

Agilent Test Automator 2.1

Getting Started Guide



Agilent Technologies

Notices

© Agilent Technologies, Inc. 2004, 2006

No part of this manual may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Agilent Technologies, Inc. as governed by United States and international copyright laws.

Manual Part Number

W1130-90012

Edition

Second Edition, May 2006

Printed in USA

Agilent Technologies, Inc.
815 14th Street SW
Loveland, CO 80537

Microsoft and Windows are US registered trademarks of Microsoft Corporation.

Warranty

The material contained in this document is provided “as is,” and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

Restricted Rights Legend

If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as “Commercial computer software” as defined in DFAR 252.227-7014 (June 1995), or as a “commercial item” as defined in FAR 2.101(a) or as “Restricted computer software” as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies’ standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as

defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

Safety Notices

CAUTION

A **CAUTION** notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a **CAUTION** notice until the indicated conditions are fully understood and met.

WARNING

A **WARNING** notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a **WARNING** notice until the indicated conditions are fully understood and met.

1	Introduction	1
	Overview	2
	Installation	12
	The Product Key	12
	System Requirements	13
2	The Sequence Definition	15
	Creating a Sequence Definition	16
	Variables	16
	Initialize Sequence	21
	Main Sequence	24
	Routines	33
	Trace Tab	37
3	Results Management	39
	Results Management	40
	Columns	40
	Results Presentation	46
	Graphs	46
A	Appendix A	51
	Test Automator Configuration File	52
	The Contents of the Configuration File	53
B	Appendix B	55
	Numeric Formats	56

Contents

Introduction

Overview 2

Installation 12

System Requirements 13

Overview

Test Automator simplifies the creation of R&D tests for design verification. It is used to process tests and generate measurements. These measurements are compared against the specifications for a Device Under Test (DUT) to see if the DUT meets specifications.

Test Automator accomplishes this through a rich set of features. It has a graphing capability with three standard graphs (normalized progress chart, limits chart, and histogram); a trace ability that makes it easy to see exactly what is happening in the sequence definition and when; and a powerful sequencing engine. The sequencing engine supports looping structures; callable routines; user-defined variables; debugging tools; export of results to Excel, csv files, or text files; re-usable routines; and DLL importing. These features are all driven from a user-friendly interface.

For demonstration purposes, open the sample file DemoSequence. You can find this file at \Program Files\Agilent\Test Automator 2.1\Samples\Sequences.

