This tutorial demonstrates how to use IBM Rational Robot to record, edit, and execute functional regression tests. It will give you a jump-start on using Rational Robot by showing the basic steps of installing the product, creating the necessary datastores, launching the tool, using it to record manual interactions with a sample application, editing the resulting test script, and playing it back against another build of the application to uncover a functional regression.

Section 1. Before you start

About this tutorial

This tutorial walks you through the basics of installing, configuring, and using Rational® Robot to record, edit, and execute functional regression tests. This tutorial does not attempt to cover all the varied aspects and capabilities of the Rational Robot package. It gives you a jump-start on using Rational Robot by showing the basic steps of installing the product, creating the necessary datastores, launching the tool, using it to record manual interactions with a sample application, editing the resulting test script, and playing it back against another build of the application to uncover a functional regression.
Who should take this tutorial?

This tutorial is designed for software testers who are first-time users of IBM® Rational Robot. The tutorial will help you create, execute, and edit your first Rational Robot regression testing script. Along the way, you will be introduced to the challenges of regression testing and learn about the solutions Rational Robot offers to address these challenges.

Prerequisites

To complete the steps and exercises in this tutorial, you will need to download IBM Rational Robot.

Rational license enforcement

Rational Robot uses Rational Common Licensing. This means that once you install Rational Robot, you will need a temporary license key. Temporary license keys can be acquired through your Rational Account Representative at 1-800-728-1212. Your Rational Account Representative will send you either a key code string or e-mail you a .upd file. Keep these handy -- you will need them in Installing the temporary license key.

About the author

For technical questions or comments about the content of this tutorial, contact Dennis at dennis.schultz@us.ibm.com.

Section 2. Why Rational Robot?

What is Rational Robot?

If you are working through this tutorial, you probably already know what Rational Robot is. Following is the official description from the Rational Robot Users Guide:
Rational Robot is a complete set of components for automating the testing of Microsoft Windows client/server and Internet applications running under Windows NT 4.0, Windows XP, Windows 2000, Windows 98, and Windows Me.

The main component of Robot lets you start recording tests in as few as two mouse clicks. After recording, Robot plays back the tests in a fraction of the time it would take to repeat the actions manually.

Other components of Robot are:

- **Rational Administrator** -- Create and manage Rational projects to store your testing information.
- **Rational TestManager** -- Review and analyze test results.
- **Object Properties, Text, Grid, and Image Comparators** -- View and analyze the results of verification point playback.
- **Rational SiteCheck** -- Manage Internet and intranet Web sites.

It can be a little confusing. Both the record/playback engine component and the entire package have the same name -- Robot. In this tutorial, I will try to refer to the record/playback engine component as simply "Robot" and use the term "Robot Package" for the entire offering.

During the course of this tutorial, I will touch on almost all of these components, although you will concentrate most of your time on the main component of Rational Robot.

Rational Robot is an **automated functional regression** testing tool. Let's break that down a little...

**Rational Robot is a functional testing tool**

Functional testing is not concerned with how quickly or slowly the application does its job. That would be performance testing.

Functional testing is not directly concerned with how robust the application is. Functional testing doesn't really care if there are memory leaks unless, of course, the lack of a robust implementation means the user cannot use the application to do what it needs to do.
Functional testing is concerned with verifying an application operates as it was intended. There are various ways to gauge how well an application functions such as how well it conforms to specifications or requirements. In the simplest terms, you can ask "Does the application do what the user needs it to do?"

Rational Robot is a regression testing tool

A regression is when something that worked in one build of an application suddenly does not work in a subsequent build. Often, this occurs when there seemingly should be no connection between the change made in the new build and the existing functionality. Regression testing involves running a suite of functional tests against build after build of an application to verify everything that was working continues to work.

Rational Robot is an automated testing tool

Functional regression testing can be done through manual testing. The problem is, as a project evolves, that functional test suite becomes longer and longer. At some point, it can take more time to run the test suite manually than you actually have between builds!

