

# Managing application testing

## Using Rational TestManager to optimize your tests

Skill Level: Introductory

[Sandra Wilkey](#)  
IBM

11 Oct 2004

Are you finding it hard to know exactly what requirements of your application have been tested? Do you struggle to understand how much testing you have completed, how much is left, and what is the real status of the application under test? If so, Rational TestManager can help. Rational TestManager is a tool designed to help project teams manage testing -- from initial test case planning, to test development, execution of the tests and analysis of the results. Rational TestManager can give you the data you need to make the critical decisions about deploying an application.

## Section 1. Introduction

### About this tutorial

All development teams would like to live in a world where a perfect application was delivered as soon as the last line of code was written. Application or system testing is often viewed as a drain on time and resources -- extending an already tight development schedule. But as all project managers know, usually from experiencing failing applications once they are in the hands of the users, testing is a necessary part of any application development project. One of the keys to delivering a high-quality application without allowing testing to derail the project's schedule is to spend time planning and executing your tests in parallel with the development of the application. By including test planning and test execution as a critical step of application development, testing will not be relegated to a short block of time at the

end of the development cycle or fall off the schedule completely. If you start testing early, you can weed out critical bugs when developers will still have enough time to fix them and the testers will have enough time to conduct thorough testing of the application.

The job of IBM® Rational® TestManager is to provide the entire project team with a centralized tool to plan, design, and execute tests and to provide reports that will allow the project team to make an informed "go" or "no-go" decision about the readiness for delivery of the application.

This tutorial is written for application testers and test managers who struggle to organize and control the testing efforts and to collect the information needed to assess the quality of an application. Here you'll learn the basic mechanics of test planning and how Rational TestManager can help you manage planning, execution, and analysis of the test results. No pre-existing knowledge of Rational TestManager or Rational RequisitePro is required.

The following topics are covered in this tutorial.

- Setting the requirements that will be the basis of testing
- Creating a test plan
- Developing manual or automated tests
- Executing tests and analyzing results
- Monitoring test progress

## Prerequisites

To complete this tutorial, you need to install [Rational TestManager](#). Download a trial copy from developerWorks. After registering, you will be prompted to download and install Rational TestManager and Rational XDE Tester. Download and install only **Rational TestManager**.

An optional, but recommended, part of this demo includes using RequisitePro to capture and store requirements and feed them to Rational TestManager. If you would like to complete those steps in the tutorial download the [RequisitePro](#) software from developerWorks.

After you have downloaded these applications, install them using the defaults.

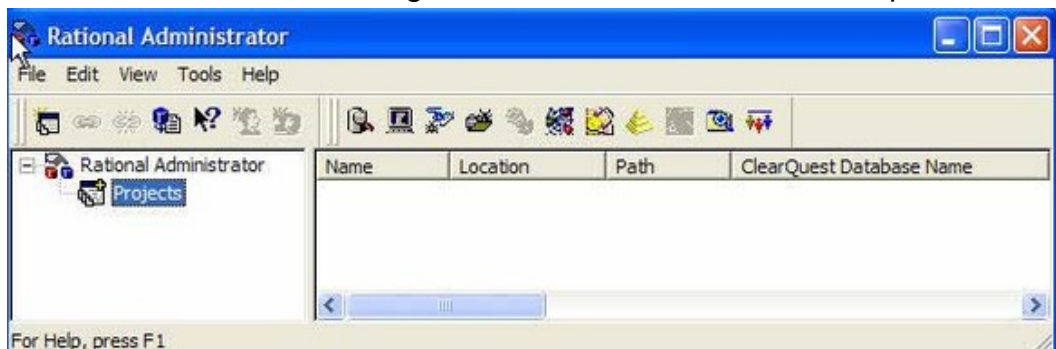
---

## Section 2. Get started

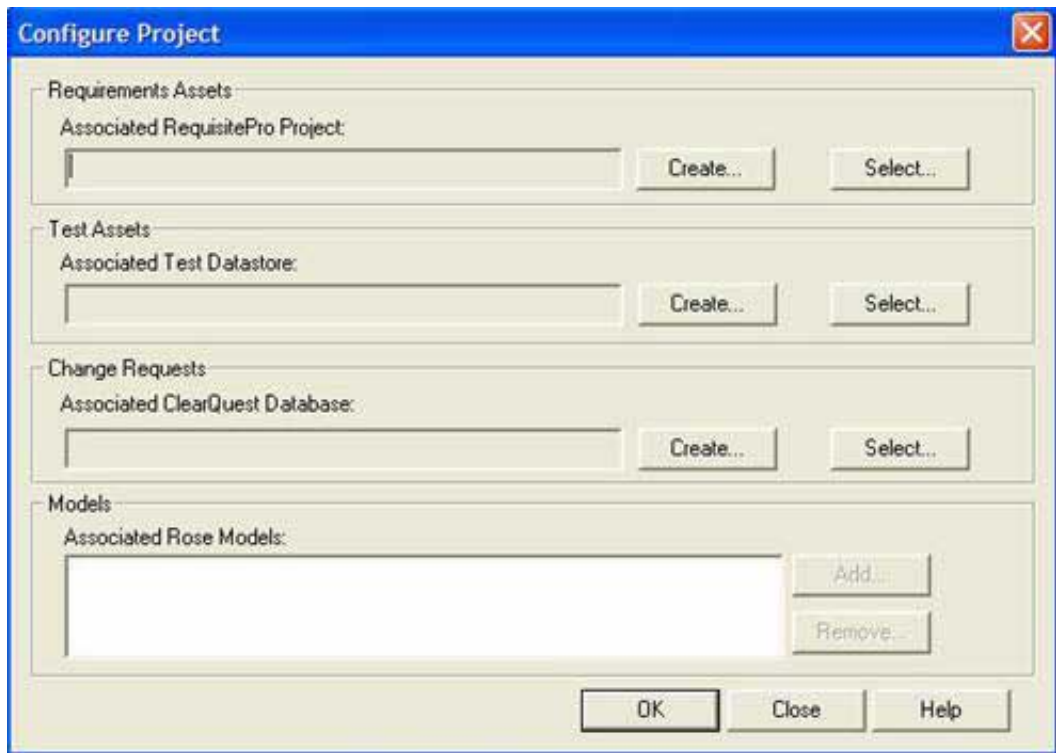
### Creating a test project

Begin by creating a test project to store all of your test assets (test cases, test results, and so forth).

1. Select **Start > Programs > Rational Software > Rational Administrator**. The following Rational Administrator window opens.



2. Select **File > New Project**.
3. Enter the name `TMTutorial` in the Project Name field and enter a location (for example `C:\TMTutorial`) in the Project location field and click **Next**.
4. Click **OK** on the window reminding you to use UNC formatting if you want to share the project.
5. In the New Project -- Security you can either enter a password or ignore this and click **Next**. If you enter a password, don't forget it. You'll need this password to access your test project.
6. In the New Project -- Summary window, make sure the Configure Project Now check box is checked and then click **Finish**.
7. The Configure Project window opens.



Complete the steps in the next section, **Creating the RequisitePro project**, only if you downloaded and installed the evaluation software for Rational RequisitePro. Otherwise, skip to [Creating the test datastore](#) to create the datastore that will hold your test assets.

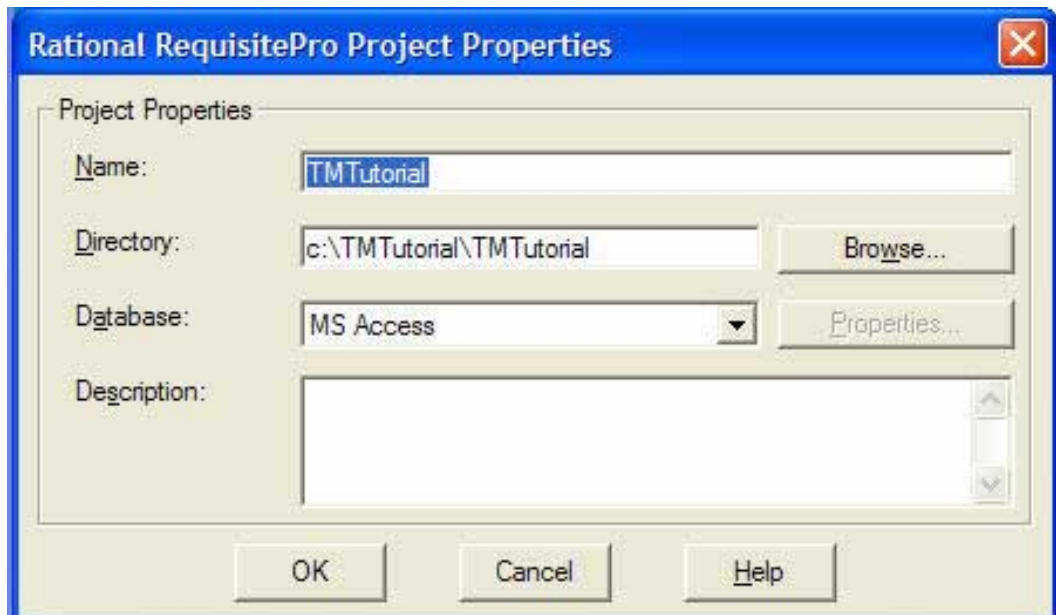
## Creating the RequisitePro project

Create a project in Rational RequisitePro to store application requirements.

1. Click the Create button next to **Associated RequisitePro Project**. The following window opens.



2. If you commonly employ use-cases to describe usage scenarios of your application, select **Use-Case Template**. If not, select **Traditional Template** and click **OK**.  
The Project Properties window opens.



3. Click **OK** then **Yes** to create the project infrastructure to store your requirements.
4. When the message stating that the project was created successfully appears, click **Close**.

## Creating the test datastore

1. At the Configure Project window click **Create** in the Test Assets group.
2. Select **Microsoft Access** then click **Next**. For this tutorial, Access is a sufficient database. In a real project with multiple users, select SQLAnywhere.
3. Click **Next** to accept the default path to the new test datastore.
4. Click **Next** on the Initialize New Test Datastore from existing assets window.
5. Click **Finish** on the Create Datastore -- Summary window.
6. Click **OK** on the window confirming successful creation of the test datastore.  
You have now finished creating the necessary datastores to contain your testing artifacts (and optionally your requirements). The project called

**TMTutorial** is displayed in the list of projects on the left.

7. Close the Rational Administrator.

---

## Section 3. Specify inputs to the test plan

### What are test inputs?

In most testing projects, you have gathered information from a variety of sources that guide you during testing. Common sources of information are the business or technical requirements of the application under development, a list of common problems on similar applications you have tested in the past, or a marketing or user specification of the application.

Application requirements are one of the most commonly used sources of information for test planning. Effectively managing and tracking all the application requirements is a critical step in developing a high-quality application. If you want to use requirements as a basis for developing tests, you can store those requirements in either Rational RequisitePro or Microsoft Excel. Both of those sources have prebuilt *feeds* into Rational TestManager. The benefit of linking your tests to requirements is that you are able to verify that every requirement is tested and you can report on the progress of testing against your list of requirements. Although you may find that you create additional test cases beyond those that test the application requirements, using requirements to guide your test plan is often a good place to start.

