emulation screen with the `[TypeInput]` command.

[Input] <caption> <prompt>

where:

<caption> a string that is placed in the title or heading of the pop-up window.

<prompt> a string that is the prompt for the user-entered text field on the pop-up window.

Type Command

Used to enter text at the current cursor position on the terminal emulation screen.

[Type] <text>

where:

<text> any string enclosed in quotes.

Type Input Command

Copies the resulting text from the last `[Input]` command to the current cursor position on the terminal emulation screen.

[TypeInput]

Type Password Command

Copies the Password from the last `[GetUIPW]` command to the current cursor position on the terminal emulation screen.

[TypePW]

Type User ID Password Command

Copies the User ID from the last `[GetUIPW]` command to the current cursor position on the terminal emulation screen.

[TypeUI]

Wait Command

Used to wait for a given number of seconds. Can also be used to search for a string. If the optional "row," "column," and "wait for text" parameters are specified, the command waits for the specified number of seconds or until the specified string is found. An error will occur if a "wait for text" string is not found.

[Wait] <number of seconds> {<row> <column> <wait for text>}

where:

<number of seconds>: (0-999) the number of seconds to wait. After waiting the given number of seconds, the automated script will resume with the next statement. If the other parameters are present, this is the maximum wait time. Setting this parameter to '0' (zero) will cause the command to wait forever unless the 'wait for text' string is found.

<row>: (0-43) the starting search row. If set to '0' (zero), **wait for text** can start in any row.

<column>: (0-132) the starting search column. If set to '0' (zero), **wait for text** can start in any column.

<wait for text>: If this text is not found at the defined row/column position within the given number of seconds, an error message displays, and the automated script terminates. When the text is found, the script resumes with the next statement. Enter the text exactly as it will appear on the 3270 screen.

Request Management

About Using Changepoint Request Management

*QA*Director integrates with Changepoint Request Management to submit, edit and delete defects.

To use Changepoint Request Management, create an integration between *QA*Director and Changepoint Request Management. The Changepoint Request Management application opens and the Changepoint Request Management Submit Defect dialog box appears. Job information is automatically entered on the tabs for Request Details, Details, History, and Other Information. This information can be edited.

To integrate with Changepoint Request Management, install and configure *QA*Center Portal to work with the *QA*Center database.

Prior to submitting a defect using Changepoint Request Management, it is recommended that you map projects between *QA*Center Portal and Changepoint Request Management. For further information refer to the Changepoint Request Management documentation.

When you save the defect, a message appears to verify that the defect was successfully submitted. The defect appears in the **Defect Information Center** where you can edit it.

Note: If installing *QA*Director from the CARS CD, TrackRecord is not supported. However, if projects are migrated from 05.00.01 to 05.01.00, and already contain TrackRecord integrations, these **are** supported.

If installing *QA*Director from the *QA*Center CD, both TrackRecord and Request Management are supported.

Managing Changepoint Request Management Integrations

*QA*Director integrates with Changepoint Request Management to submit, edit and delete defects for systems and services. Only one Changepoint Request Management integration can be created. After a Request Management integration has been established, the **New Product Integration** option is not available until the integration is deleted.

Use this process to integrate with Request Management.

To set up a Changepoint Request Management integration with *QA*Director:

1. Open the **Project Information Center.**
2. Select **Defect Management** in the Project panel.
3. Select **Request Management**, and click **Browse**, or click **Tools>Manage Product Integrations**. The Manage Product Integrations dialog box appears.
4. Choose the **Request Management** folder from the **Integrations** list.
5. Click **New Product Integration**.
6. Type an integration name in the **Name** field.
7. Choose **Request Management** from the **Type** list.
8. Type the URL address where **Request Management** resides.
9. Type the following information in the appropriate fields:

- **Server** field: *Changepoint server path*
- **Database** field: *Changepoint database name*
- **User** field: *User name for the Changepoint database.*
- **Password** field: *Password for the Changepoint database.*

- Click the **Save** icon or **Save**.
- Click **OK**.

Third-Party Tools

Use Compuware testing tools or third party testing tools with *QA*Director.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Compuware Testing Tools

If using a Compuware testing tool such as *QA*Run, *QA*Hiperstation, File-AID/CS, or *QA*Load, set up a tool domain for the testing tool. A tool domain is a set of information entered about the testing tool. This information is used to browse, select, create, and execute scripts from within *QA*Director.

Third-Party Testing Tools

If using a third-party testing tool with *QA*Director, create a new integration for the tool. With this integration, it is possible to:

create or select a script to include in a test procedure

run the tool with the selected script

analyze the result to determine if the script passed or failed

communicate the result back to *QA*Director.

Optionally, the integration can help with browsing results of failed tests and editing scripts that were previously inserted.

Add the testing tool to be used first by clicking **File>New>Testing Tool**. The Add Tool Dialog Box appears. Use this to enter information about the tool, such as executable files.

Testing Tool Executables

Note: This feature is not available if using *QA*Director with CARS Workbench.

For the new integration, create or edit the executable files using the Add Testing Tool Dialog Box. Some files are required while others are optional. The following table lists the directory, name, and purpose of each file. **When creating the files, replace XXX with the name of the testing tool.**

Directory and Name	Purpose
*QA*Director\bin\QCXXXRun.exe	**Required:** The program used to run the script.
*QA*Director\bin\QCXXXStart.exe	**Optional:** The program that starts the tool in order to record a script or lets you choose an existing script to use.
*QA*Director\bin\QCXXXEdit.exe	**Optional:** The program used to edit an existing script.

Write these programs in any language. Before beginning, become familiar with the files and file structure of both the testing tool and *QA*Director. Remember that the programs created are called from the test

procedure directory. All directory movement, file movement, or permission changing will occur in this directory. Use the %QC_RESULT_DIR% variable to move files to the corresponding test procedure result directory.