When this happens, test teams are faced with tough choices. They have to find a way to decide which tests are "less critical" and can be omitted on some or all build tests. Now the door has been opened for regressions to slip into those untested capabilities of the application.

Rational Robot is an *automated* functional regression tool. This means you use a computer to execute the tests much faster than you could manually. The key to doing this is to have a way to capture interactions with an application and make those interactions available to be repeated later.

---

**Section 3. Get started**

The demo application
There are many factors at work when you attempt to record and play back automated test scripts against a software application, including the language used to implement the application (for example, Visual Basic, Java, HTML, PowerBuilder), the environment in which it runs (including application data, Windows settings), and the skills the test team possesses (such as programming knowledge, business understanding of the application).

In an attempt to normalize all these factors, in this tutorial you will use a simple Visual Basic demo application called Classics Online, included with Rational Robot as Application Under Test (AUT). Once you have explored some of Rational Robot's capabilities with Classics Online, you will be better prepared to apply those capabilities in a more "real world" setting -- your setting.

Installing Rational Robot

Now that you have downloaded the Rational Robot installation program and have your evaluation license keys, you are ready to install it.

1. Run the executable you downloaded in Prerequisites.
2. During installation, you will need to provide the following information:
   1. Location to save files -- This is a temporary location where you will expand the installation files. You need at least 370MB of disk space. You can delete these expanded files once installation is complete.
   2. The Product to be installed -- Choose Rational Robot. The license key server is required only for floating license keys. You will be using node-locked license keys for this evaluation.
   3. Deployment Method -- Choose Desktop installation.
   4. For all other settings, take the defaults.

Installing the temporary license key

You should see the Rational License Key Administrator window on your screen. As was mentioned earlier, Rational Robot utilizes the Rational Common Licensing enforcement mechanism. You should have either a key code string or a .upd file furnished by your Rational Account Representative. If you have a .upd file, skip to
Installing the temporary license key from a .upd file of this tutorial.

If you have a text license string, do the following:

1. Open the Rational License Key Administrator application if it is not already open.
2. When asked to choose a method of adding a license key, select Enter a Temporary or Evaluation License Key.
3. The license type will be a Node-Locked license key.
4. On the License Key Information page, select Rational Robot as the product.
5. Use the information from your Rational Account Representative to enter the expiration date and license key value.

Installing the temporary license key from a .upd file

If you installed the license key from a text string in the previous section you can skip this section. If you have a .upd file containing your temporary license key, do the following:

1. Open the Rational License Key Administrator application if it is not already open.
2. When asked to choose a method of adding a license key, select Import a Rational License Key File.
3. Browse to the .upd file on your hard drive.
4. Click OK on the confirmation window and close the License Key Administrator.

Section 4. The Rational Administrator Project
What is a Project?

Rational software develops tools that support all phases of software development. The Rational Administrator Project is a logical grouping of multiple datastores that includes requirements management projects, defect management databases, visual models, and test datastores. This logical collection is managed through a component of Rational Robot called the Rational Administrator. For this tutorial, you will only be interested in the test datastore portion of the Rational Administrator Project, but your real test team can use an integrated set of datastores that allows you to trace your test case planning back to requirements or visual model components or submit defects into your ClearQuest defect tracking database directly from your test logs. These integrations are beyond the scope of this tutorial, but to learn more, check out another developerWorks tutorial *Managing application testing: Using Rational TestManager to optimize your tests*.

Creating the Rational Administrator Project

1. To create the Rational Administrator Project, start the Rational Administrator component of Rational Robot by selecting **Start > Programs > Rational Software > Rational Administrator**.

2. Choose **File > New Project**. Enter a simple project name and a path where you would like to store this project. To keep it simple, use the name tutorial and the path `C:\rational\tutorial`.