If you installed Rational RequisitePro and want to see how to store requirements in this tool, click **Next** in this tutorial to continue. If you did not install Rational RequisitePro skip to [Using inputs from Microsoft Excel](#).

### Using Rational RequisitePro as an input source

The full value and capabilities of Rational RequisitePro is not covered in this tutorial. A tutorial, [Managing your requirements and more with Rational RequisitePro](#) is available from the *developerWorks* site.

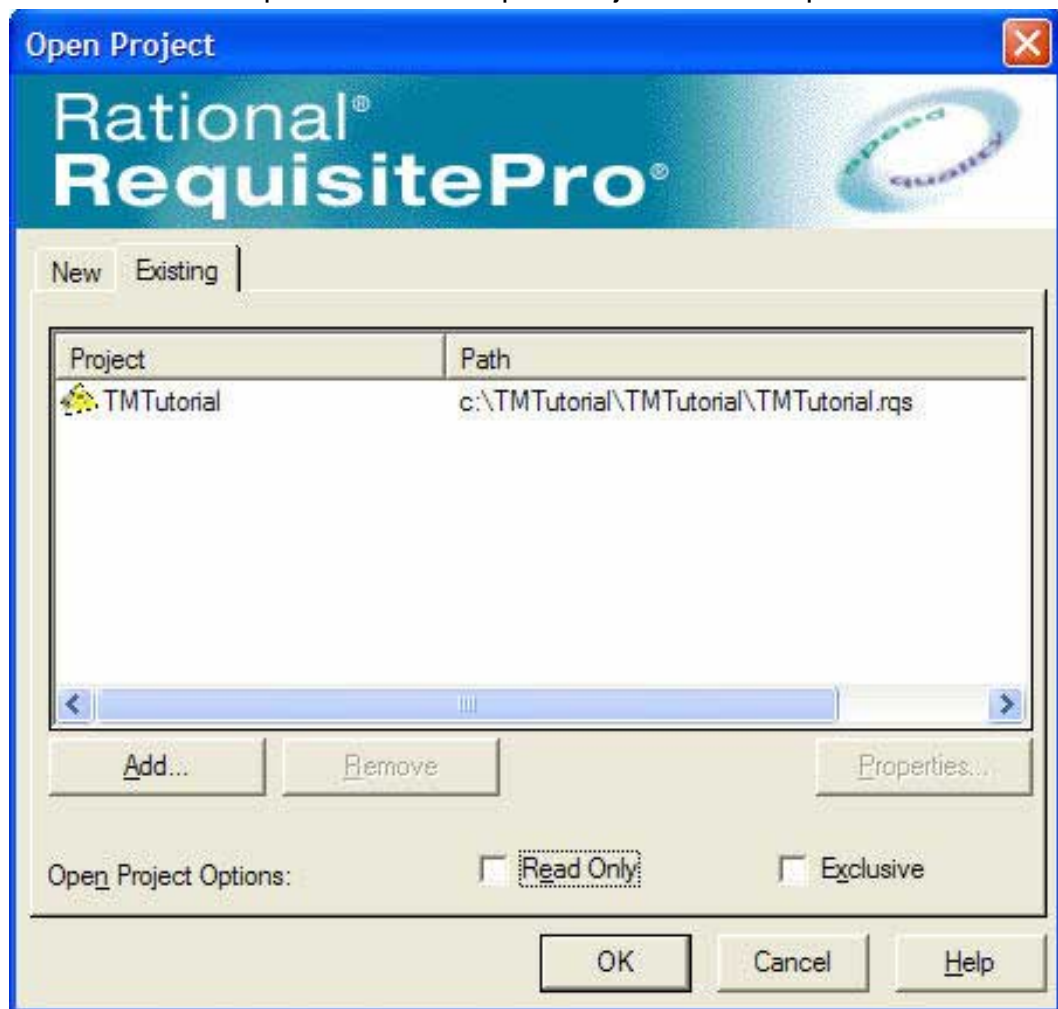
In this tutorial you will input the minimum amount of information needed in Rational RequisitePro for you to understand how Rational TestManager and Rational RequisitePro can work together.

Rational RequisitePro is a requirements management tool that improves the

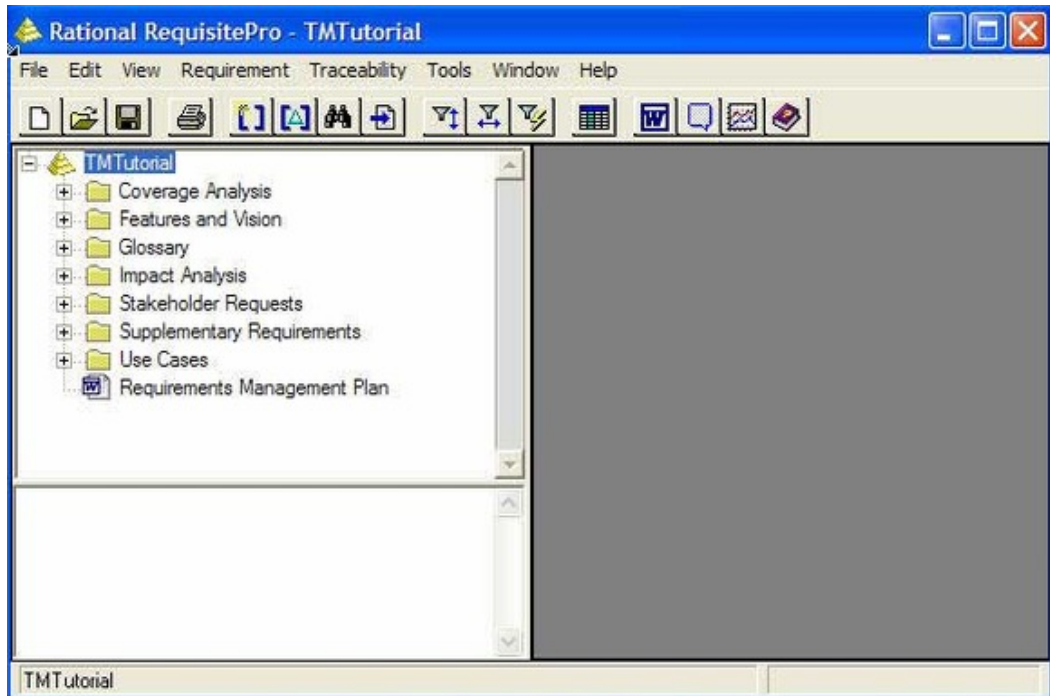
efficiency of documenting and communicating requirements and managing changes to requirements. It unites Microsoft Word and a database to allow project analysts to document requirements, along with their context and detail, in a Word document while using a linked database for tracking requirement attributes, querying, sorting and communicating changes.

In this tutorial you will be using only the database component of RequisitePro to enter a few requirements. Refer to the tutorial referenced in the previous section for more detailed information about the benefits and capabilities of RequisitePro.

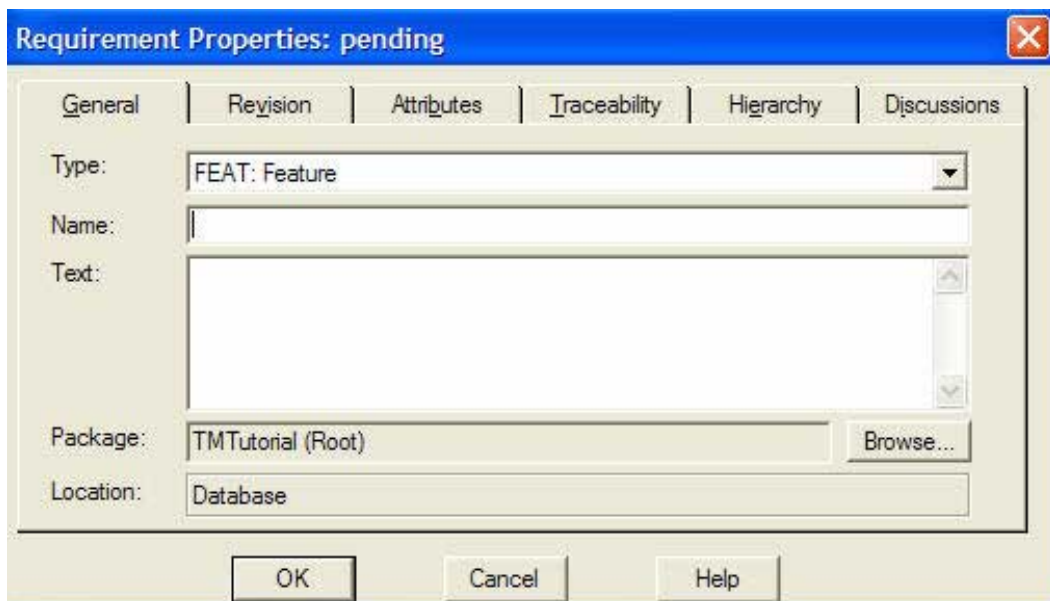
1. Start Rational RequisitePro. The Open Project window opens.



2. Click the TMTutorial project and click **OK**.  
The Rational RequisitePro main project window opens.



3. Select **Requirement > New** to open the Requirements Properties window.

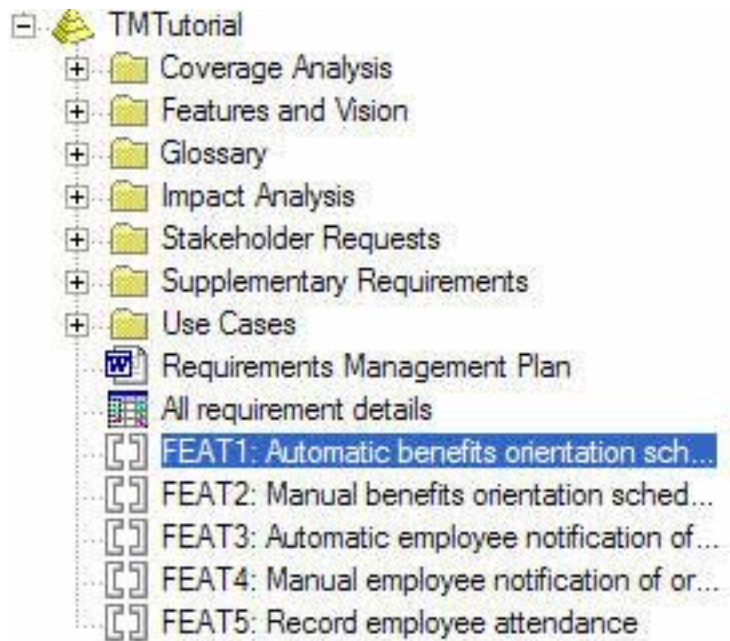


The Type field displays either **Feature** or **Use-case** depending on the template you selected when opening this tool.

Assume that you are documenting requirements for an application that manages the new employee orientation process at your company.