QCXXXRun.exe

This program parses the command line and finds the name of the script file. It then starts the testing tool with this configuration file and runs the tool in "batch" mode. The QCXXXRun.exe program should not exit until the script/tool is finished. Before it exits, the program must communicate to *QA*Director whether the test passed or failed and if it failed, why. The easiest way to indicate a pass is to return a 0 exit status. To cause a script to fail with a specific error message, the program must shell off the following command:

```
qc_exec fail "your failure message here"
```

Instead of using the VB shell command, use the runproc utility defined in this document, as the VB program must wait for the qc_exec program to finish before exiting.

The QCXXXRun.exe program must be able to examine a log file, or otherwise identify the criteria for a passed or failed test, in order to know whether to call qc_exec or not. Often, this is done by looking at a log file or examining the return code of a spawned application.

If the QCXXXRun.exe program needs to store important files generated by the test run, the files can be copied from their original location to the new, special result area. Because the location of this area changes each time the test is run, it is referenced using the environment variable QC_RESULT_DIR. There are also many other environment variables that are available to QCXXXRun.exe, such as the name of the test being run, the name of the script, the purpose of the test, any other attributes defined in the default_template_attrs file, and so on. All can be accessed through environment variables.

QCXXXStart.exe

This program parses the command line to find the name of the script to create and then starts the program needed to create the script.

QCXXXEdit.exe

If a tooledit line is included in the XXX.tool file, the QCXXXEdit.exe program will be called when the user clicks **Edit** on the Test Scripts Manager dialog box. The QCXXXEdit.exe program must open the tool used to configure the script.

Advanced feature

If you call qc_exec fail -file filename "message", the button named **More Info...** on the result tab of a test procedure result will run the filename specified.

For example, if the call to qc_exec was as follows:

```
qc_exec fail -file log.txt "The test failed"
```

When the user clicks **More Info...** in the test procedure result pane, the file "log.txt" runs, and since .txt files are associated with QADirector's internal viewer, that file will be automatically opened in the viewer when double clicked).

Note: The log.txt file must be in the result directory as specified by the %QC_RESULT_DIR% environment variable when the test is run. This is the directory where the test results are stored. If the file specified was "log.doc", then the file would automatically open in Microsoft Word (the associated program in Windows).

Adding a Testing Tool

If using a Compuware testing tool such as *QA*Run, *QA*Hiperstation, File-AID/CS, or *QA*Load, set up a tool domain for the testing tool. A tool domain is a set of information entered about the testing tool. This

information is used to browse, select, create, and execute scripts from within *QA*Director. If using a third-party testing tool with *QA*Director, create a new integration for the tool.

To add a testing tool:

1. Open the **System Administration Center**.
2. From the Testing Tools panel **Action** menu, choose **New**. The Add Tool dialog box appears.
3. Type the name of the testing tool and enter the following executables, commands and statuses:

   Browse information

   Tool Start

   Tool Edit

   Tool Run

   Tool Option

   Tool Exit

   Tool Help

   Tool Start, Tool Edit and Tool Run are executables that require familiarity with the testing tool and QADirector. See Test Tool Executables for a complete discussion.

4. Click **Apply** to apply the changes or click **OK** to save the changes and close the dialog box.

Viewing Third-Party Test Results

Note: This feature is not available if using *QA*Director with CARS Workbench.

To view the results of test executed by third-party tools, create the necessary file associations. For example, for executables that enable an integration with SilkTest, the program named QCSilkView.exe must be associated with the silk-view file extension. The results of the SilkTest script execution creates a file with this extension. View this file in the result details dialog. Launch SilkTest to display the results by double-clicking on the icon associated with the file.

Create the file association within Windows to view the result:

1. Choose **Tools>Folder Options** from **Windows Explorer**.
2. Choose the **File Types** tab.
3. Click **New.**
4. Type the file extension.
5. Click **OK**. This defines the extension, but does not associate it with any program that can read that file type.
6. Click **Change**. A Windows dialog displays. Choose a program to open this file type. If a suitable program is not displayed in the list, browse to the directory containing the appropriate executable.

Creating and Editing Files For a Third Party Integration

Note: This feature is not available if using *QA*Director with CARS Workbench.

For the new integration, create or edit several files using the Add Testing Tool Dialog Box. Some files are required while others are optional. The following table lists the directory, name, and purpose of each file. **When creating the files, replace XXX with the name of the testing tool.**

Directory and Name	Purpose

*QA*Director\bin\QCXXXRun.exe	**Required:** The program used to run the script.
*QA*Director\bin\QCXXXStart.exe	**Optional:** The program that starts the tool in order to record a script or lets you choose an existing script to use.
*QA*Director\bin\QCXXXEdit.exe	**Optional:** The program used to edit an existing script.
*QA*Director\bin\QCXXXView.exe	**Optional:** The program used to view the results of a script run.

Record-Playback Tools

Integrating a record-playback tool with *QA*Director makes it possible to record a test. When scheduling a test to run, *QA*Director automatically opens the record-playback tool and plays back the test that was recorded.

Use multiple record-playback tools and place their scripts anywhere in the suite. One job can contain tests recorded with multiple record-playback and load-testing tools. Tests can be executed automatically without human intervention. Install the tool on the machine where the test is being recorded and played back.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Integrating a defect tracking tool with *QA*Director allows for defects to be tracked within *QA*Director. When scheduled to run the test, *QA*Director automatically starts the defect tracking tool and submits the defect. This makes it possible to view the defect.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To integrate with defect tracking tools:

1. Create an executable.
2. Create custom menu commands.

Creating Custom Menu Commands

Create a custom menu command for specific actions that aren't available in the standard *QA*Director menus.

To create a custom menu command:

1. Click **Tools>Manage Custom Menus**. The Manage Custom Menus dialog box appears.
2. In the **Commands by Category** field, select `Global, Project, or User`.
3. In the **Command name** field, type the command that will appear in the menu.
4. Select whether to add the command to the ***QA*Director Custom** menu or the **Job Results Custom** menu.
5. In the **Run Command** field, type the run command.
6. In the **Directory** field, type the directory path or browse to the desired directory.
7. In the **Status Bar Text** field, type the text to appear on the status line of the *QA*Director main window when the command is selected.
8. In the **Arguments** field, enter the desired argument.