3. A warning window opens. This window is warning you that if you create a project on your local C: drive, no one else on your team will be able to share it with you. When you create a project for use in a real team environment, you should use a network share to store your projects. For this tutorial, the C: drive will be fine. Click **OK**.

4. Rational Administrator Projects can be password protected. For this tutorial, do not set a password and just click **Next**.

5. Ensure that **Configure Project Now** is checked on the Summary page.

6. Configure the test datastore that will be part of the Rational Administrator Project you just created. On the Configure Project screen, click **Create...** in the Test Assets group.
7. Test datastores can use either Sybase SQL Anywhere or Microsoft Access as the database engine. Rational recommends using SQL Anywhere for any project being shared by two or more testers concurrently. For this tutorial, select **Microsoft Access** on the Create Test Datastore (Page 1 of 4) window.

8. Accept the defaults on the remaining pages. When the datastore has been created, you can close all windows and exit the Rational Administrator.

---

**Section 5. The demo application**

**Setting up the demo application**

To set up the demo application, choose **Start > Programs > Rational Software > Rational Test > Setup Rational Test Samples**. Click **Next** to install the default applications. When setup is complete, click **Finish**.

**A quick tour of Classics Online**

Classics Online is a simple Visual Basic application that simulates a classical music shopping application. The main screen consists of a tree control where the buyer can select the CD to purchase. Details and an image of the cover of the selected item are displayed in the lower portion of the screen.
The press here to order button takes you to an order screen where a credit card number and expiration date must be entered to place the order. Note that this is purely a simulated purchasing system. A valid credit card is not required. In fact, any text you enter in the credit card number and expiration fields will suffice.
Section 6. Rational Robot

Starting Rational Robot

To start Rational Robot, choose Start > Programs > Rational Software > Rational Robot. A login window appears. You must log into a Rational Administrator Project.
to continue. The project you just created should be selected. If not, select it from the drop down list. Projects are created with an Admin user which has a blank password. In a real project, you would, of course, want to create usernames for each tester and have them set their own passwords. For this tutorial, just click OK to use the Admin username.

The main screen of Rational Robot opens.

![Rational Robot Interface](image)

Until you have a script recorded, there isn't much to see in the Rational Robot interface. The interface is described after you record your first test script.

---

**Section 7. Record a test script**
Starting the recorder

You are now ready to record your interactions with the application into a Rational Robot script. Once you begin recording, Rational Robot records everything you do on your computer -- all clicks, drags, and typing. Keep this in mind when recording scripts. If you stray off to other tasks such as reading e-mail, checking the score of the big game on the Web, instant messaging your friends, to name a few, Rational Robot records it all! It is easy enough to edit those unwanted things out of your script later, but it is much easier to just avoid it in the first place. You might want to print out the entire "Recording a test script" section of this tutorial before continuing so you don't have to switch back and forth to your browser for instructions.

To begin recording:

1. Click **Record GUI Script** on the Rational Robot toolbar.

   Note that there are two record buttons, one for GUI script recording and one for VU script recording. Be sure to use the GUI button.

2. Provide a name for the script. Enter **OrderItem** for the script name and click **OK**.
What just happened?!? When you start recording, Rational Robot minimizes itself to get out of your way. A GUI Record toolbar appears.

The GUI Record toolbar always stays on top of other windows. This is how you will know Rational Robot is in record mode. Remember, Rational Robot is recording all your interactions with any objects on the screen now. If you do need to do something on your machine during a recording session, use the Pause button on the GUI Record toolbars.

Rational Robot will ignore any actions you take while recording is paused. You can resume recording by clicking the Pause button again.

**Invoking the Application Under Test**

The first thing to do is start the Application Under Test (AUT):

1. Click **Display GUI Insert Toolbar** on the GUI Record toolbar.

2. The GUI Insert toolbar has many tools, which you will use while recording test scripts. You will have a chance to look at several of these in a moment. For now, click **Start Application**.
3. In the Start Application window, use the Browse button or enter "C:\Program Files\Rational\Rational Test\Sample Applications\Classics Online\ClassicsA.exe" as the application name, then click OK.