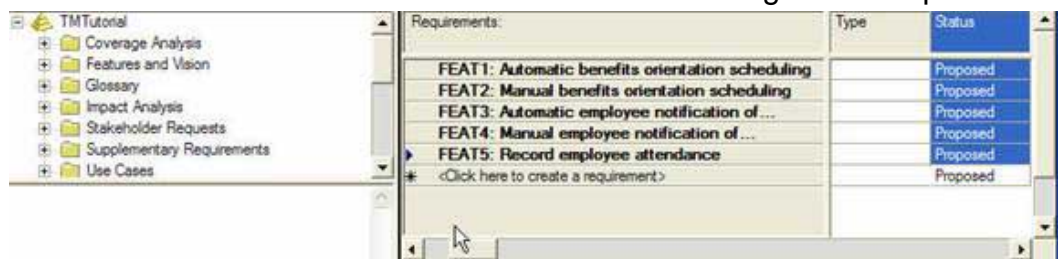
4. In the Name field enter, Automatic benefits orientation scheduling
5. In the Text field enter, The application will automatically prompt the administrator to schedule the new hire for the next available benefits orientation.
6. Click **OK**.  
You should now see the requirement name listed in the left pane.
7. Repeat steps 3 through 6, entering the following four requirement names and descriptions:  
**Name:** Manual benefits orientation scheduling  
**Text:** The application will allow the administrator to manually schedule the benefits orientation.  
**Name:** Automatic employee notification of orientation  
**Text:** The application will send an email to each new employee containing the date, time and location of their benefits orientation.  
**Name:** Manual employee notification of orientation  
**Text:** The application will allow the administrator to send an email to each new employee containing the date, time and location of their benefits orientation.  
**Name:** Record employee attendance  
**Text:** The application will allow the administrator to record the date of the new hire's attendance along with any questions that the employee may have.

When you are finished, the left pane of the window should display the requirements as shown. (Remember that you might see the tag **USCSC** instead of **FEAT** depending on which template you selected.)



## Setting requirement attributes

1. Double-click on any requirement to open it.
2. Click on the Attributes tab. Here you can see that there are a series of attributes assigned to a requirement. These attributes are customizable and are used to sort and query on requirements. For this tutorial, you'll want all of your requirements to have the status of *approved*.
3. Click on the TMTutorial project name at the top of the tree structure in the left pane.
4. Select **File > New > View**.
5. In the View Properties window enter the text `All requirements detail` in the Name field and click **OK**. The following window opens.



6. Select the status property on all five requirements by dragging the mouse across the **Proposed** status of all five requirements.
7. Click the **Set the attribute of selected requirements with a single value** toolbar button. It looks like a table or grid.
8. In the Set Value window, select **Approved** then click **OK**. All requirements should have the status of Approved.
9. You now have enough information stored in RequisitePro to use it as an input to the test plan.
10. Close RequisitePro and respond **Yes** to the prompts to close the project and save the View.

To see how information stored in Excel can also be used as input to your test plan, click **Next** to continue in this tutorial. Otherwise, skip to the section [About test plans](#).

## Using inputs from Microsoft Excel

Rational TestManager can use information stored in other sources as inputs to the test plan. Rational TestManager provides you with an API that allows you to create links to data sources such as a database or a document. If you use Microsoft Excel to capture and organize requirements, Rational TestManager has a prebuilt link that can access data stored in an Excel spreadsheet and link it to your test plan.

1. Create a new Excel spreadsheet.
2. Begin by entering the following column headings:  
In cell A1 enter the text `Requirement Name`  
  
In cell B1 enter the text `Description`  
  
In cell C1 enter the text `Status`  
  
In cell D1 enter the text `Last Modified`  
  
In a real project the spreadsheet would likely be more complex with more information, but this will be sufficient for the purposes of this tutorial.
3. Select column D and set the format for all cells in that column to **Date**. Pick a format that includes date and time (such as 3/14/01 12 PM)

- Enter the requirement names, descriptions, and statuses as shown in the following image. Enter the current date/time in the Last Modified field.

	A	B	C	D
1	<b>Requirement Name</b>	<b>Requirement Description</b>	<b>Status</b>	<b>Last Modified</b>
2	Automatic benefits orientation scheduling	The application will automatically prompt the administrator to schedule the new hire for the next available benefits orientation.	Approved	8/19/04 12:00 PM
3	Manual benefits orientation scheduling	The application will allow the administrator to manual schedule the benefits orientation.	Approved	8/19/04 12:00 PM
4	Automatic employee notification of orientation	The application will send an email to each new employee containing the date, time and location of their benefits orientation.	Approved	8/19/04 12:00 PM
5	Manual employee notification of orientation	The application will allow the administrator to send an email to each new employee containing the date, time and location of their benefits orientation.	Approved	8/19/04 12:00 PM
6	Record employee attendance	The application will allow the administrator to record the date of the new hire's attendance along with any questions that the employee may have.	Approved	8/19/04 12:00 PM

- Save the spreadsheet with the name `TMTutorial_Requirements` and store it in the same location as the project (for example `C:\TMTutorial`).

You now have an Excel spreadsheet with requirements data that can be used as the basis for developing your test plan and test cases.

The next section shows you how to create a test plan using Rational TestManager and link the test cases in the test plan to the application requirements that you are tracking in this Excel spreadsheet.

## Section 4. Create a test plan

### About test plans

A test plan is the mechanism used in Rational TestManager to organize and create all the test cases for your project. A test plan can be created stand-alone (meaning with no references to inputs from sources such as RequisitePro or Excel) or it can be

created by starting with test inputs and creating one or more test cases for each input. In our sample project, you have requirements stored in either RequisitePro or Excel. The benefit of linking test cases and application requirements is that you will have the ability to report on test progress against your requirements and to receive notification of which test cases are impacted when requirements are changed.

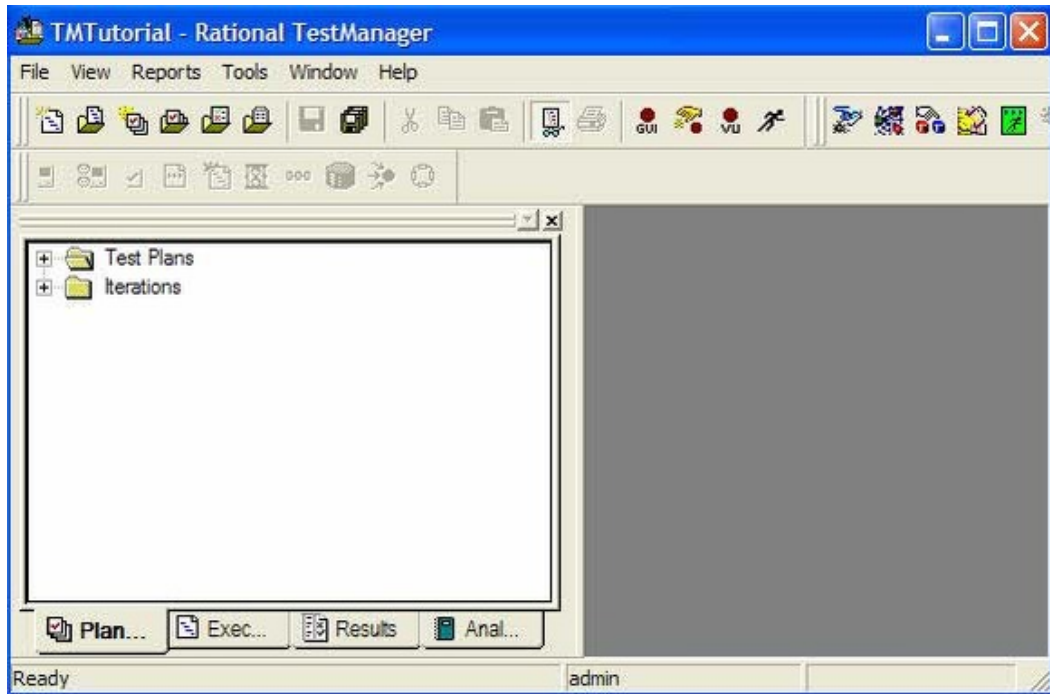
Test cases, along with their attributes and their organization, are the meat of a test plan.

If you created requirements in RequisitePro, this source has been automatically linked and will be available when you log into Rational TestManager. If you created your requirements in Excel, you need to perform an additional step to connect your Excel spreadsheet to your Rational TestManager project.

In the following sections, you will use Rational TestManager to create the structure of your test plan. After this step you will learn how to connect your test cases to the requirements they validate.

## Creating the test plan structure

1. Start Rational TestManager.
2. At the login screen select the project **TMTutorial**.
3. Enter the name `Admin`.
4. If you created a password when you created the project, enter that password. Otherwise, leave the password field blank.
5. The Rational TestManager window opens. (If you do not see a Test Plans folder in the left pane click on the Planning tab in the bottom left corner of the window.)



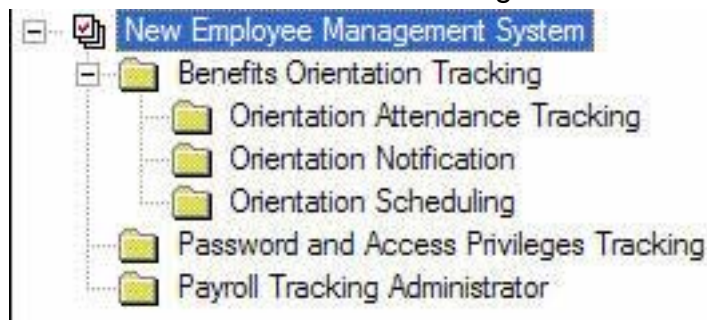
6. Expand the Test Plans folder.
7. Right-click on the default test plan named **Test Plan 1** and select **Rename**.
8. Enter the name `New Employee Management System`.
9. Double-click on this test plan to open it.
10. Next, create folders to organize your test plan. There are many ways to organize a test plan. Some common choices are by:

- Component of the application under test
- Business function
- Test case priority

In this tutorial, you will create test case folders based on business function.

11. Right-click on the folder named **Default** and delete it. Now you are ready to create the test plan folders to store your test cases.
12. Right-click on the test plan name (or a folder) then select **Insert Test**

**Case Folder** to create the following folder structure.

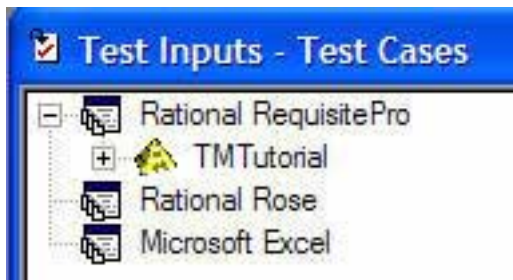


You have now created the folder structure for your test plan. If you used RequisitePro to document your requirements, click **Next** to continue to the next section. If you used Excel to document your requirements, skip to [Establishing Microsoft Excel test inputs](#).

## Viewing RequisitePro test inputs

From Rational TestManager, view the requirements stored in RequisitePro and attach test cases to those requirements.

1. From the Rational TestManager menu select **View > Test inputs**.
2. The following window opens.



If you expand the TMTutorial folder you will see several folders and the five requirements that were created in RequisitePro. From this view, you can right-click on a requirement and select **Insert Test Case**. This allows you to create a test case, select the folder to place it in, and connect it to the requirement that it validates. You can attach one or more test cases to a requirement as needed. As you create test cases, they will appear beneath the requirement. Once test cases are linked to requirements you have the following capabilities:

- Track the progress of test case planning, development, and execution for

each requirement.

- Track the results of test case execution for each requirement.
- Notification of a requirement change and which test cases are impacted by the change.