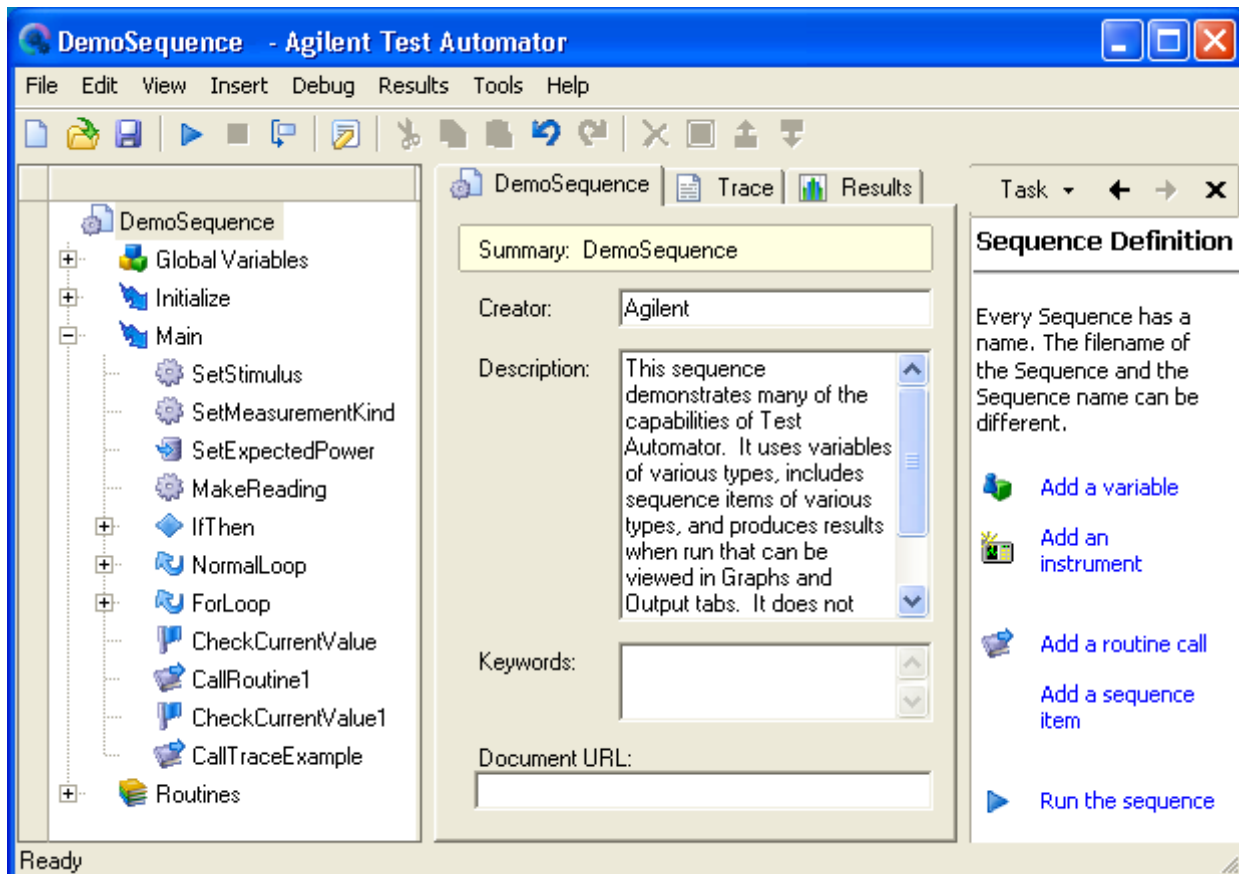


Figure 1 The User Interface for the Test Automator

The user interface for the Test Automator has four key parts:

- The Task Guide on the right side of the application
- The main menu and icon toolbar across the top of the application
- The Sequence Tree on the left side of the application
- The Details Tab, Trace Tab, and Results Tab in the center of the application.