Placing an order with Classics Online

The login window for Classics Online appears.
The Classics Online application has no real logins. Click OK to continue.

Now you are at the main screen of Classics Online. Double-click Beethoven in the tree control, then click Symphony No. 9. Now click press here to order to place an order for this CD.
Once you have clicked the press here to order button, the Make An Order window appears.

**Verification points**

So far, Rational Robot has recorded navigation through the application. The resulting script is useful to verify the application can be navigated, but at this point, it really isn't a test because you haven't verified anything.

The CD, quantity, price, and purchaser's details are automatically filled in for you on this screen. This may be the application functionality you wish to verify. Because you logged in as Trent Culpito, let's verify that is the name that appears on this form.
On the GUI Record toolbar, click Display GUI Insert again. Notice some of the tools on the GUI Insert toolbar. There are many tools you can use to insert various types of verification points (VPs). Let's take a look at some of the available verification point types:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Symbol" /></td>
<td>Alphanumeric</td>
<td>Verifies alphanumeric data. Used for edit boxes, pushbuttons, labels, text fields, and so forth.</td>
</tr>
<tr>
<td><img src="image" alt="Symbol" /></td>
<td>Clipboard</td>
<td>Verifies the contents of the Windows clipboard.</td>
</tr>
<tr>
<td><img src="image" alt="Symbol" /></td>
<td>Menu</td>
<td>Verifies the menu values and optionally their state (enabled or disabled) of a window.</td>
</tr>
<tr>
<td><img src="image" alt="Symbol" /></td>
<td>Object Data</td>
<td>Tests data content of objects.</td>
</tr>
<tr>
<td><img src="image" alt="Symbol" /></td>
<td>Object Properties</td>
<td>Tests object attributes such as color, font, position.</td>
</tr>
<tr>
<td><img src="image" alt="Symbol" /></td>
<td>Region Image</td>
<td>Graphically compares an area of the screen you specify.</td>
</tr>
<tr>
<td><img src="image" alt="Symbol" /></td>
<td>Window Existence</td>
<td>Tests to see if a particular window does or does not exist on the screen.</td>
</tr>
<tr>
<td><img src="image" alt="Symbol" /></td>
<td>Window Image</td>
<td>Graphically compares an entire window such as a window box.</td>
</tr>
</tbody>
</table>

This set of verification point types provides a broad capability for verifying the state and data of an application under test. In all cases, the basic premise is to capture the current state of the application as a baseline. This baseline is compared to the actual state of the application later during playback. If they are the same, the VP passes. If they are different, it fails.

To verify the name on the form as well as the font used, the best choice is an Object Properties Verification Point.

1. Click the Object Properties VP button on the GUI Insert toolbar. The Verification Point Name window enables you to give the verification point a name, and set some additional information about what is expected.

2. Enter Customer Name in the Name field and click OK.
3. You must designate the object you want to test. There are two ways to do this: by dragging a pointer to the object, or by browsing for the object in a list. Because Rational Robot can test even hidden objects, browsing is sometimes the only way to designate the object. Most often, pointing to the object is easier.
4. Click the **Object Finder tool** and drag it over the Customer Name text field, then release the mouse button. Notice that a tool tip appears as you drag the Object Finder tool over the screen. This tool tip gives you information about the object you are hovering over and how it will be recognized by Rational Robot.

![Select Object dialog box](image)

5. When the Select Object window returns, click **OK**. Remember you are inserting an object **properties** verification point. The window that appears next shows you all the properties that you *could* verify with this VP. Notice how Rational Robot can see many, many properties of the object. It isn't always a good idea to test for every property just because you can. Testing unimportant properties can make your tests take longer to execute and can make the tests more likely to
fail when something changes that you really don't care about. For example, if you choose to verify the background color of a text field, that VP might fail when run on a PC with different desktop settings. This is probably not a real failure of the application, but rather a situation of checking too many properties. For this tutorial, verify the text of the name, the font, and that the text is bold.