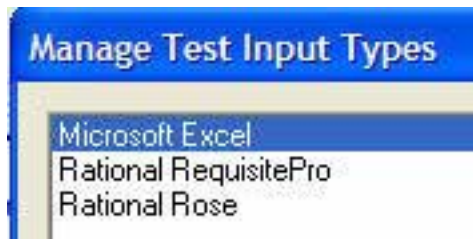
A test case holds information about the test to execute. Some of the common data elements attached to a test case are: a description, an owner, associated test inputs and an associated test script that will be used to either manually or automatically validate the test case.

Once you have viewed your requirements in RequisitePro, skip to [Creating test cases](#) to learn how to create test cases and connect them to the requirements stored in RequisitePro.

## Establishing Microsoft Excel test inputs

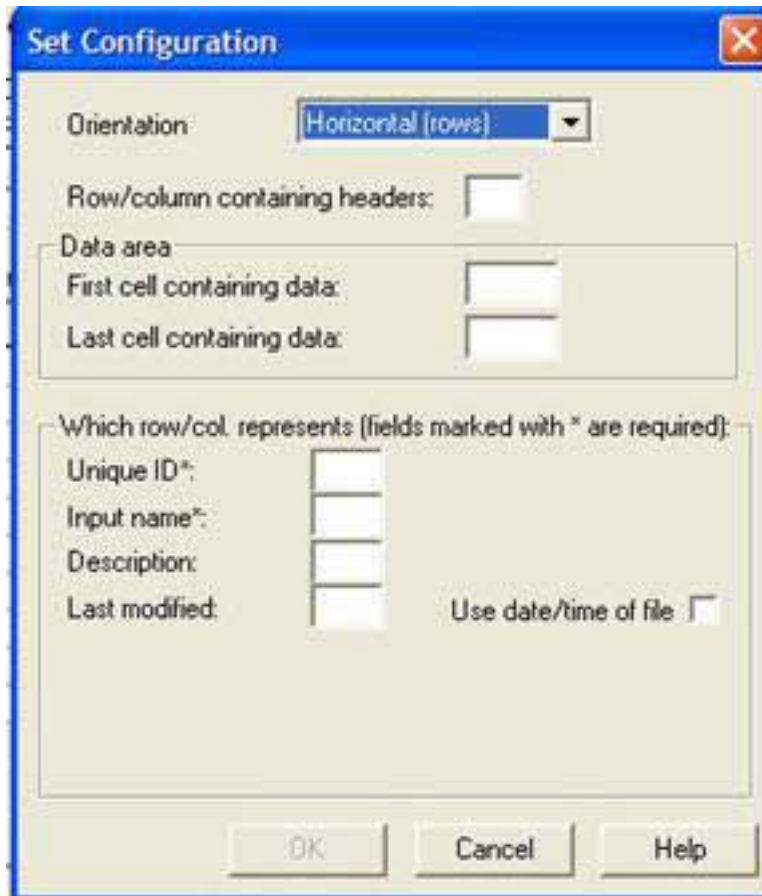
If you used Excel to store your requirements, you can view them from Rational TestManager once you have created a connection to the Excel spreadsheet. Rational TestManager can connect to one or more spreadsheets in a project. Complete the following steps to create a connection between Rational TestManager and your Excel spreadsheet.

1. Select **Tools > Manage > Test Input Types**.  
The list of input types is displayed.



2. Select **Microsoft Excel** and click **Edit**.
3. Click on the Sources tab.
4. Click **Insert**.
5. In the New Input Source window enter the name `Application Requirements`.
6. Click on the Connection Data tab.

7. In the Data path field browse to the Excel spreadsheet (for example C:\TMTutorial\TMTutorial\_Requirements.xls)
8. Click **Set Configuration** and respond **Yes** to save the source.
9. The Set Configuration window opens.



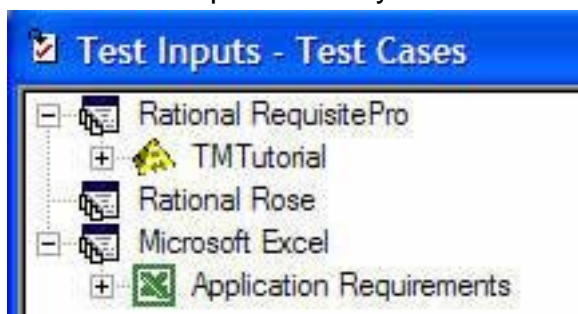
In this window, specify which columns contain the information that will be used to display information in Rational TestManager and to keep track of changes to the requirements. Complete this window with the following information:

10. In **Row/Column containing headers** enter 1.
11. In **First Cell containing data** enter A2.
12. In **Last Cell containing data** enter D6.
13. In **Unique ID** enter 1 (indicating the first column).

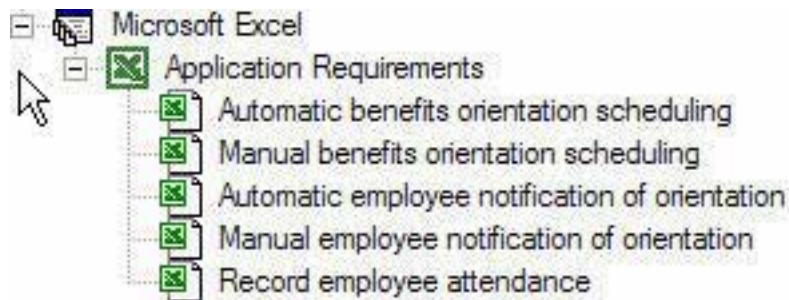
14. In **Input Name** enter 1.
15. In **Description** enter 2.
16. In **Last Modified** enter 4.
17. Click **OK** to save this configuration.
18. Click **OK** on the New Test Input Source window.
19. Click **OK** on the Test Input Type window.
20. Click **Close** on the Manage Test Input Types window.  
You should now be able to see the contents of your Excel spreadsheet from Rational TestManager. Follow the steps in the next section to view the inputs to your test plan.

## Viewing Microsoft Excel test inputs

1. From the Rational TestManager menu select **View > Test inputs** (if the Test Inputs window is already open, close and reopen it to refresh its content).
2. The following window opens. **Note:** The window does not display a source for RequisitePro if you used Microsoft Excel.



3. Click on the **+** next to **Application Requirements** to view the following list of requirements.



From this view, you can right-click on a requirement and select **Insert Test Case**. This allows you to create a test case, select the folder to place it in, and connect it to the requirement that it validates. You can attach one or more test cases to a requirement as needed. As you create test cases, they appear beneath the requirement. Once test cases are linked to requirements, you have the following capabilities:

- Track the progress of test case planning, development and execution for each requirement.
- Track the results of test case execution for each requirement.
- Notification of a requirement change and which test cases are impacted by the change.

A test case holds information about the test to execute. Some of the common data elements attached to a test case are: a description, an owner, associated test inputs and an associated test script that will be used to either manually or automatically validate the test case.

Here you can see the names identifying each requirement to be tested.

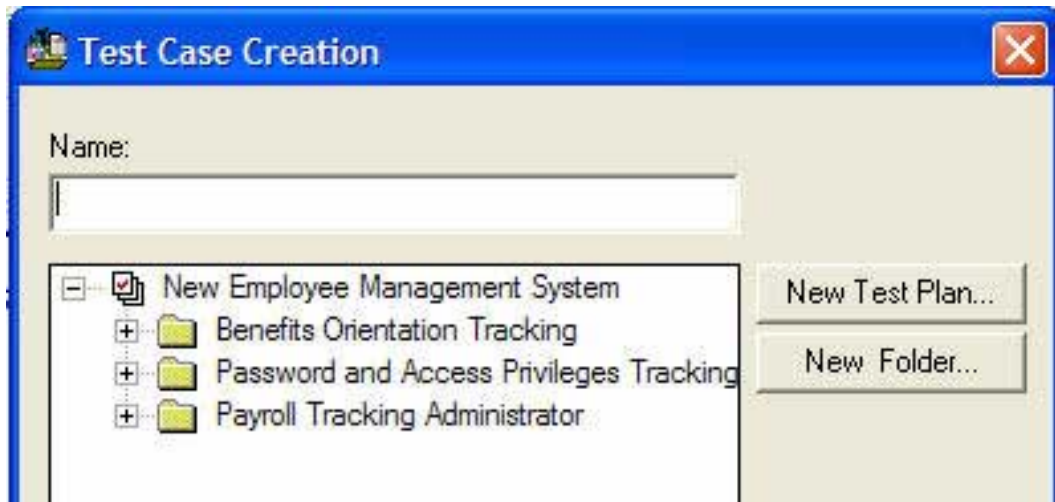
If you right-click on a requirement and select **Properties**, you can view the other details in the spreadsheet for each requirement.

Continue to the next section to create test cases and connect them to the requirements stored in Excel.

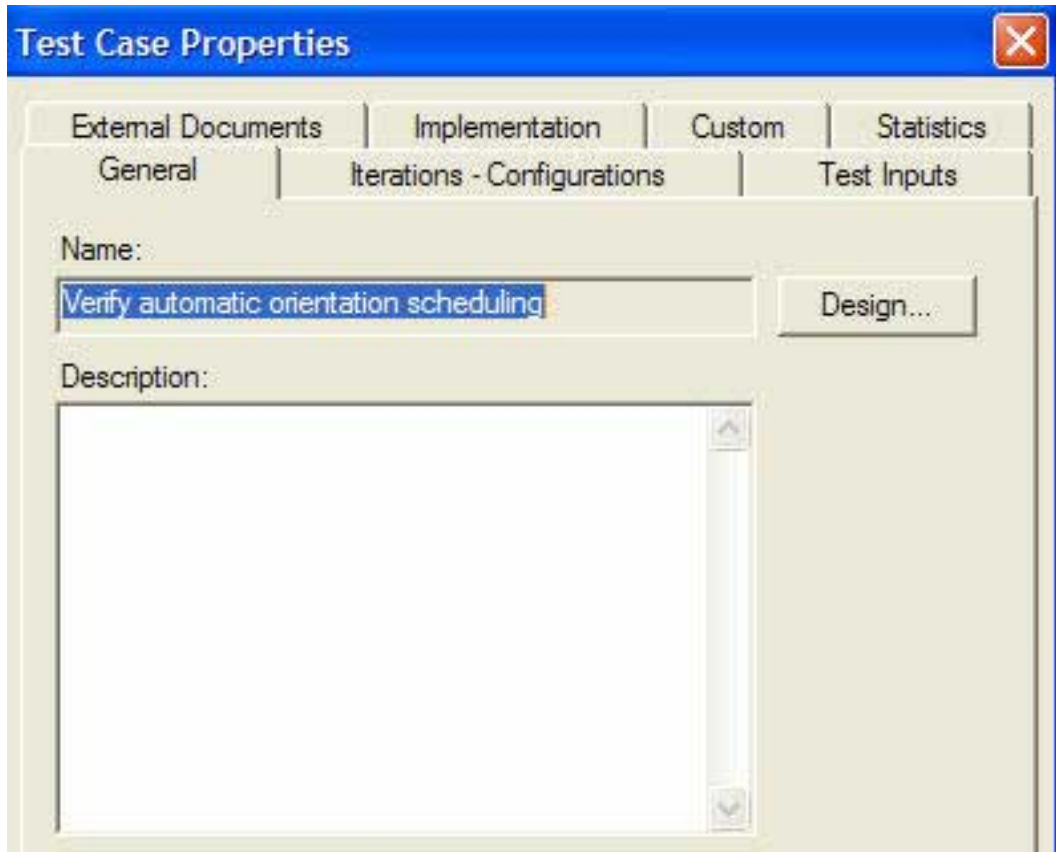
## Creating test cases

1. Ensure that the Test Inputs window is open. If you used RequisitePro to document your requirements expand **TMTutorial** under RequisitePro. If you used Excel, expand **Application Requirements** under Microsoft Excel. Work through the following directions using whichever requirements source you created.