9. Click **New**. The command is added to the **Custom Command** list and will appear in the *QA*Director's **Custom** menu. If this is the first custom command added to *QA*Director, the **Custom** menu will appear for the first time when adding the new command.
10. Click **OK**.

Creating an Executable

Create executable commands to use with submitting, viewing and tracking defects.

Note: This feature is not available if using *QA*Director with CARS Workbench.

To create an executable:

1. Using a development environment such as Visual Basic or Visual C++, create an executable that will accomplish three different functions:

   Starting a defect tracking system

   Submitting a defect

   Viewing a defect
2. Name the executable **XXXX**Integ, where **XXXX** is the abbreviation of the defect tracking tool.
3. Include the /start, /submit, and /view commands in the development code.

/start

Use this command to start the defect tracking tool either by using `CreateProcess` or by any interface exposed by the defect tracking tool.

/submit

When this command is used, a set of environment variables that have the test execution information becomes available to the integration program. The integration program places all of the required information into the tool. The tool must have an interface, such as a COM, exposed so that the integration program can create a defect. Once you submit the bug into the defect tracking tool, you will get a defect ID. Use the COM interface of *QA*Director to set the defect ID attribute.

/View

With this command, the integration program will find the `QA_DEFECT_ID` environment variable and use it in the defect tracking tool's COM interface.

PVCS Tracker

PVCS Tracker is a third-party defect tracking tool. After adding custom menu commands to the *QA*Director menu bar, submit defects to PVCS Tracker, or view defects within PVCS Tracker.

For complete information on PVCS Tracker refer to the PVCS Tracker Help System.

Note: This feature is not available if using *QA*Director with CARS Workbench.

Requirements Management Tools

CaliberRM Integration Add-In

About CaliberRM Integration Add-In

About the GUI

CaliberRM's main window is the starting point for all test management activities accessed through *QA*Center. The CaliberRM **Tools** menu provides an efficient way to access these functions and features via five custom menu choices.

The **QACenter-Login** menu choice launches the **Login** dialog box. Via this dialog box, you can provide user name and password information that can be saved and re-used in future sessions until you end the session via the **QACenter-Logout** menu choice.

The **QACenter - Create** dialog box can be selected via the **Create** menu choice for the initial integration. Future access to projects can have integrations updated via the **Update** menu choice from which the **QACenter - Update** dialog box appears. Once projects have been integrated, test plan traceability reports can be generated for tracking purposes.

At any time after a project has been integrated, you can choose to remove all integration information from CaliberRM and *QA*Center. The **QACenter-Reset** menu choice from the CaliberRM **Tools** menu allows you easy access to this action.

At times, various options on dialog boxes may be disabled. The troubleshooting list may help you determine how to solve the most commonly seen problems with the CaliberRM Integration Add-In.

About the CaliberRM Add-In

The CaliberRM Integration Add-In provides an automated integration solution between *QA*Director and Borland's CaliberRM requirements management solution.

You can access the CaliberRM **Installation and Configuration** Guide via our FrontLine support Web site. FrontLine provides you with fast access to critical information about your *QA*Center product. You can read or download documentation, frequently asked questions, and product fixes, or e-mail your questions or comments.

Using CaliberRM Integration Add-In

Using the CaliberRM Add-In

Using the CaliberRM Add-In involves a basic progression of steps:

1. Logging in to QACenter from CaliberRM.
2. Performing one of these possible actions:
    - Creating Asset Information
    - Updating Asset Information
    - Resetting Integration Information in CaliberRM and QADirector

Logging in to *QA*Center from CaliberRM

Users of the CaliberRM Add-In for *QA*Center will be presented with a **Login** dialog box. Login information will be stored for future use for a configurable amount of session time.

To log in to QACenter from CaliberRM:

1. From the CaliberRM **Tools** menu, choose **QACenter - Login**.
2. In the **CaliberRM** frame, type your CaliberRM user name and password in the appropriate fields.
3. In the **Host** field, type host name of the location where CaliberRM is installed.
4. In the **QADirector** frame, type your *QA*Center user name and password in the appropriate fields.
5. In the **Web Service URL** field, type the host information of the Web Services Server.
6. Click **OK**.

Changing the Default Session Time-Out

Once you log in to *QA*Center using the CaliberRM Integration Add-In, your login information will be saved for a configurable amount of time. The default session time specifies no limit on how long login information will be saved. You can change this default by editing the *QA*Center configuration file. Session time-out will be calculated based on when you closed the integration dialog.

To change the default session time-out

1. Using a standard text editor, open the `QADirector.xml` configuration file. The default location for this file is the `C:\Program Files\Common Files\Compuware\` directory.
2. Within the text, find the line that details the `RMTimeOut` value:

```
<RMIntegUserID value="userID" />
<RMWebServiceURL value="URL" />
<RMIntegLastUsed value="time" />
<RMIntegPassword value="password" />
<RMTimeOut value="" />
```

3. Change the value of the variable listed to the new default session time. Use whole numbers to indicate the number of minutes each session lasts or leave the value blank to have no default session timeout.
4. Save your changes to the configuration file.
5. If you are already logged in to the CaliberRM Integration Add-In, log out and log back in to activate the changed session time.

Correcting the Custom Tools Registry Entries

The CaliberRM Custom Tools registry requires specific entries in order for the requirements management add-in to operate properly. If the registry is altered in any way during or after installation of the add-in, various features of the add-in will appear inoperable.

To ensure Custom Tools registries entries are correct:

1. From the Start menu, select **CaliberRM>Tools>Options>Custom Tools**.
2. Ensure that there are Menu Entries with the following values for:

Menu Entry Name	Command	Argument
QACenter – Login	<install path>\QACCaliberRM.exe	login
QACenter – Logout	<install path>\QACCaliberRM.exe	logout
QACenter – Create	<install path>\QACCaliberRM.exe	create
QACenter – Update	<install path>\QACCaliberRM.exe	update
QACenter – Reset	<install path>\QACCaliberRM.exe	reset

Creating Asset Information

Authorized users of CaliberRM will see the **QACenter - Create** dialog box appear once their login credentials have been verified and a project chosen. Any previous session information will populate the fields by default.