6. Click the Edit List button on the Object Properties Verification Point window. This opens a window captioned Edit Property List: Thunder TextBox. Use the buttons in the center to move all properties to the Available list, and then add FontBold, FontName, and Text properties back to the Selected list.

7. Click Save as default. This tells Rational Robot that you want to check these three properties each time you create an Object Properties Verification Point for a TextBox. Of course, you can change which properties you check on each individual object, but this provides a way to have each VP default to the most common set of properties you test for that type of control.

8. Click OK to set the list.
9. Back on the Object Properties Verification Point window, click OK to complete the definition of the verification point.

Completing the order

All that is left is to enter a credit card number and expiration date. In this sample application, you do not need a valid credit card number or date. The fields must be non-blank, however. Click in the Card Number field and type any sequence of ten digits as a simulated credit card number. Now click in the Expiration Date field and enter any expiration date. Click Place Order.
Classics Online provides an Order Confirmation window. Click OK to confirm your order.

**Verifying the order was placed**

Classics Online provides an interface that enables a customer to check existing orders. For a tester, this interface can be very useful to confirm that the order actually made it to the database correctly.

1. From the Classics Online menu, choose **Order > View Existing Order Status**. The grid control in the resulting window lists all the orders in the system for this user.

   ![View Existing Orders](image)

   If you have already run through this scenario with Classics Online, you might have additional orders in the system that are not shown in the preceding screen shot. Those orders should not impact the ability to complete this tutorial.

2. Use an Object Data Verification Point to verify that the order has gone into the database properly. Click the Display GUI Insert toolbar button on the GUI Record toolbar.

3. Click the Object Data button on the GUI Insert toolbar.
4. Name this VP Existing Orders.

5. Click OK.

6. As before, drag the Object Finder tool over the data table on the View Existing Orders window and release the mouse button.

7. Click OK on the Select Object window.

8. Different GUI controls offer various ways to capture the data in them. For this test, capture All Data in the grid. Click OK.
9. Finally, you see the Object Data Verification window.
The Object Data Verification window provides you with a number of options for how you want your verification point to be created. For example, in the Verification method area, you can specify if the test should be case sensitive. In other words, **Order Initiated** and **ORDER INITIATED** would not match and, therefore, fail the verification point.

**Select the range to test** allows you to pick the parts of all the data that you care about. You may only want to test one column of data in the table or even just a single cell.
Identification method area allows further refinement. Let's say you selected a single row out of several rows in the Select the range to test area above. Do you want to just make sure that row appears somewhere in the table, or do you want to make sure it always shows up in the same row? In the Identification method area, you can set the Rows value to **By Location** to verify the row is always in the same place or **By Content** to just verify that row is somewhere in the table.

10. For this test, take the defaults, which perform a case-sensitive test of all the data and expects each row and column to appear in the same place each time. Click **OK** on the Object Data Verification window.

### Closing Classics Online and stopping the recording

Rational Robot goes back to recording mode. Your test scenario is almost complete. To make sure the database is reinitialized for your next playback, delete this order:

1. Click on the Row for the order you just placed in the table.
2. Click **Cancel Selected Order**.
3. Click **Close** on the View Existing Orders window.
4. Select **File > Exit** from the Classics Online main menu.
5. Stop Rational Robot recording by clicking the square Stop Recording button on the GUI toolbar.

---

**Section 8. The test script**

**Script window**
Now that you have finished recording, Rational Robot restores itself. Maximize Rational Robot and maximize the script window within Rational Robot.