2. Right-click on the requirement **Automatic benefits orientation scheduling** and select **Insert Test Case**. The Test Case Creation window opens.



3. In the Name field enter `Verify automatic orientation scheduling`.
4. Expand **Benefits Orientation Tracking** then select **Orientation Scheduling**.
5. Check **Edit properties now** in the lower left corner then click **OK**. The Test Case Properties window opens.



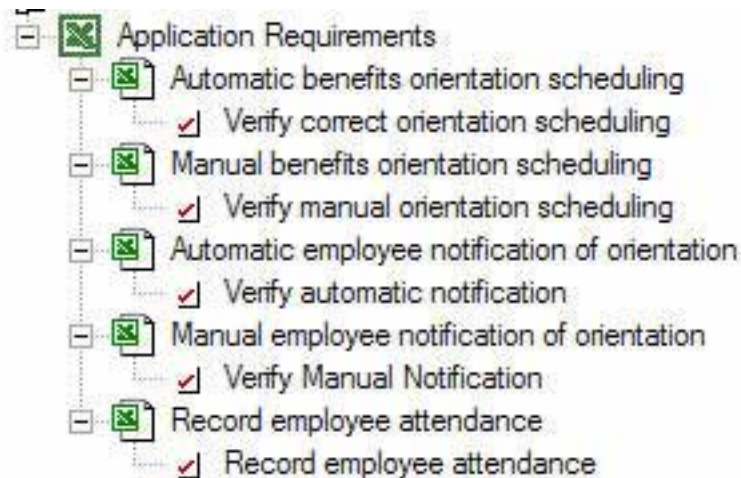
6. In **Description** enter: This test will verify that upon adding a new employee to the system, the application will pick the next available date/time after the new hire's start date and schedule that date for orientation.
7. You will add other details later, so click **OK** to save this test case. The test case opens beneath the requirement. This is just one way to create test cases and attach them to requirements. You could also create all the test cases in the Test Plan view first, and then use the Associated Test Case option to attach the test cases to requirements.
8. Using the data from the following table to add a few more test cases to your test plan. Entering the description text is optional.

Requirement	Folder location	Test Case Name	Description
Manual benefits orientation scheduling	Benefits Orientation Tracking > Orientation Scheduling	Verify manual orientation scheduling	This test will verify that upon adding a new employee to the system, the administrator can

			override the automatic scheduling and select the date/time for orientation.
Automatic employee notification of orientation	Benefits Orientation Tracking > Orientation Notification	Verify automatic notification	This test will verify that the new employees automatically receive an email with the date/time and location of their benefits orientation.
Manual employee notification of orientation	Benefits Orientation Tracking > Orientation Notification	Verify manual notification	This test will verify that the administrator can overview the notification and send a manual email with orientation information.
Record employee attendance	Benefits Orientation Tracking > Orientation Attendance Tracking	Record orientation attendance	This test will verify that the administrator can record the date/time of attendance and questions the new hire posed.

When you are finished, your Test Inputs window contains the test cases and links shown in one of the following two pictures (depending on whether you used RequisitePro or Excel to document your requirements).

- [-] [ ] FEAT1 Automatic benefits orientation scheduling
  - [-] [ ] ✓ Verify automatic orientation scheduling
- [-] [ ] FEAT2 Manual benefits orientation scheduling
  - [-] [ ] ✓ Verify manual orientation scheduling
- [-] [ ] FEAT3 Automatic employee notification of orientation
  - [-] [ ] ✓ Verify automatic notification
- [-] [ ] FEAT4 Manual employee notification of orientation
  - [-] [ ] ✓ Verify Manual Notification
- [-] [ ] FEAT5 Record employee attendance
  - [-] [ ] ✓ Record employee attendance

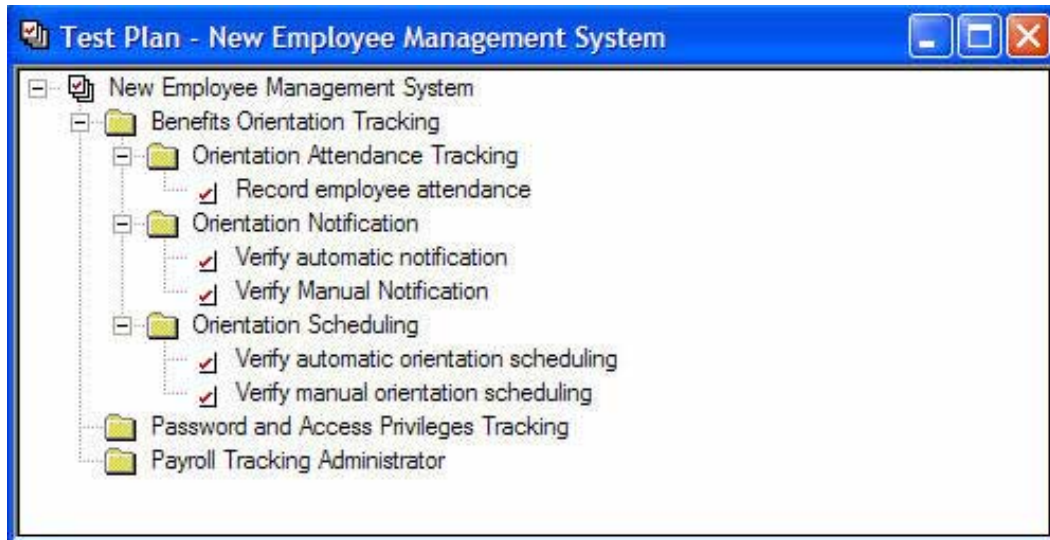


All of the requirements that you entered have at least one test case to verify them. In the following section, you will take a closer look at a test case and its content.

## Understanding test case details

Let's take a closer look at the data associated with a test case.

1. Switch to the Test Plan window if it is already open. Select **View > Refresh** to ensure it shows all the latest content. If the Test Plan window is not open, double-click on the test plan name in the left pane to open it.
2. When the test plan opens, right-click on the test plan name **New Employee Management System** then select **Expand Node**.
3. You should see the test plan displayed as shown.



As you have already learned, test cases contain names and descriptions, but they also hold other information about the test case.

4. To look at some of the choices, right-click on the test case **Verify automatic orientation scheduling** then select **Properties** from the menu.

The test case properties window contains several tabs of information.

The General tab should already contain a name and description. From this tab you can also click on the Design button to create an outline of the high-level steps for this test case. A test case design is generally used as the basis for creation of either a manual or automated test script. One productivity feature is the ability to automatically generate a manual test directly from the test case design. Additionally, if you already have test case design information in Excel you can import that into TestManager.

In the Test Inputs tab you will see the test input (requirement) associated with this test case. If this test case was related to more than one test input, you can use this tab to create associations to other test inputs. For example, it is possible that a test case is related to a requirement and a record of data that might be stored in a spreadsheet. From this tab you can point to both of those sources.

The External documents tab is used to attach files to the test case.

The Custom tab shows the values of any TestManager fields that were customized.

The Statistics tab contains the date, time, and creator of the test case, as well as the date the test case was last modified.

The Iterations/configurations tab allows you to assign these attributes to your test case. These items are discussed in the next topic.

---

## Section 5. Work with test iterations

### What is an iteration?

An *iteration* is a phase of development that typically results in a deliverable (executable application). For example, in iteration 1 of development, the developers build and deliver to the testing team features a, b, c, and d. The testers then test features (or requirements) a, b, c, and d as part of iteration 1. It can take several releases (or builds) to deliver all four features. So within iteration 1, the testing team can actually test one or more successive builds of the application. Once this is complete, the next iteration of development and testing starts and a new set of functionality is delivered for testing. This cycle continues through several iterations until all functionality has been delivered and tested.

### Why use iterations?

For many testing teams it is important to be able to track testing progress by iterations. The testing team will also want to document for which iteration a test case is applicable. Use of assigned iterations can help prevent execution of test cases before the functionality is complete and ready for testing, and provide milestones against which test progress can be reported. Rational TestManager comes with a set of default iteration names. You can see them in the Planning Tab under the test plan name in the left-hand pane. You are free to delete, edit, and add iterations that are appropriate for your development process -- or if this concept of iterative development does not fit your work model, you can ignore this and report on testing progress using other categories. When you define an iteration you can assign it an owner and a start and end date.

---

## Section 6. Work with test configurations

### What is a configuration?

The Iterations - Configurations tab also provides the tester/designer the ability to attach configurations to the test case. This option is used to classify test cases by the system configurations that they are designed to run on. Rational TestManager has three predefined configurations or create your own configurations and the attributes that make up a configuration.

To create custom configuration, select **Tools > Manage > Configuration Attributes**. From this window, create the custom attributes that are used to define a configuration. For example, create an attribute called "browser" that contains a list of browsers types and/or versions that you want to execute your tests on. Once the "browser" attribute is created, add it to a configuration by selecting **Tools > Manage > Configurations**. Using configured test cases provides the following capabilities:

- The ability to define variations of test cases that are configuration-dependent. This includes the ability to have a different test script run depending on the configuration of the computer the test is executing on.
- The ability to report on test progress by configurations.
- The ability to have Rational TestManager ensure that a test case executes only on its assigned configurations.

## Creating configured test cases

To assign the three prebuilt configurations to the currently open test case:

1. Make sure the test case called Verify automatic orientation scheduling is open.
2. Click on the Iterations -- Configurations tab.
3. Click **Select** in the Configurations group.
4. Click **>>** to select all three configurations.
5. Click **OK** to close this window.
6. Click **OK** again to close the Test Case Properties window.
7. Three configured test cases appear in the configurations window. When the test case properties window is closed you will see the additions to the test plan as shown in the following image.



Once configured test cases are created, you have the option to run the same test script for each configured test case or have a test script execute that is tailored for a specific configuration. To change attributes of configured test cases, simply right-click on the configured test case that displays in the test plan and select **Properties**. Whether you use the same test script or configuration-dependent test scripts, you will be able to report on the progress of your testing on all of your configurations. Additionally, Rational TestManager will prevent execution of configured test cases on mismatched workstation configurations.

Continue to the next section to learn about developing and assigning manual and automated test scripts to test cases.

---

## Section 7. Develop manual or automated test cases

### Choosing the test implementation type

Test automation tools such as Rational Functional Tester or Rational Robot allow you to create automated test scripts which you can attach to your test cases. With Rational TestManager's open test execution architecture, you can also attach custom tests written in Java, Visual Basic, or any test script language that can be executed from the command line (perl scripts, .bat files, and so forth).