To create asset information:

1. Log in to QACenter from CaliberRM.
2. From the **QACenter Project Association Information** area, select one of the available options to create test assets for a new project or for an existing project.
3. From the first list menu in the **QACenter project association options** area , select the number of levels to use as classes from the requirements hierarchy when integrating projects.
4. In the **New Suite** field, type the name of a new suite if one is being created for the test assets.
5. Click **OK**.

Once various creation options have been chosen, the system checks to make sure that the project specified adheres to standard business rules (for example, the project name must be unique). If a chosen project has already been integrated, a message will be displayed with instructions to perform an update of the project instead of creating a new one.

Generating a Traceability Diagram

To generate traceability information, the CaliberRM integration add-in must be configured. For more information, see "Adding a Vendor Add-In Module to CaliberRM" in *Using QADirector with CaliberRM Installation and Configuration Guide*.

To generate a Traceability Diagram:

1. Ensure the integration has been enabled for the CaliberRM Integration Add-In.
2. Click the **Traceability Diagram** button.

Generating a Traceability Matrix

To generate traceability information, the CaliberRM integration add-in must be configured. For more information, see "Adding a Vendor Add-In Module to CaliberRM" in *Using QADirector with CaliberRM Installation and Configuration Guide*.

To generate a Traceability Matrix:

1. Ensure the integration has been enabled for the CaliberRM Integration Add-In.
2. Click the **Traceability Matrix** button.
3. Click the **Filter** button. The **Traceability Filter** dialog box appears.
4. Click the **QACenter** tab.
5. Select both the **Rows** checkbox and the **Columns** checkbox.
6. Click **OK**.

DOORS Integration Add-In

About DOORS Integration Add-In

About the DOORS Integration Add-In

The DOORS Integration Add-In provides an automated integration solution between *QA*Director and Telelogic's DOORS requirements management solution.

You can access the DOORS **Installation and Configuration Guide** via our FrontLine support Web site. FrontLine provides you with fast access to critical information about your *QA*Center product. You can read or download documentation, frequently asked questions, and product fixes, or e-mail your questions or comments.

About the GUI

DOORS's formal module window is the starting point for all test management activities accessed through *QA*Center. The **DOORSConnect** menu provides an efficient way to access these functions and features via custom menu choices.

The **QACenter - Create** dialog box can be selected via the **Create** menu choice for the initial integration. Future access to projects can have integrations updated via the **Update** menu choice from which the **QACenter - Update** dialog box appears. Once projects have been integrated, you can select the **QACenter Integration>Trace/Report** menu choice to generate reports and view trace based on job results, test type, and test plan.

At any time after a project has been integrated, you can choose to remove all integration information from DOORS and *QA*Center. The **Reset** menu choice from the **DOORSConnect** menu allows you easy access to this action.

The **Login** menu choice launches the **Login** dialog box. Via this dialog box, you can provide user name and password information that can be saved and re-used in future sessions until you end the session via the **Logout** menu choice. The dialog box also allows you to provide a URL specifying the Web services server through which information about *QA*Center projects are obtained.

At times, various options on dialog boxes may be disabled. The troubleshooting list may help you determine how to solve the most commonly seen problems with the DOORS Integration Add-In.

About the DOORSConnect Menu

The DOORSConnect menu provides an efficient way to access software functions and features via custom menu choices.

QACenter Integration>Create allows you first-time access to integration information about a specified project.

QACenter Integration>Update allows you access to information about projects that have already been integrated at some point in the past.

QACenter Integration>Reset allows you to remove all the integration for a project from CaliberRM and *QA*Center.

QACenter Integration>Trace/Report allows you to generate a variety of test plan traceability report formats, giving the user the ability to generate reports and views based on job results, test types, and test plans.

QACenter Integration>Login allows you to access the Login dialog box and set the user name and password that will be saved and reused for future sessions or until your session times out.

QACenter Integration>Logout allows you to log out from your current session.

Using DOORS

Using the DOORS Integration Add-In

Using the DOORS Add-In involves a basic progression of steps:

1. Logging in to QACenter from DOORS.

2. Performing one of these possible actions:
    - Creating Asset Information
    - Updating Asset Information
    - Resetting Integration Information in DOORS and QADirector
    - Generating Traceability Reports

Logging in to *QA*Center from DOORS

Users of the DOORS Add-In for *QA*Center will be presented with a **Login** dialog box. Login information will be stored for future use for a configurable amount of session time.

To log in to QACenter from DOORS:

1. From the **DOORSConnect** menu, choose **QACenter Integration> Login**.
2. In the **User name** field, type the name of the user accessing DOORS.
3. In the **Password** field, type the password for the user name typed in the **User name** field immediately above.
4. In the **Web Service URL** field, type the host information of the Web Services Server.
5. Click **OK**.

Troubleshooting the Add-In Installation, Configuration, or Use

Changing the Default Session Time-Out

Once you log in to *QA*Center using the DOORS Integration Add-In, your login information will be saved for a configurable amount of time. The default session time specifies no limit on how long login information will be saved. You can change this default by editing the *QA*Center configuration file. Session time-out will be calculated based on when you closed the integration dialog.

To change the default session time-out

1. Using a standard text editor, open the `QADirector.xml` configuration file. The default location for this file is the `C:\Program Files\Common Files\Compuware\` directory.
2. Within the text, find the line that details the `RMTimeOut` value:

```
<RMIntegUserID value="userID" />
<RMWebServiceURL value="URL" />
<RMIntegLastUsed value="time" />
<RMIntegPassword value="password" />
<RMTimeOut value="" />
```

3. Change the value of the variable listed to the new default session time. Use whole numbers to indicate the number of minutes each session lasts or leave the value blank to have no default session timeout.
4. Save your changes to the configuration file.
5. If you are already logged in to the DOORS Integration Add-In, log out and log back in to activate the changed session time.