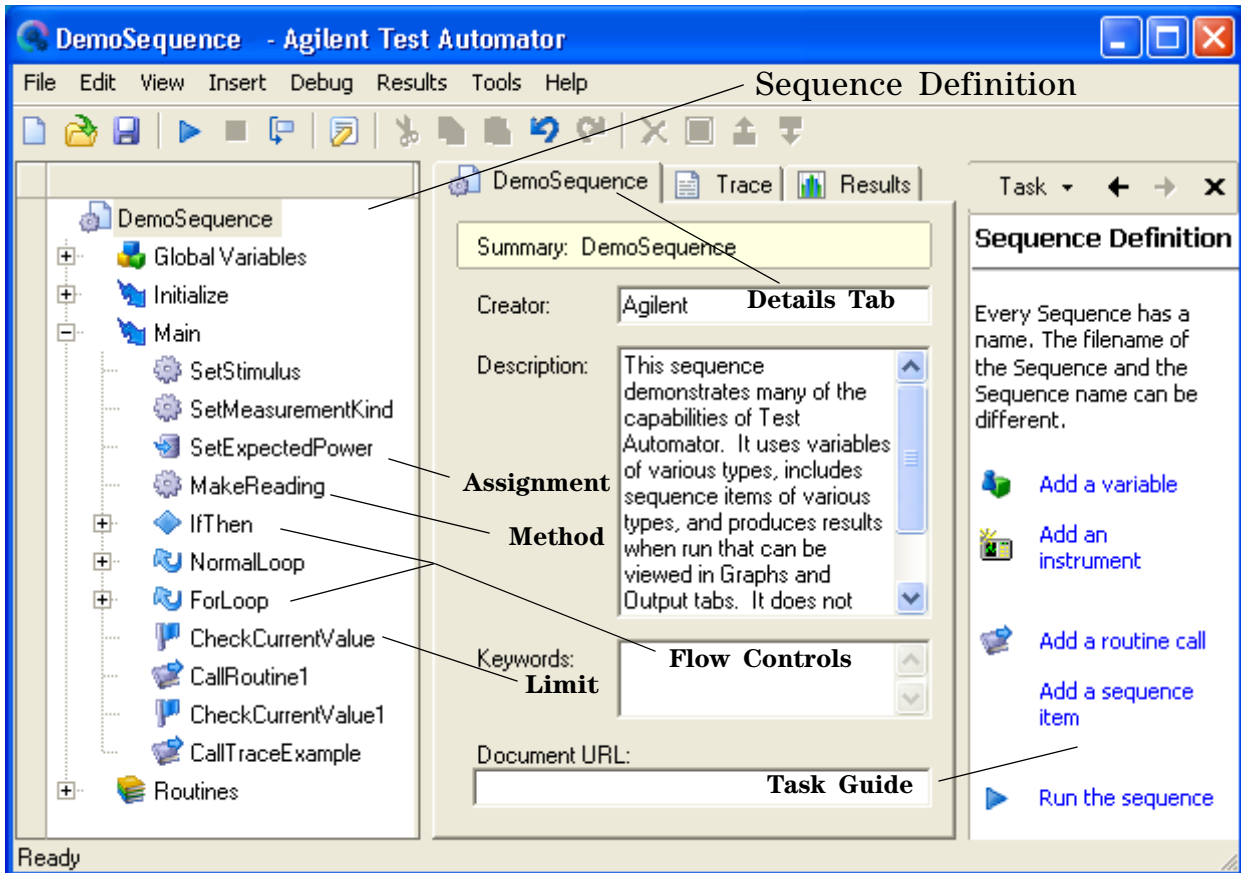


Figure 2 Sequence Tree Described

Task Guide The Task Guide is a true helper. As you use Test Automator, the Task Guide provides tips and tricks and actually performs commands that you pick from the menu. If you are stuck on something or want to use a shortcut to get something done, turn to the Task Guide first. It can save you valuable time.

Sequence Tree The Sequence Tree displays an overview of the Sequence Definition. A Sequence Definition is made up of Variables, the Initialize Sequence section, the Main Sequence section, Looping and Logic Flow Controls, Method Calls, DirectIO, Assignments, Groups, Result Checks, and Routines. The terms variable, method, and assignment are common programming terms that need no definition here. The other terms do require brief descriptions.

The Initialize Sequence and Main Sequence sections are both unique. The Initialize Sequence section is primarily used to initialize object variables and instrument variables you define in the Global Variables section of your Sequence Definition. These only need to be initialized once when the sequence definition runs the first time. There is no need to do this again, so the Initialize Sequence section runs only once during the run of a Sequence Definition. See [“Initialize Sequence”](#) on page 21

The Main Sequence section is where most sequence runs start. It contains all of the calls to routines and methods, assignments, results publish, results check, DirectIO, groups, and flow controls that make up your sequence.

The four Flow Controls used in the Test Automator are the four control structures, For loops, Try/Catch, If/Then, and Group statements. Flow controls guide the execution of sequence items and may occur in other sequences, Routines, or other flow controls.

Test engineers use limits to determine whether the DUT has met the design specification. You can check measurement results against one or two limits, and then publish these results and the judgment of the results against the limits for analysis, reporting, and graphing to the data results table.

There are two kinds of Routines, Local and External. Either routine can be called multiple times from within a Sequence Definition. The developer creates the local routine in the Routines area and then calls it from within Main Sequence. External routines are similar except they have been defined externally in different and unique sequence definitions and are being called from within Main Sequence.

The Details Tab With the Details Tab selected, choose any item from the Sequence Tree to see the details about that item. In [Figure 3](#) on page 7, the Variables item was selected and a listing of all currently declared variables appears on the Details Tab.