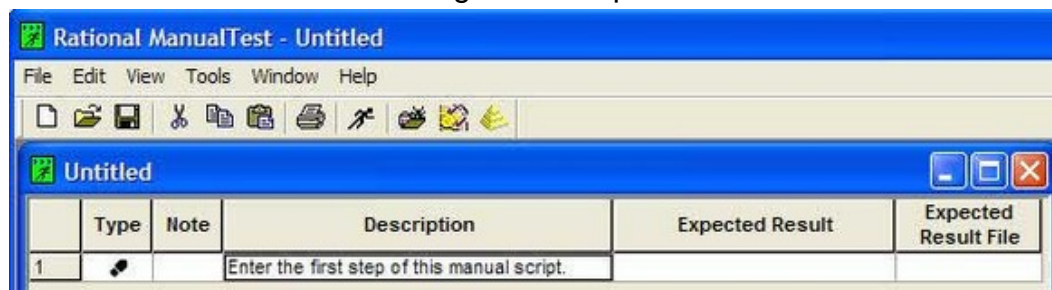
If you are not using a test automation tool to create automated test scripts, you can use the Manual Test creation tool built in to Rational TestManager to create manual test scripts. Manual test scripts contain a description of the steps to perform and the application behavior to verify to perform the test case. Manual tests can also contain notes, attached files, and other information to make test execution directions and results reporting more precise.

You do not have to use all manual or all automated tests. Mixing both types of test script in a single test plan is quite common.

This tutorial does not cover the use of test automation tools. If you are interested in that capability you should explore the Rational Robot and Rational Functional Tester sections of *developerWorks*. In this tutorial you will create two manual tests so that you can understand the process of attaching test scripts to test cases and executing test cases.

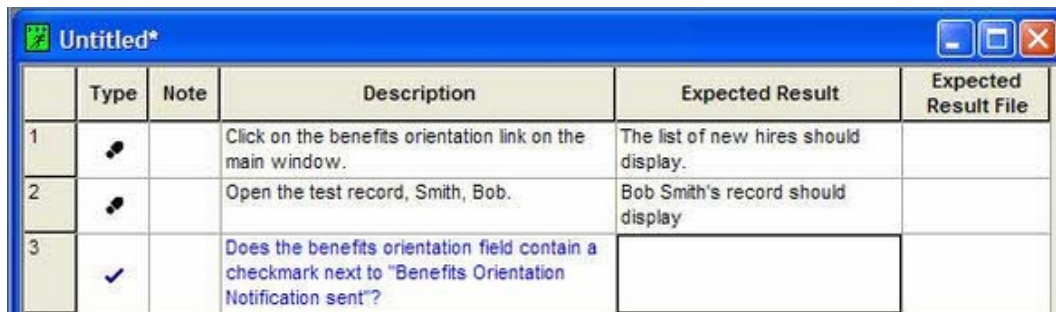
## Creating a manual test script

1. From Rational TestManager select **Tools > Rational Test > Rational ManualTest**. The Rational Manual Test tool will start.
2. Select **File > New**. The following window opens.



3. In the manual test window enter instructions, expected results, notes, and file attachments to describe a step in a manual test. Each row can be marked as either a "step" or a "verification point". A step is a direction to the test executor. A verification point is a question about the behavior of the application under test. The test executor must enter a result for any statement marked as a verification point. In this section, create two very simple manual tests.
4. In the Description field of row 1, enter: Click on the benefits orientation link on the main window.
5. In the Expected Result field of row 1, enter: The list of new hires should display.
6. Use the <Enter> key to move to the next row.
7. In the Description field of row 2, enter: Open the test record, Smith, Bob.
8. In the Expected Result field of row 2, enter: Bob Smith's record should display.

9. In the Description field of row 3, enter: Does the benefits orientation field contain a checkmark next to "Benefits Orientation Notification sent"?
10. Note that the type will automatically change to a Verification Point because your description ended with a question mark. If your type did not automatically change, click in the Type field in row 3 and it will change to a blue checkmark.  
After entering the preceding information the Manual Test window should look similar to the one shown here.



	Type	Note	Description	Expected Result	Expected Result File
1	<input type="radio"/>		Click on the benefits orientation link on the main window.	The list of new hires should display.	
2	<input type="radio"/>		Open the test record, Smith, Bob.	Bob Smith's record should display	
3	<input checked="" type="checkbox"/>		Does the benefits orientation field contain a checkmark next to "Benefits Orientation Notification sent"?		

11. Select **File > Save** and enter the name Verify Automatic Notification.
12. Select **File > New** to create one more test script and enter the following information:
  - In the Description field of row 1, enter: Click on the benefits orientation link on the main window.
  - In the Expected Result field of row 1, enter: The list of new hires should display.
  - Use the <Enter> key to move to the next row.
  - In the Description field of row 2, enter: Open the test record, Jones, Anne.
  - In the Expected Result field of row 2, enter: Anne Jones' record should display.
  - In the Description field of row 3, enter: Click on the manual notification radio button and select the next available orientation from the list.
  - In the Expected Result field of row 3, enter: The selected date/time should appear next to the benefits

orientation.

- In the Description field of row 4, enter: `Press the Send button.`
- In the Description field of row 5, enter: `Wait 2 minutes, and then open up the test email account for Anne Jones.`
- In the Description field of row 6, enter: `Is there an email from "Benefits Scheduler" with the date/time selected?`
- Make sure there is a check mark next to the question. If not, click in the Type field to change it to a check mark.

13. Select **File > Save** then type the name: `Verify Manual Notification.`

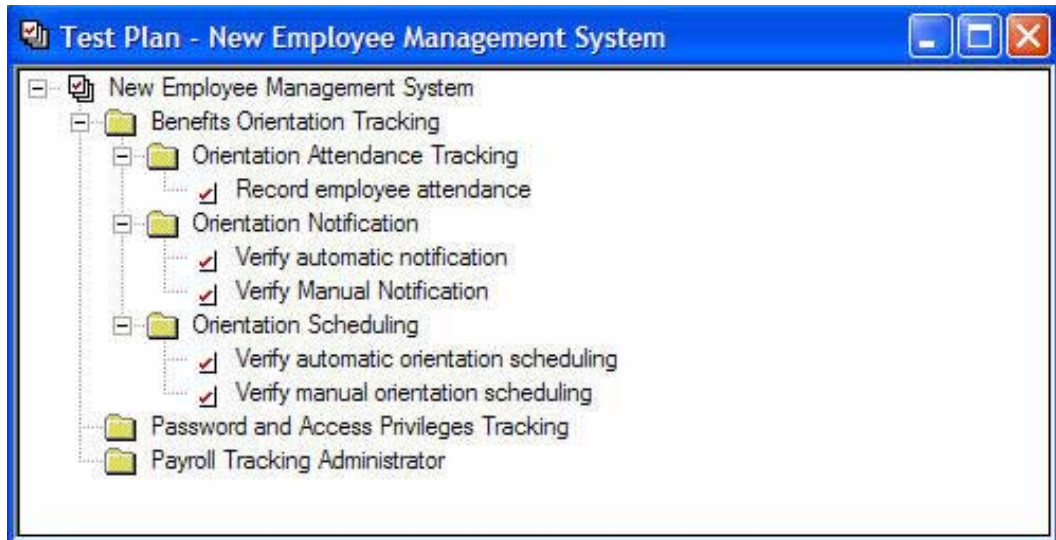
You have now created two manual test scripts. The last step to complete your test cases is to attach the test scripts to their associated test cases.

## Attaching test scripts to test cases

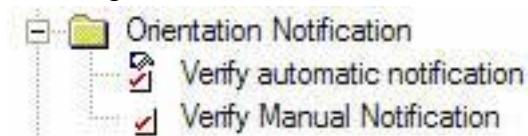
Now that you have created manual test scripts, you can associate them with their test cases. This is referred to as *implementing the test case*. When the test case is selected for execution, Rational TestManager runs the manual or automated test script attached to the test case. Although test scripts can be executed by themselves (without being attached to a test case), only a test case can be linked to a requirement. Therefore, to report results against requirements, it is necessary to attach your test scripts to test cases first, and then execute the test cases.

Complete the following steps to attach the two manual tests to two test cases:

1. Open the test plan as shown in the following image.

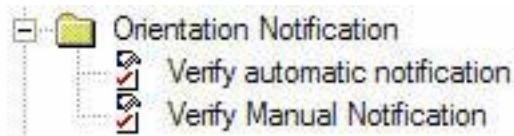


2. Right-click on the test case **Verify automatic notification** and select **Properties**.
3. Click on **Implementation**.
4. Click **Select** in Manual Implementation.
5. In the Select Script window, click on the script **Verify automatic notification** and then click **OK**.
6. Click **OK** to close the Test Case Properties window. The test plan should have a "pointing finger" icon next to this test case as shown in the following image. This indicates that the test case has been implemented with a manual test script. If an automated test script was used, you will see a "gear" icon.



7. Repeat steps 2 through 6 using the test case Verify Manual Notification and attach the test script Verify Manual Notification.

Both test cases now have manual test implementations and the test plan should appear as follows.



Once a test case has been implemented, it can be executed as a stand-alone test case or placed in a test suite along with other test cases for batch (grouped) execution. There are also standard reports available to help you determine which test cases have been implemented and which have not.

In the next section, you'll learn how to execute test cases and analyze their results.

---

## Section 8. Execute the test and analyze results

### Methods of test execution

There are many ways to execute test cases. If there is just one test case that you want to execute, you can run it directly from the Test Plan. To do that, right-click on a test case (or test case folder) in the test plan and select **Run**. If the test case has an automated test, the test begins. If the test case has a manual test script, it displays a window that prompts you to perform each of the steps documented in the manual test and to record the results.

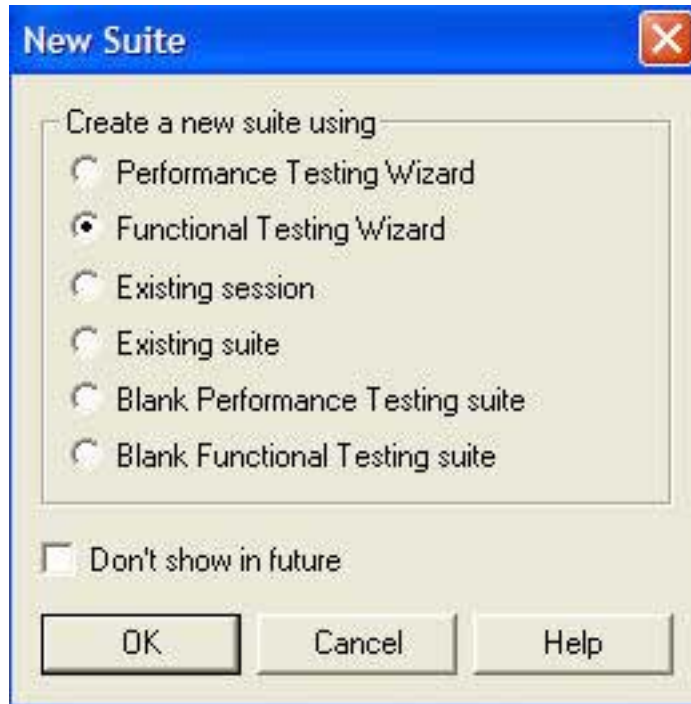
This mode is good for quickly executing a small number tests ad-hoc. A more efficient method of test case execution is to create a suite that contains the test cases that you want to execute. A suite can be saved and executed again at any point in time. Suites also provide additional capabilities such as distributing tests to remote workstations for execution and allowing you to use functions such as randomization, synchronization, delays, groups, and so forth.