Creating Asset Information

To create asset information:

1. Log in to QACenter from DOORS.
2. Select **QACenter Integration>Create** from the **DOORSConnect** menu. The **QACenter - Create** dialog box appears.

3. From the **QACenter Project Association Information** area, select one of the available options to create test assets for a new project or for an existing project.
4. From the first list menu in the **QACenter project association options** area , select the number of levels to use as classes from the requirements hierarchy when integrating projects.
5. In the **New Suite** field, type the name of a new suite if one is being created for the test assets.
6. Click **OK**.

Once various creation options have been chosen, the system checks to make sure that the project specified adheres to standard business rules (for example, the project name must be unique). If a chosen project has already been integrated, a message will be displayed with instructions to perform an update of the project instead.

Resetting Integration Information in DOORS and QADirector

Previously integrated projects may require removal from DOORS and *QA*Director. To reset the integration information and remove that information from DOORS and *QA*Director, follow the steps detailed below.

To reset integration information in DOORS and *QA*Director:

1. From the **DOORSConnect** menu, choose **QACenter Integration>Reset**. The **Login** dialog box appears if you are not already logged in.
2. Log in to *QA*Center from DOORS if you have not already done so. A message asking you to confirm the reset appears.
3. Click **Yes** to remove all the integration information from DOORS and *QA*Center for the project specified.

Traceability

About Traceability Reports

Three types of traceability reporting options are available:

Filter by Test - Filter test results by test type and job result. Results can be viewed in text-based form (the default) or graphically.

Full Trace View - Produce a full traceability report detailing all requirements, the actual results of the tests on them, and the name of the test used. Results can be viewed in text-based form (the default) or graphically.

Test Plan View - Detail the requirements, test types, actual results, expected results, and the end time related to any tests run. Results can be viewed in text-based form (the default) or graphically.

Generating Traceability for a Full Trace

To generate a full trace:

1. From the start menu, select **DOORSConnect>QACenter Integration>Report/Trace>Full Trace View**. The full trace appears in text form.
2. To view traceability in graphical form instead of the textual default, then from the **Available views** box, select **Graphics**. A color-coded, graphical representation of the results appears.

Generating Traceability Test Plan Details

To generate test plan traceability details:

1. From the start menu, select **DOORSConnect>QACenter Integration>Report/Trace>Test Plan View**. The test plan traceability details appear in text form.

2. To view traceability in graphical form instead of the textual default, then from the **Available views** box, select **Graphics**. A color-coded, graphical representation of the results appears.

RequisitePRO

About RequisitePRO Integration Add-In

About the RequisitePRO Integration Add-In

The RequisitePRO Integration Add-In provides an automated integration solution between *QA*Director and Rationale's RequisitePRO requirements management solution.

You can access the RequisitePRO **Installation and Configuration Guide** via our FrontLine support Web site. FrontLine provides you with fast access to critical information about your *QA*Center product. You can read or download documentation, frequently asked questions, and product fixes, or e-mail your questions or comments.

RequisitePRO's main window is the starting point for all test management activities accessed through *QA*Center. The **Tools** menu provides an efficient way to access these functions and features via five custom menu choices.

The **QACenter>Login** menu choice launches the **Login** dialog box. Via this dialog box, you can provide user name and password information that can be saved and re-used in future sessions until you end the session via the **QACenter>Logout** menu choice.

The **QACenter - Create** dialog box can be selected via the **Create** menu choice for the initial integration. Future access to projects can have integrations updated via the **QACenter - Update** dialog box.

At any time after a project has been integrated, you can choose to remove all integration information from RequisitePRO and *QA*Center. The **QADirector>Reset** menu choice from the **Tools** menu allows you easy access to this action.

At times, various options on dialog boxes may be disabled. The troubleshooting list may help you determine how to solve the most commonly seen problems with the RequisitePRO Integration Add-In.

Using RequisitePRO

Using the RequisitePRO Integration Add-In

Using the RequisitePRO Integration Add-In involves a basic progression of steps:

1. Logging in to QACenter from RequisitePRO.
2. Performing one of these possible actions:
    - Creating Asset Information
    - Updating Asset Information
    - Resetting Integration Information in RequisitePRO and *QA*Director

Logging in to QACenter from RequisitePRO

Users of the RequisitePRO Add-In for *QA*Center will be presented with a **Login** dialog box. Login information will be stored for future use for a configurable amount of session time.

To log in to QACenter from RequisitePRO:

1. From the **Tools** menu, choose **QADirector> Login**.
2. In the **RequisitePRO** frame, type your RequisitePRO user name and password in the appropriate fields.

3. In the **Host** field, type host name of the location where RequisitePRO is installed.
4. In the **QACenter** frame, type your *QA*Center user name and password in the appropriate fields.
5. In the **URL** field, type the host information of the Web Services Server.
6. Click **OK**.

Once logged into *QA*Center, authorized users will see the **QACenter - Create** dialog box appear.

Changing the Default Session Timeout

Once you log in to *QA*Center using the RequisitePRO Integration Add-In, your login information will be saved for a configurable amount of time. The default session time specifies no limit on how long login information will be saved. You can change this default by editing the *QA*Center configuration file. Session time-out will be calculated based on when you closed the integration dialog.

To change the default session time-out

1. Using a standard text editor, open the `QADirector.xml` configuration file. The default location for this file is the `C:\Program Files\Common Files\Compuware\` directory.
2. Within the text, find the line that details the `RMTimeOut` value:

```
<RMIntegUserID value="userID" />
<RMWebServiceURL value="URL" />
<RMIntegLastUsed value="time" />
<RMIntegPassword value="password" />
<RMTimeOut value="" />
```

3. Change the value of the variable listed to the new default session time. Use whole numbers to indicate the number of minutes each session lasts or leave the value blank to have no default session timeout.
4. Save your changes to the configuration file.
5. If you are already logged in to the RequisiteProIntegration Add-In, log out and log back in to activate the changed session time.