The largest pane is the script itself.

```vbs
Sub Main
  Dim Result As Integer

'Initially Recorded: 10/25/2004 10:41:33 AM
'Script Name: OrderItem
StartApplication ""C:\Program Files\Rational\Rational Test\Sample Applications"

Window SetContext, "Name=frmExistingLogin", ""
PushButton Click, "Name=cmdOK"

Window SetContext, "Name=frmMain", ""
TreeView Click, "Name=treMain;\;ItemText=Beethoven", ""
TreeView DBIClick, "Name=treMain;\;ItemText=Beethoven", ""
TreeView Click, "Name=treMain\;\;ItemText=Beethoven->Symphony No. 9", ""
PushButton Click, "Name=cmdOrder"

Window SetTestContext, "Name=frmOrder", ""
Result = EditTextBoxV (CompareProperties, "Name=txtName", "UP-Customer Name")
Window ResetTestContext, "", ""

Window SetContext, "Name=frmOrder", ""
EditTextBox Click, "Name=txtCreditCard", "Coords=17,9"
InputKeys "1234567890"
EditTextBox Click, "Name=txtExpirationDate", "Coords=29,5"
InputKeys "12/06"

```

The commands you see are in a language called SQABasic. SQABasic is an extension of the Visual Basic programming language. As you can see, it is easy to read and understand what the script will do. Commands like StartApplication, PushButton Click, and InputKeys are pretty self-explanatory. The parameters provide additional information to the commands. For example, look at the command:

```vbs
EditBox Click, "Name=txtCreditCard", "Coords=17,9"
```

The **Name** parameter tells Rational Robot on playback which EditTextBox in which to click. The value of the name parameter is the name the developer gave the control when it was placed in the GUI of the application. Finding a stable property to use to identify an object during playback is crucial to creating scripts that are robust against GUI changes from one build to the next. Rational Robot searches through all the available properties of an object to find the one that is the most stable for a given environment. In Visual Basic, that is typically the name property. If a developer changes the location, size, or even caption on a control, it is unlikely that the developer will change the name of the control because that would require recoding everywhere that control is referenced in the source code.
The **Coords** property tells Rational Robot where within that control you clicked. Note that these are the X and Y coordinates relative to the upper-left corner of the control, not the screen or window. Should this EditBox be relocated on the window in some future build, Rational Robot will still be able to locate the control based on its name, not its absolute position on the screen. This is very important for maintainable, robust scripts.

**The verification point window**

You created two verification points during recording that carried a lot of information with them. All that you see in the script is something like:

```
Result = EditBoxUP (CompareProperties, "Name=txtName", "VP=Customer Name")
```

Where is all that information and how can you review it? The verification points are collected in the tree control on the left side of the user interface.

![Verification Points](image)

If you double-click on the **Customer Name** VP in this list, you can see the properties and the values of those properties that you are testing.

![Properties](image)

You can even edit the values or add and remove properties to the test from the menus on this window. This comes in handy when you know a change is coming in the next build of the application. You can edit your baseline data in advance and be ready when the new build becomes available to you for testing.

---

**Section 9. Edit the script**

---

---
Why edit a script?

Recording and playing back a script provides a lot of value, but as users advance in their application of Rational Robot, they often want to do more with scripting than they can with just recording. The SQABasic language provides a rich set of capabilities for adding decision logic, looping, and even interfacing with external tools. One example might be using COM programming to enable your Rational Robot script to open an Excel spreadsheet and read input values and expected values from a table. There is really no limit to what you can do with a little programming ability.

The SQABasic language scripts can be edited through the editor interface. Several menu items provide methods to insert items into the script. You can also simply type commands into the editor window.

Invoking Build B

Let's say you have just been notified that a new build of your application has been completed. Now that you have a functional test you developed against Build A, you want to run that script against Build B as a regression test -- you want to confirm that the application still does in Build B what it did in Build A.

The Build B application was installed when you set up your sample applications. To invoke it, edit the StartApplication command in your OrderItem script.

Locate the line in your script that begins with **StartApplication**. It should be approximately on line 6. Scroll to the far right end of that line. Replace ClassicsA.exe with ClassicsB.exe.

Click the Compile button on the Rational Robot toolbar to verify that no syntax errors were made while making that change.