### Creating a test suite

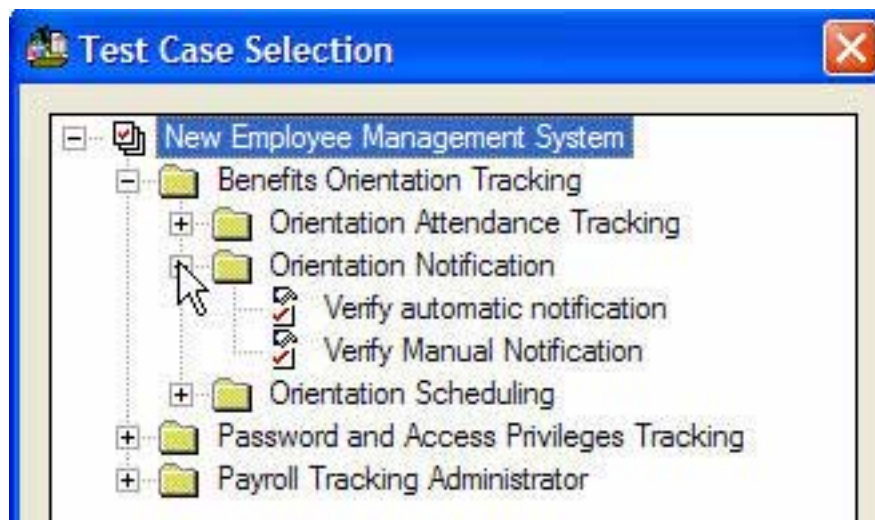
Complete the following steps to create a test suite that contains the two test cases that you have implemented with manual tests.

1. From Rational TestManager, click **Execution** in the lower left corner.
2. Right-click the Suites folder and select **New Suite**.

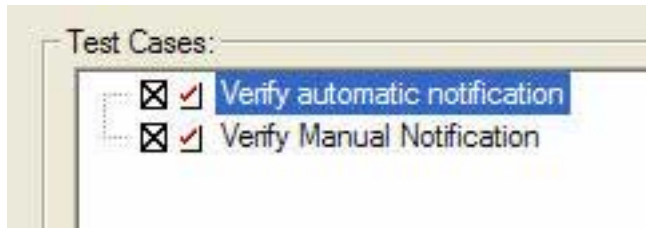
- The following window opens.



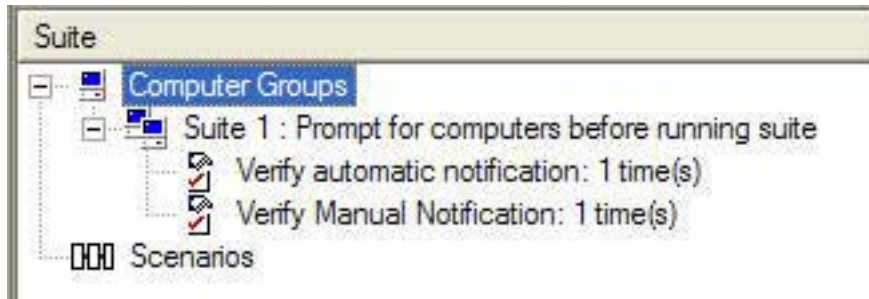
- Select **Functional Testing Wizard** and then click **OK**.
- Select **Test Cases** from the Test Plan.
- Expand the Test Plan and use <CTRL> click to select the test cases **Verify manual notification** and **Verify automatic notification** and then click **OK**.



7. Select **Next** on the resulting window.



8. In the window Functional Suite Wizard -- Step 2 click **Next** because you do not need to add any stand-alone scripts. The scripts you want to execute are already attached to the test cases that you have selected.
9. Click **Finish** on the Functional Suite Wizard -- Step 3 window.
10. The following image shows the resulting suite.



This basic test suite executes the two test cases in the order specified on your local machine. However, a suite can, and generally will, contain more complex conditions. Some examples include assigning test cases to execute on remote machines, grouping test cases into scenarios that represent a business function, inserting delays or synchronization points to control timing and synchronization of different tests, and randomization of test case selection. A test suite can contain many types of entries. In this example, you have two test cases; each runs a manual test script. However, the suite could also contain test cases that have automated or custom-built tests and stand-alone test scripts. Suites are also used to execute both functional and performance tests.

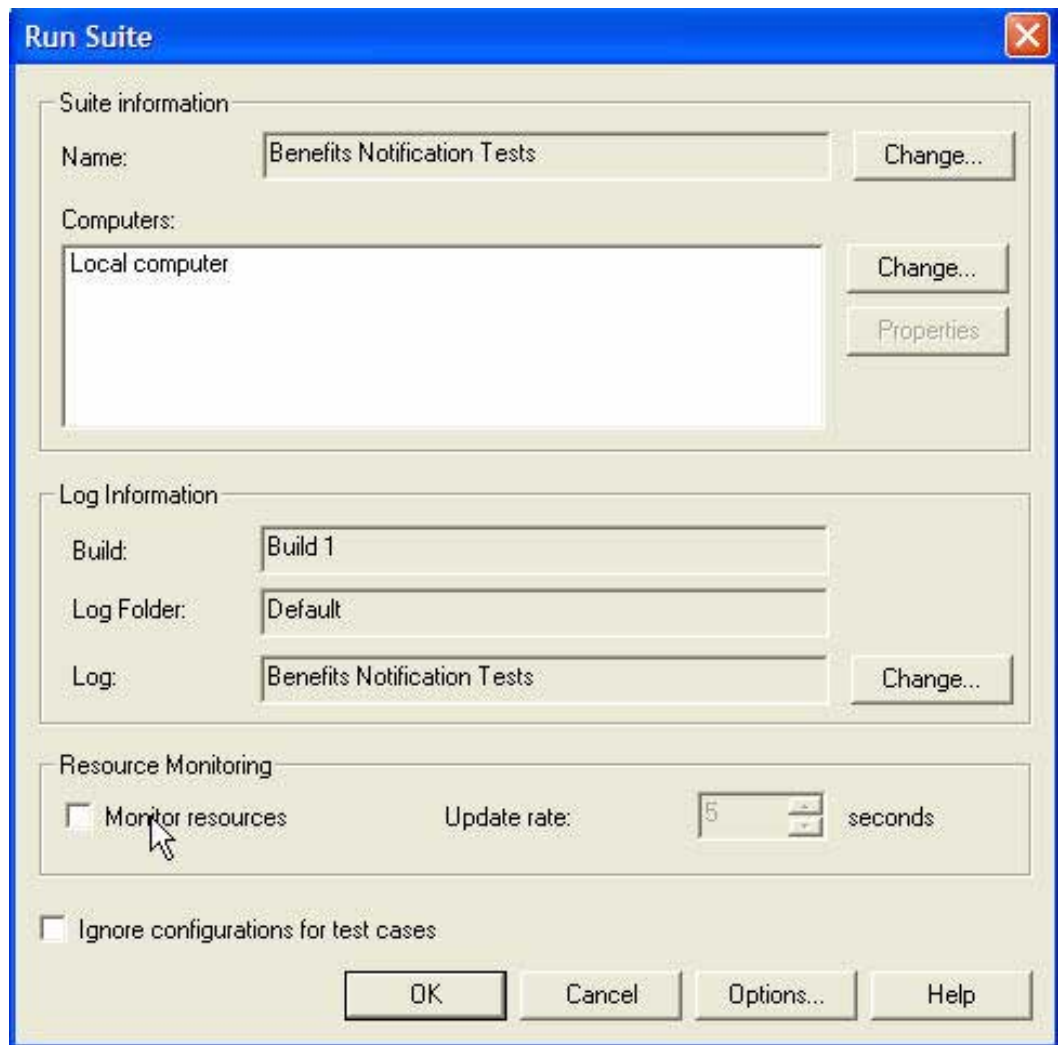
## Executing a test suite

Once a suite is created, you can save the suite if you want to run it again at a later point. You can also make changes to saved suites.

1. Select **File > Save** then enter the name `Benefits Notification`

Tests.

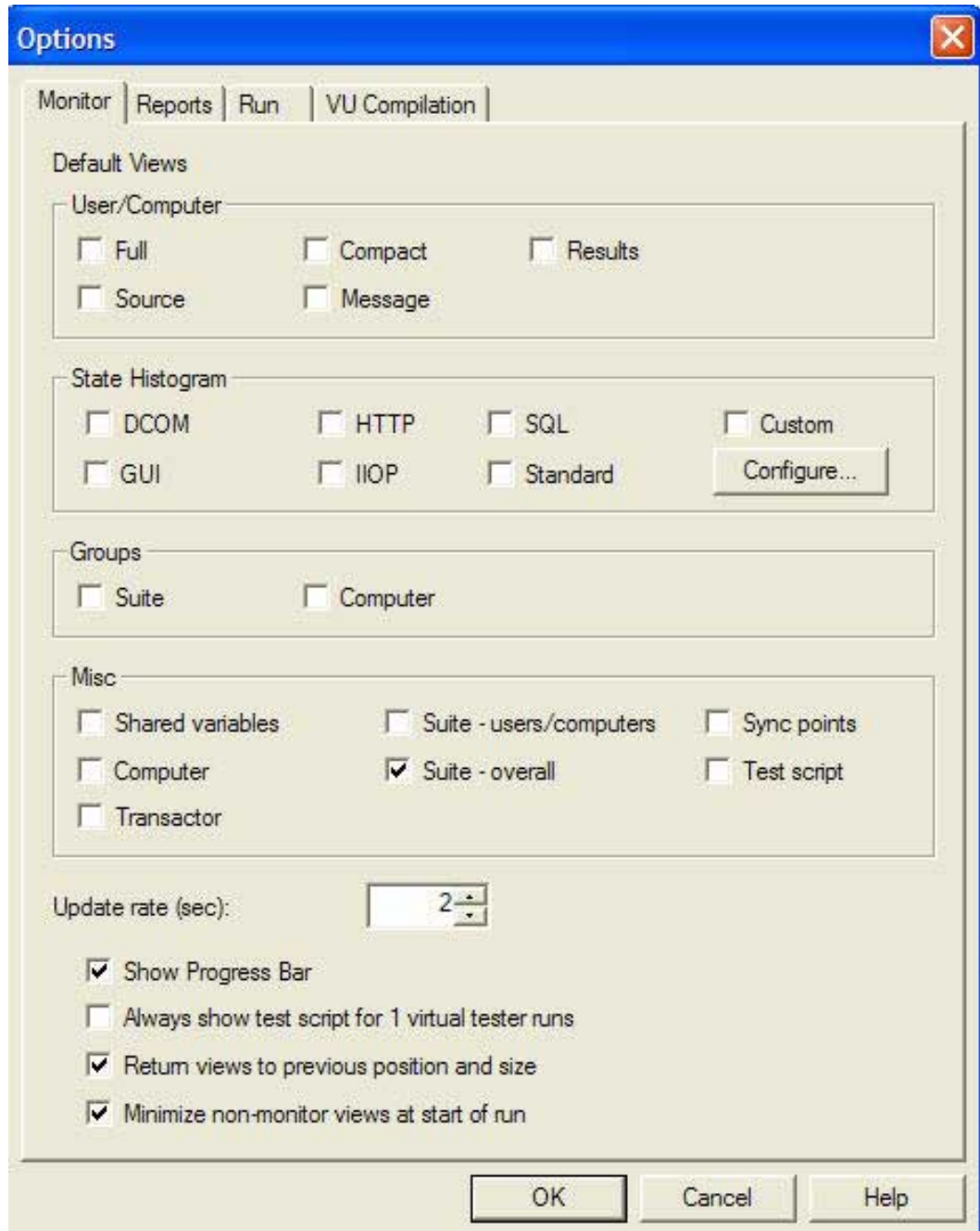
2. To run the test suite click on the toolbar button that looks like a running man.  
The following window opens.