Creating Asset Information

Authorized users of RequisitePRO will see the **QACenter - Create** dialog box appear once their login credentials have been verified. Any previous session information will populate the fields by default.

To create asset information:

1. Log in to QACenter from RequistiePRO. The **QACenter - Create** dialog box appears.
2. From the **QACenter Project Association Information** area, select one of the available options to create test assets for a new project or for an existing project.
3. From the first list menu in the **QACenter project association options** area , select the number of levels to use as classes from the requirements hierarchy when integrating projects.
4. In the **New Suite** field, type the name of a new suite if one is being created for the test assets.
5. Click **OK**.

Once various creation options have been chosen, the system checks to make sure that the project specified adheres to standard business rules (for example, the project name must be unique). If a chosen project has already been integrated, a message will be displayed with instructions to perform an update of the project instead.

About the Requirements Management Add-In Server Component

Compuware's requirements management integration add-In components provide an automated integration solution between *QA*Director and Telelogic's DOORS, Rational's RequisitePro, and Borland's CaliberRM requirements management solutions.

As part of each of these solutions, a server component requires installation and configuration to enable the add-in components to operate properly.

About the *QA*Center Database Configuration for Requirements Management Dialog Box

The ***QA*Center Database Configuration for Requirements Management** dialog box allows users to configure the server component of the requirements management add-ins for *QA*Center. The controls on the dialog box are organized in two major areas: the Database Configuration area and the command buttons.

Database Configuration

SQL Server/MSDE: Select to choose SQL Server/MSDE as the database to be used with the requirements management add-ins.

Oracle: Select to choose Oracle as the database to be used with the requirements management add-ins.

Server name: Type the name of the database server.

Database name: Type the name of the database.

DSN/Service: Blank by default.

User ID: Type the ID of the database user used to connect to the specified database

Password: Type the password of the database user ID used to connect to the specified database

QACenter Integration URL: The URL for the QACenter integration is automatically generated.

Test: Click once all fields are appropriately filled with information to test the database integration.

Command Buttons

Help: Click to launch the help application.

OK: Click to accept the entries and choices.

Cancel: Click to cancel the options selections.

Glossary

Glossary

Ancestor
Test in the test suite hierarchy from the root class to the parent of a specified test.

Associated File
File required by a test in order to execute properly.

Attribute
Name-value pair that *QA*Director uses to store information about test classes and test procedures. The values of some attributes appear in corresponding fields in the GUI so you can assign and edit their values.

Automated Script
Script captured using a record/playback testing tool, or a user defined script.

Child
Test asset that is an immediate descendant of a specified parent class.

Class Templates
Classes, procedures, and scripts arranged in a permanent structure that can be used across multiple suites. When the class template is changed, it is changed everywhere.

Cleanup Command
One of four kinds of rules you can define in *QA*Director. Cleanup commands perform tasks needed after tests execute, such as copying files or terminating processes. Also known as cleanup rule.

Descendant
Test in the test suite hierarchy from the children of a specified test class down to their leaf nodes (test procedures).

Detach
Occurs when removing a test reference from the suite to the Test Library.

Environment Variable Rule
One of four kinds of rules you can define in *QA*Director. Environment variable rules behave like shell environment variables. *QA*Director evaluates environment variable rules before test execution.

Event
An occurrence or happening of significance to a task or program, such as the completion of an asynchronous input/output operation.

Failed
Outcome if any of the following occur: actual output was not the expected output, the script exits with an unexpected status, the script or a pass/fail rule issues qc_exec fail, or a pass/fail rule failed or exited with a non-zero status.

Filter
Searches the database and displays the information requested. A filter is used to refine information that is displayed on the screen in whatever area of QADirector you are working.

Global
Setting that makes item available to all users, or projects depending on what is being designated as global. Global scripts are available to all projects. Global tests are available to all users. Global scripts can not be changed back to normal library scripts.

Global Command
Command that is invoked within *QA*Director rules and scripts.

Job
Submission for execution of a job description.

Job Description
Settings to specify how a group of tests is to run.

Locked
State of test while changes are being performed to an existing class or procedure.

Manual Test
Test that requires interaction from the tester, whether it be answering questions or performing some action. It is possible to use manual tests when automated testing is impractical or not yet implemented.

Not Executed
Test procedure status that is assumed when a setup rule exits with a nonzero status or a setup rule issues qc_exec fail telling *QA*Director to mark the command as failed and the test procedure not executed. A test procedure that is not executed is distinguished from a test procedure that ran and either Passed or Failed.

Outcome
One of the following states of an executed test procedure: Passed, Failed, or Not Executed.

Parallel Execution
Children of a test class run at the same time on an available machine in the network.

Parent
Direct ancestor class of a test class or test procedure.

Pass/Fail Command
One of four kinds of rules that can be defined in *QA*Director. A pass/fail command checks a test procedure during test execution for the criteria specified. Also known as pass/fail rule.

Passed
Test procedure status that is assumed when no condition occurred during test execution to cause the outcome to be Failed or Not Executed.

Project
Highest level in the test hierarchy. Contains requirements, test suites, defects, and the test library.

QA 3270 Emulator Service
Utility program that surveys communication between 3270 mainframe emulator and Compuware products.

Referencing
Occurs when copying a test from the Test Library to a suite, or if creating a test in a suite and attaching it to the Test Library. The suite does not actually contain the test. Instead, a marker resides in the suite and references the actual test in Test Library. The suite always references the Test Library unless a test within a suite has been detached from the Test Library.

Risk-Based Testing
More complete testing on those components of a system that present the highest risk of failure. Users must determine what those components are within a system and what the consequences of those components failing.

Root Class
Highest level class in the test suite hierarchy, from which all other test classes and test procedures descend. A root class does not have siblings.

Rule
Command that prepares for test execution, collects information during test execution, determines whether a test procedure failed, or cleans up afterward. *QA*Director has four types of rules: environment variables, setup commands, pass/fail commands, and cleanup commands.

Script
A series of instructions, manual or automated, in a test procedure that runs a test of an application.