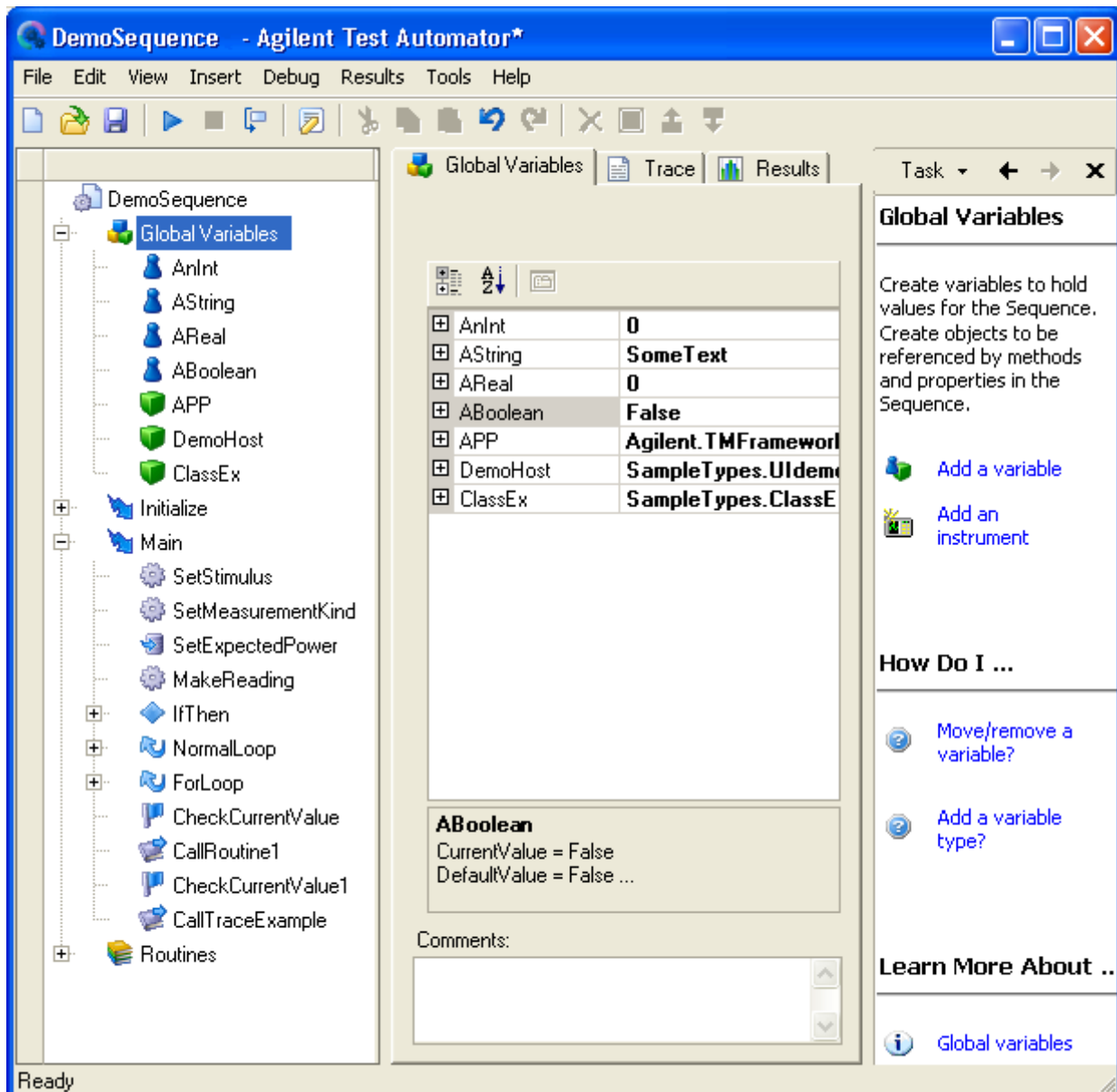


Figure 3 Displaying the Variables Listing

The Trace Tab The Trace Tab is used for tracing sequence activity. As you can see in [Figure 4](#) on page 9, when the DemoSequence was run, all sequence run activity was logged and made available as a debugging tool on the Trace Tab. In fact, any Debug or Trace information written by code that is called by the Test Automator appears in this view.

There are also grey check marks and green balls next to the sequence items that pass. A red, square x appears next to sequence items that fail.