When a suite is executed, you must specify the application build that you are testing, the log folder for storing results, and the name for the log file. In this tutorial, using the default choices is sufficient. In a real project, define your own build numbers, log folders, and log names so that you can track the results of your tests against different builds of the application.

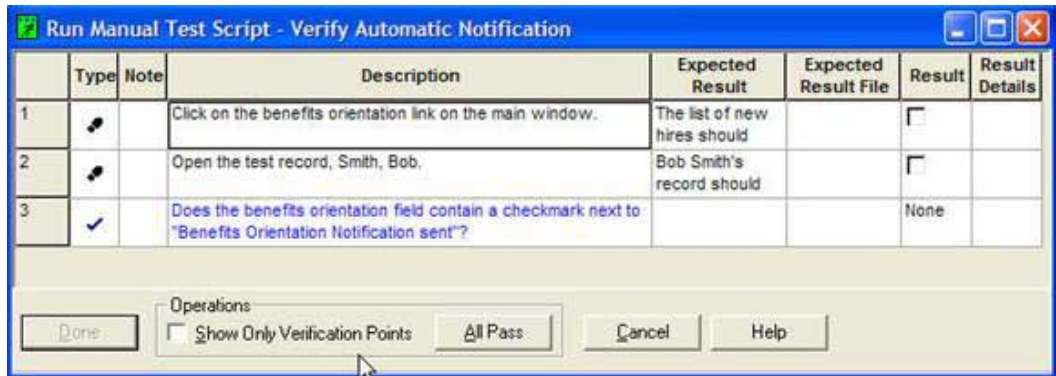
3. Click the Options button.

4. Make sure your options are set as the following image shows. There are four default view groups. Only the option Suite-overall in Misc should be selected. Don't change the default selections for the four check boxes at the bottom of the screen.



5. Click **OK** on the Options window and then click **OK** again to begin running the suite.

The first manual test opens as shown here.



Because you do not actually have the application to test, you have to pretend that you are completing the test directions. For each step or verification point, you can enter a detailed description of the results and/or attach a file (such as a screen shot) in the Result Details column. For each test step, check the box in the Result column when the step is complete. For verification points, select the appropriate pass/fail results in the Result column.

6. Check the Result check box for steps 1 and 2 and set the verification point result to **Pass**.
7. Click **Done** and the second manual test opens.
8. Check the Result check box for steps 1 through 5 and set the verification point result to **Fail**.
9. In **Result Details** next to the verification point, enter the text: The email was sent but didn't contain the date or time of the orientation.
10. Click **Done**. After a moment the test log with results is displayed.

## Analyzing test results

A test log with the results of the test is displayed as shown. You should have one Pass and one Fail. If a test script contains multiple verification points, all of them must pass in order to have a pass for the entire test case. One verification point failure results in a failure for the entire test case. This is true regardless of whether you are using manual or automated test scripts in your test cases.

Name	Actual Result	Interpreted Result	Promoted
Verify automatic notification	Pass	Pass	<input type="checkbox"/>
Verify Manual Notification	Fail	Fail	<input type="checkbox"/>

1. Click the Details tab at the bottom of the screen.
2. To make the log easier to read, click the toggle **Activate the Workspace window** on the toolbar. This hides the workspace on the left and provides more space to view the log.
3. Expand the suite results and the failed test case (Verify Manual Notification) so that you can see the details as shown in the following window.

Event Type	Result
[-] Suite Start (Benefits Notification Tests)	Fail
[-] Computer Start (Suite 1 [1])	Fail
+ TestCase Start (Verify automatic notification)	Pass
[-] TestCase Start (Verify Manual Notification)	Fail
[-] Script Start (Verify manual notification)	Fail
Manual Step (Click on the benefits orientation link on the main wi...)	Completed
Manual Step (Open the test record, Jones, Anne.)	Completed
Manual Step (Click on the manual notification radio button and s...)	Completed
Manual Step (Press the Send button.)	Completed
Verification Point (Is there an email from "Benefits Scheduler" wit...)	Fail
Script End (Verify manual notification)	Fail
TestCase End (Verify Manual Notification)	Fail
Computer End	Fail
Suite End (Benefits Notification Tests)	Fail

Here you are able to see each step that was completed and which verification points in the test script failed. In this log, the failure was from the single verification point in the script (Is there an email from....)

4. Right-click on the failed Verification Point line in the log and select **Properties**.
5. In the Event log window you can scroll through the details and should be able to see the results text that you entered.  
In the Details tab you can see the cause of a failure. For manual tests, the cause is generally apparent if you were the person who ran the test and entered the results. For automated tests, that generally run unattended, use the details in the log to determine the cause of a failure and the required course of action. If this failure was the result of an automated

test execution, automatic comparators is displayed for each verification point and highlight the differences between the expected and the actual behavior of the application.

6. Close the Log Event window.  
If you also use Rational ClearQuest for defect and change tracking, you can right-click on any line in the log and submit a defect directly to the database. The details of the defect is automatically logged, along with the test case name, description, and other important information to help a developer reproduce and fix this problem. In this tutorial however, you did not install and set up a ClearQuest project so you can not submit defects.
7. Click on **Test Case Results** at the bottom of the window.
8. Now that you have evaluated the results, you can promote them so that they are available for reporting. Click on both boxes in the Promoted column.
9. Close the test log using the close button in the upper-right corner of the log window.
10. When prompted if you want to save the test case results, select **Yes**.
11. Click on the toggle **Activate the Workspace window** on the toolbar to redisplay the workspace.

You have completed analysis of the test results. Next, learn how to run reports to monitor your test progress.

---

## Section 9. Monitor your progress

### Progress reporting

As you create and execute tests against each successive build of your application, you will want to keep track of your progress. The reports provided in Rational TestManager allow you to track the progress of defining test cases for your requirements, creating and associating test scripts with your test cases, and executing your test cases. In this section you will run two reports that will provide information about the progress of testing.

Rational TestManager has several predefined reports that you can run to track test development progress and results.

If you look at the Analysis tab, you will see a list of all the reports available. You will be working with the reports in the group called Test Case reports.

## Running a test case distribution report

One predefined report provides information about your progress on creating and implementing test cases.

1. Click on **Analysis** in the lower left corner.
2. Expand the folder labeled **Test Case Distribution**.  
These reports provide information on test progress against test inputs (requirements) or against the test plan (test cases). The last two reports in this group are used to provide information about test cases that are affected by modified requirements. You can use the reports out of the box, modify them, or create your own. For this tutorial, you will use the predefined reports.
3. Right-click on the report titled **Test Input Development Coverage** and select **Run**.

When the report displays you will see a list of test requirements (from either RequisitePro or Microsoft Excel) in the first column of the report, similar to the one that follows.

(Note: If you do not see the information as shown here, close then reopen your Test Inputs and Test Plan windows and select **Refresh All**.)

+ [ ] FEAT1 Automatic benefits orientation scheduling	4	0	0
+ [ ] FEAT2 Manual benefits orientation scheduling	1	0	0
+ [ ] FEAT3 Automatic employee notification of orientation	1	1	100
+ [ ] FEAT4 Manual employee notification of orientation	1	1	100

The second column displays the number of test cases planned for each input. The third column contains "Implemented Test Cases", which refers to test cases that have manual or automated test scripts associated with them. You should see a "1" next to "Automatic employee notification of orientation" and "Manual employee notification of orientation". In the final column is the percentage of inputs

implemented. The two inputs that have test cases and test scripts should show 100%. This number then rolls up to the top level group of inputs.

## Running a test case results report

The next report that you will generate will report on test results rather than test planning and development.

1. Expand the folder **Test Case Results Distribution**. You should see two reports.
2. Right-click on the report **Test Plan Execution Coverage**.
3. In the Select Test Logs window expand **Build 1** then the Default folder.
4. Click on the test log then click the > button to move it to the selected Test Logs groups.
5. The following report will display.

	Implemented Test Ca...	With Results	Passed Test Cases	Failed Test Cases
[-] New Employee Management System	2	2	1	1
[-] Benefits Orientation Tracking	2	2	1	1
[+] Orientation Attendance Tracking	0	0	0	0
[+] Orientation Notification	2	2	1	1
[+] Orientation Scheduling	0	0	0	0
[-] Password and Access Privileges Tracking	0	0	0	0
[-] Payroll Tracking Administrator	0	0	0	0

6. Expand the folder **Orientation Notification**.

You will see test case results in the following report.

[-] Orientation Notification	2	2	1	1
[-] Verify automatic notificatio	1	1	1	0
[-] Verify Manual Notificatic	1	1	0	1

The results of two test cases display in this report. One test case passed and the second test case failed. This is one example of the many pre-built reports available to assist you in monitoring and evaluating the testing progress. You may also create custom reports to generate the information that will meet your team's reporting and data collection standards.

---

## Section 10. Wrap-up

### Summary

In this tutorial, you have explored a number of capabilities provided by Rational TestManager to manage your testing progress. However, this tutorial has just touched the surface of Rational TestManager's capabilities. The key benefit of TestManager is that it provides your entire team a central tool to plan, manage, execute, and analyze the progress of testing. Rational TestManager can plan test cases against your application's requirements, allow you to define test cases to validate requirements, then execute manual or automated test scripts and store the results of those test executions. A well-organized testing effort ensures that you have a comprehensive test plan and factual information to make decisions about the fitness for use of your application.

## Resources

- [Participate in the discussion forum for this content.](#)
- [Using Rational TestManager to expedite test plan reviews](#)
- [Using Rational TestManager to report test coverage and progress](#)
- [Extending IBM Rational TestManager reporting](#)
- [Integrated test and defect management](#)
- [Traceability and impact analysis](#)
- [Overview of TestManager](#)

## About the author

### Sandra Wilkey

Sandra Wilkey is a technical marketing specialist focusing on IBM Rational software quality tools. Sandra has been with the IBM/Rational organization for over 12 years and has spoken at numerous user conferences and industry trade shows. Sandra most recently spoke at the 2004 Rational Software Developer User Conference in Dallas, Texas, where she gave a well-received talk on tips and tricks with Rational TestManager.