Sequential Execution
Occurs when sibling tests run one at a time in the order they appear in the test suite.

Setup Command
One of four kinds of rules you can define in *QA*Director. Setup commands prepare for test execution. Also known as setup rule.

Test
Refers to either a test class or a test procedure. In *QA*Director, tests are combinations of test classes, test procedures, or both. The fundamental difference between a test class and a test procedure is that a test procedure performs a test of an application and reports whether the test passed or failed. A test class does not test an application and cannot report on the outcome. A test class is simply a grouping of test procedures and other test classes.

Test Assets
Test classes, test scripts and test procedures.

Test Class
Logical grouping of test procedures and other test classes in the test suite. Determine the logic behind each test class grouping. For example, create test classes to group tests by features, functions, or structure of an application, or by any other useful criteria. A test suite can have as many levels of test classes as is necessary. Test classes can have as many descendants and siblings as needed (except for the root class which has no siblings).

Test Execution Server (TES)
Allows jobs to run on a computer. When requested by the Test Management Server, the TES starts the test driver, which subsequently starts the test engine. The TES also stores information about who is permitted to execute jobs on the computer, and whether the execution is remote or local.

Test Library
Repository containing tests classes, test procedures, and test scripts.

Test Management Server (TMS)
Reads job submissions from the database, manages jobs, tracks the machines available for test execution, and manages the licenses used for manual test Web execution. It also reports on the status of jobs in progress.

Test Procedure
Performs the actual test and reports whether the test passed or failed. Thus, every test procedure must include at least one script, which contains the instructions for performing the test. Test procedures can be placed directly below the root class in the test suite hierarchy, or they can be grouped into sub-classes. Test procedures can have as many siblings as necessary but they cannot have children.

Is the smallest unit of the test suite that is run. A script cannot run on its own; instead, the procedure containing the script must be run.

Test Suite
Group of related tests in a multi-level hierarchy. A test suite might include all the tests necessary to test an entire application, or it might include the tests for only a single component of the application. Within the test suite, test procedures hold the actual scripts. Organize the test procedures into various test classes within the suite.

After a test suite is added, *QA*Director automatically creates a single root class and gives the root class the same name as the new suite. The root class is at the base of every test suite structure, from which all the other tests in the hierarchy descend. The root class cannot have siblings. Each node in a test suite is either a test procedure or a test class.

Testing Tool
A program used to test, analyze and create reports for other programs.

Timeout
Time in minutes after a script or rule starts executing that *QA*Director will wait before killing the process.

Tool Domain
Set of information about the testing tool that allows a user to browse, select, and create scripts from within *QA*Director. Necessary for use with a Compuware automated-testing tool such as *QA*Run, *QA*Hiperstation, *QA*Load, or File-AID/CS. It is necessary to set up at least one *tool domain* for each tool.

Unexpected Outcome
Occurs when a test does one of the following:

(a) *Passed* when the expected outcome was *Failed* or *Not Executed*. (b) *Failed* when the expected outcome was *Passed* or *Not Executed*. (c) *Not Executed* when the expected outcome was *Passed* or *Failed*.

VTAM
Virtual Telecommunications Access Method. A data communications access method compatible with IBM's Systems Network Architecture.

Troubleshooting

Running Tests

How do I designate which version of the executable to test?

1. Click the **Rules** tab of the **Test Class** dialog box of the root class.
2. Click the **Add** button to open the **Add/Edit Rule** dialog box.
3. Select **Environment variable**.
4. Enter the following in the **Name** field:

   path

5. Enter a value in the **Value** field that is similar to the following example:

   c:\TeamBuild\bin;%path%

Where c:\TeamBuild\bin is the directory containing the version of the executable you want to test.

6. Click **OK** to close the **Add/Edit Rule** dialog box.
7. Select **File>Save** to save the change.

Note: When you or a team member executes any tests in the suite of the previous example, *QA*Director sets the path environment variable rule. The path environment variable rule prepends the path to the application to the existing path. The test suite tests the application in the specified path regardless of who executes tests. The path remains the same until the next time you change the path environment variable rule.

You can place an environment variable rule in any test. For example, if a suite tests two applications, you can define the path environment variable rule in two sibling classes. The procedures of one class would test one of the applications, and the procedures of the other class would test the other application.

How do I make a process available whenever tests are run?

Add a setup command to start the process before any tests run:

1. Click the **Rules** tab on the Properties dialog box of the desired test.
2. Click the **Add** button to open the **Add/Edit Rule** dialog box.
3. Choose **Setup**.
4. Enter the full path to the process in the **Command** field. For example:

   c:\TeamBuild\bin\server

5. Click **OK** to close the **Add/Edit Rule** dialog box.
6. Select **File>Save** to save the change.

Note: In the example, you enter the full path to the process that you want available to your tests. Or, you could enter only the command name in the setup rule if you also apply an environment variable rule that sets the path to the directory containing the command.

When you or a team member executes any tests in the suite, *QA*Director starts the process. The process is available when the tests execute.

You can add a cleanup rule to any test. For example, if some tests in the suite cover clients as well as the server, you can define a setup command to start a client in the class that contains the procedures that test that client.

To start a process in the background, at a command prompt, enter:

```
Start “(windowtitle)” <command>
```

How do I stop a process after execution?

Add a cleanup rule to the desired test that shuts down the process after all the tests execute:

1. Click the **Rules** tab of the **Properties** dialog box.
2. Click the **Add** button to open the **Add/Edit Rule** dialog box.
3. Select **Cleanup**.
4. Enter the full path to the command that terminates the process in the **Command** field. For example:

   c:\TeamBuild\bin\ServerShutDown

5. Click **OK** to close the **Add/Edit Rule** dialog box.
6. Choose **File>Save** to save the change.

Note: In the example, you enter the full path to the command that shuts down the process. Alternatively, you could enter only the command name in the cleanup rule if you also apply an environment variable rule that sets the path to the directory containing the command.

When you or a team member executes any tests in the suite, *QA*Director automatically shuts down the process after tests run.