Section 10. Play back the script
What to expect

Before you play the script, you should have an idea of what happens. Rational Robot minimizes itself as it begins to play back the script. **Do not touch the keyboard or mouse during playback!** Any manual interaction with the machine might disrupt playback.

Rational Robot plays the script much faster than you could manually. Besides the repeatability of an automated testing approach, the computer can perform tests much faster than you ever could manually, reducing your test cycle times.

Rational Robot will start up Classics Online Build B and start performing exactly the same interactions you recorded in Build A. Note the arrangement of the main window of Classics Online Build B. The controls have been rearranged on the screen, the look-and-feel of the tree control has changed, even the caption on the press here to order button has changed to **Order It!**. None of these changes keep Rational Robot from replaying the script. This resilience to change is a key capability of Rational Robot, which minimizes ongoing maintenance to test scripts.

Near the end of the scenario, a simulated network error window appears. Again, resist the urge to interact with the screen. After a minute, Rational Robot dismisses the window and completes the playback. This ability to deal with what are called "unexpected active windows" is another key feature of Rational Robot that helps make your scripts more resilient to application changes and unexpected events during playback.

Launching playback

Click **Playback Script** on the Robot toolbar. The Playback window lists all the scripts in your datastore. If you have other scripts, be sure that **OrderItem** is selected then click **OK**.

Before Rational Robot launches the playback, it asks you where you want to store the results log. Rational Robot uses an arrangement of builds, log folders, and logs to store results. This hierarchy allows you to organize your results logs in a manner that works for your team. Click **OK** and sit back and watch the playback. Again, do not interact with the screen until you see **TestManager** appear.
Section 11. Analyze results

Test log

Upon completion of the playback, the test log opens in Rational TestManager. Expand all the nodes in the test log and widen the Event Type column so you can see more of the names.

The Test Log captures all the significant events during playback such as application starts, verification points, and unexpected active windows encountered. If any event fails, its parent will be marked as a failure. From this log, you can see the OrderItem script was successful in starting the Classics Online application and the first verification point passed -- the VP that verified the customer name field properties. The second verification point failed, however. To investigate, double-click on the Verification Point (Existing Orders - Object Data) event in the Test Log. This brings up the Grid Comparator. The Grid Comparator shows what was expected on the left and what was actually seen on Build B on the right. To make it easier to see in limited screen real estate, select View > Transpose View from the Grid Comparator menu.
Cells highlighted in red did not match between the Baseline and the Actual. From this you can see that, although you placed an order for Beethoven Symphony No. 9, an order was placed for Bach Brandenburg Concertos Nos. 1 & 3. This is obviously a bug that has crept into Build B that you were able to uncover through your automated functional regression testing.

**Section 12. Summary**

In this tutorial, you took a look at some of the basic capabilities of Rational Robot. You saw how to:

- Install Rational Robot
- Create the necessary datastores
- Set up a sample application
- Record a script with Rational Robot
- Edit the script in the editor window
• Playback the script against a new build of the application
• Analyze the results

There are many other capabilities provided by Rational Robot that you can explore on your own. See Resources for additional recommended reading. Give Rational Robot a try on your own application!
Resources

- Participate in the discussion forum for this content.
- Download Rational Robot.
- Read the *Rational Robot User’s Guide* for more information on advanced capabilities and how to use them.
- Learn more about Rational Robot from IBM developerWorks.

About the author

Dennis Schultz

Dennis Schultz joined Rational in 1995 as a Technical Sales Engineer. For eight years, he worked closely with numerous clients implementing Rational solutions in their projects. Dennis helped deploy solutions for Software Configuration Management, Change Management, Requirements Management, and Test Management and Implementation. Since 2003, Dennis has been a Technical Marketing Engineer for IBM Rational software. Dennis holds a B.S. in Computer Engineering from Iowa State University. He is based in St. Louis, Missouri and fills his non-work time with his four children.