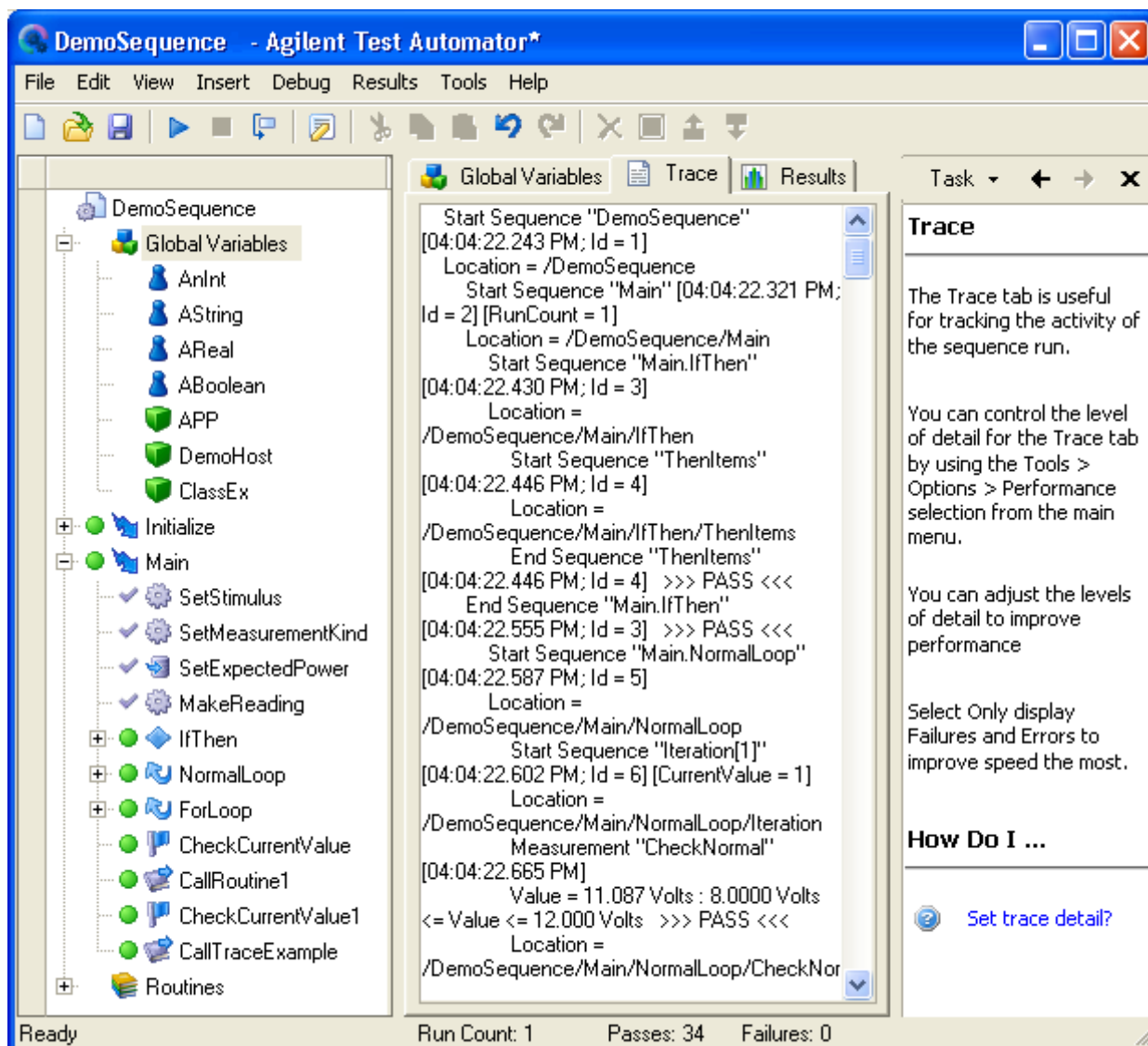


Figure 4 The Trace Tab

The Results Tab The Results Tab provides three snapshot graphs, some basic statistics, and a powerful data management table. The graphs and statistics are intended to provide very basic data review. The table provides tools and options to adjust the view of the data in the table. It is populated through result check items that publish data to the table, which in turn feeds the graphs.

You can pick which records are included in the graph output by selecting from the column of check boxes on the right side of the Sequence Tree (the Table column). By unselecting and selecting these check boxes, you can see how your graph would look by excluding or including results.

The screenshot displays the 'DemoSequence - Agilent Test Automator' window. The 'Results' tab is active, showing a 'Normalized Progress' graph and a 'Results Table'. The graph plots a blue line representing progress over time, with a red upper limit and an orange lower limit. The x-axis has markers at 17, 40, and 67. Below the graph is a 'Results Table' with the following data:

Use	Time	Value	Status
<input checked="" type="checkbox"/>	04:04:22.665	11.087 Volts	Pass
<input checked="" type="checkbox"/>	04:04:22.743	10.519 Volts	Pass
<input checked="" type="checkbox"/>	04:04:22.758	9.8493 Volts	Pass

At the bottom of the window, the status bar shows 'Run Count: 2', 'Passes: 34', and 'Failures: 0'. On the right side of the Results panel, there are several interactive links: 'Select graph type', 'Select Progress report inputs', 'Configure graph', 'Change Normalized Progress content', and 'Save graph'.