You can place a cleanup rule in any test. For example, if only one class in a suite requires a process, you can define a setup rule in the class to start the process and a cleanup rule in the same class to shut down the process.

Creating and Sharing Tests

Why can't I see my co-worker's test suites in *QA*Director?

There are several possible reasons:

You may not be connected to the same database that your co-worker used to create the test. Use the *QA*Director Administrator module to connect to the appropriate database.

If you can view the test suite in the list but are not allowed to open it, the test suite owner may not have given you permission to view the suite. See Assigning User Passwords and Permissions.

Running Diagnostics

About Running Diagnostics

If *QA*Director is not performing optimally, it is possible to test its operation. Compuware's Customer Support can use the diagnostic results to help troubleshoot issues. The diagnostic tool tests basic *QA*Director operations, such as:

Windows version

Winsock information

Machine information

Current user information

*QA*Director installation information

Existing execution programs

Test Management Server availability on current machine

User privileges on current machine

Accessing the Diagnostic Tool

To access the diagnostic tool:

From the **Help** menu in *QA*Director, choose **Diagnostic** or from the Windows **Start** menu, choose **Programs>Compuware>*QA*Director>Diagnostic**. The Diagnostic dialog box appears.

Running a General Diagnostic

The general diagnostic test reviews the Windows and WinSocket versions, verifies the host name, address, and current user, and tests the application directories, *QA*Director executables, test engine executables, and create processes. It also tests the **Test Management Server** availability, installation location, and service status.

To run a general diagnostic test:

1. Click the **General** tab.
2. Select the **Current Machine** check box or the **Machine** check box and type the machine name in the **Machine** field.

Note: Administrative permissions are required to check the **Test Management Service** availability on other machines. Also, use the same user name and password as on the current machine.

3. Click **Start Diagnostics**. *QA*Director runs the general diagnostic test and displays the results in the **Results** field.
4. For more results details, click **Show Details**. *QA*Director displays the details for all results.

Saving Diagnostic Results

Save the results of the diagnostic test.

To save any of the preceding diagnostic test results to a text file:

1. Click **Save Results**. The Save As dialog box appears.
2. Choose a directory, name, and file type, then click **Save**.
3. Click **Exit** to close the Diagnostics dialog

Testing the Connection

The connection checks the communication between two machines. One machine acts as a "listener" and the other machine becomes the "client."

To test the connection:

1. Specify port numbers between the machines.
2. Click the **Connection** tab.
3. Designate a listener machine:
    - Select the **Listener** radio button.
    - Type the port number in the **Connecting Port** field.
4. Click **Start**. The listener machine waits for the client to connect. After accepting the request from the client, the listener machine adds the connection information and results to the **Status** field.
5. Set up a client machine:
    - From the client machine, select the **Client for** radio button.
    - Type the listener machine name in the **Client for** field.
    - Type the port number in the **Connecting Port** field.
    - Click **Start** to connect to the listener. The connection status and speed display in the **Status** field.

Checking Port Availability

From the **Port Scanner** tab, check port availability on specified machines.

To test the port:

1. Click the **Port Scanner** tab.
2. Type the machine name in the **Host** field.
3. Select a port range from the **Port Range** area, or type a custom range in the **From** and **To** fields.
4. Click **Scan**. The results display in the **Results** field.

Running a Ping Test

Use the ping test to check for the presence of another machine.

To run the ping test:

1. Click the **Ping Test** tab.
2. Type the machine name in the **Host** field.
3. Choose a duration from the **Ping Every** field.
4. Choose the number of times to ping from the **Time(s)** field.
5. Click **Start**. The results display in the **Results** field.

Starting and Running QADirector

Why can't I connect to the Test Management Server?

Your machine may not be set up properly on the network. This happens when a machine was originally configured to use Dynamic Host Configuration Protocol (DHCP), but now is run off the network. Supplying a static IP address or reconnecting to the network are two workarounds for this problem.

Why can't I start the Test Management Server?

To help Technical Support resolve the problem, complete the following procedure.

1. From a DOS command line, enter: `cd c:\program files\compuware\QADirector\x86-win32\bin`
2. Enter: `set QC_DEBUG=1`
3. Run: `qc_tmsrv -console`
4. Send the resulting file named `c:\temp\tmsrv_msgs.<yournamehere>` to *QA*Director Technical Support at hotline_support@compuware.com.

Using QADirector With Other Tools

Why can I run the QARun tests on one computer but not another?

The most likely reason is that the *QA*Run tool domain uses a local path to point to the *QA*Run database. This local path is not accessible from all machines. For example, on machine A, the path to the *QA*Run database may be c:*QA*Run*QA*Run.mdb. Because this path is not valid on machine B, the test will not run there. The *QA*Run database must be referenced with a shared drive letter that is the same on all machines. Details

Customer Support

Compuware Customer Support

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. Please obtain the following information before calling Compuware's 24-hour Product Support Hotline:

- The version of *QA*Director you are using, which is displayed in the **About *QA*Director** dialog box of *QA*Director components
- The version of each operating system(s) in which each product component is installed
- The place in the *QA*Director software where the problem occurred and the steps taken before the problem occurred
- The exact *QA*Director error message, if any
- System error messages, if any.

Contact information for Compuware Customer Support is as follows:

Compuware North America
Compuware Corporation
One Campus Martius
Detroit, Michigan 48226
USA
Telephone: (313)227-7300
Product Support Hotline: (800) 538-7822

Compuware United Kingdom
Compuware Ltd.
163 Bath Road
Slough, Berkshire SL1 4AA
United Kingdom
Telephone: +44 1753 774 000
Fax: +44 1753 774 200

Compuware International Operations
Compuware Corporation
One Campus Martius
Detroit, Michigan 48226
USA
Telephone: (313)227-7300

FrontLine Support Web Site

You can access online customer support for Compuware products via our FrontLine support Web site at http://frontline.compuware.com. FrontLine provides you with fast access to critical information about your *QA*Center product. You can read or download documentation, frequently asked questions, and product fixes, or e-mail your questions or comments.

Index