Figure 5 The Graphs Tab

Installation

Before installing the Test Automator, review the included ReadMe information for the latest changes and updates to the application.

To install Test Automator:

- 1 Test Automator is installed with T&M Toolkit.
- 2 Insert the T&M Toolkit CD-ROM into your CD-ROM drive.
- 3 The setup process should start automatically. If it does not:
 - Go to **Start > Run**.
 - Enter [Your CD-ROM drive letter]:\setup
 - Press **Enter**.
- 4 Follow the prompts until the installation is complete.

The Product Key

Test Automator is activated with a Product Key. This key appears on the certificate that you received with your order. (If you are evaluating Test Automator, use the Product Key **eval**.)

System Requirements

Processor

Minimum 600 megahertz processor. A faster processor is always better.

RAM

256 megabytes of RAM.

Disk Space

30 megabytes of available disk space.

Monitor

Minimum 800 X 600 resolution, with at least 256 colors.

Operating Systems

Microsoft Windows® 2000 (SP4 or later) or Windows XP Professional or Home (SP2 or later).

Microsoft .NET Framework 2.0. (.NET Framework 2.0 is installed with Visual Studio 2005. It can also be installed by running the Windows Update service from Internet Explorer, or downloaded directly from www.microsoft.com.)

Other

Agilent IO Libraries Suite 14.1 or later (included with T&M Toolkit).

1 Introduction

2 The Sequence Definition

Creating a Sequence Definition 16

Trace Tab 37

Creating a Sequence Definition

Using the following topics, you will create a Sequence Definition equivalent to the GettingStarted sequence that is distributed with the samples. You begin with the EmptySequence from the same sample set and build from there.

The intent of the Getting Started sequence is to introduce you to the various features of Test Automator.

To open the EmptySequence, go to the installation directory and open it from the samples\sequences sub-directory. This is typically, C:\Program Files\Agilent\Test Automator 2.1\Samples\Sequences.

Variables

Most of the variables you need for the GettingStarted sequence are already defined. To lead you through the exercise of deleting a variable, expand the Variables item by clicking on the plus sign next to it. Select AString by left clicking on it. Choose the red X that is circled in [Figure 6](#) on page 17 and delete the AString variable.

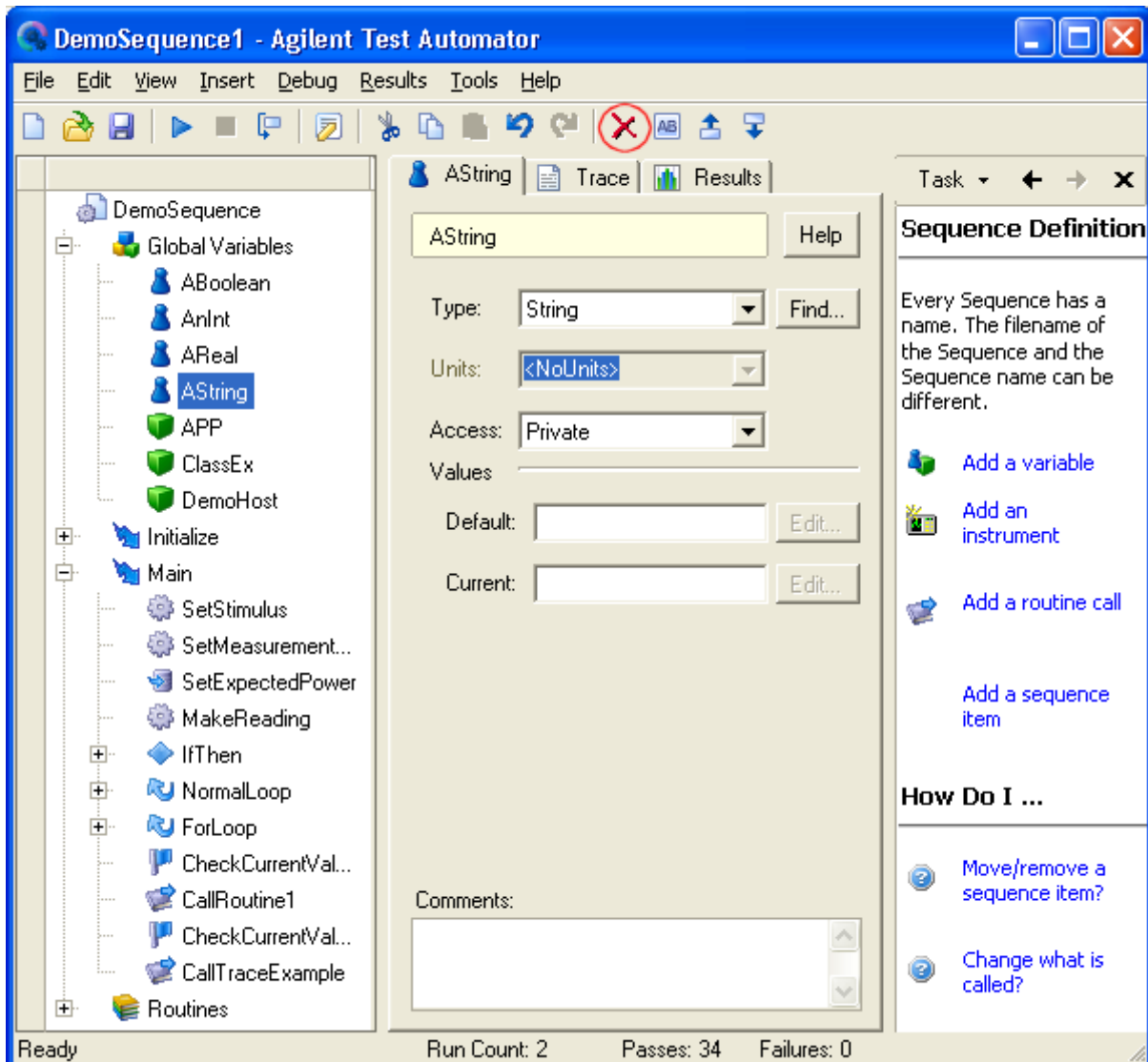


Figure 6 Deleting the AString Variable

2 Using Test Automator

Create a new AString variable by selecting **Insert > Variable** from the main menu. Name the variable AString and accept all of the other defaults. Now click **OK**.

Delete the AString variable using the context menu (highlight the AString variable and right click). Undo the deletion by using the undo icon (↶). Now that you have undone the deletion, redo the deletion by using the redo icon (↷). Finally, press the undo icon one more time to bring the AString variable back.

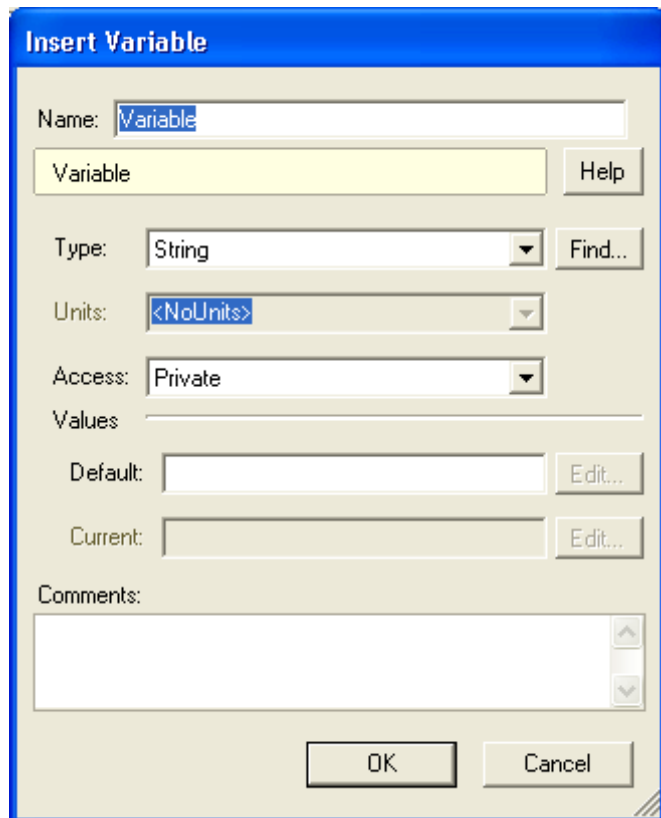


Figure 7 Inserting a New Variable, AString

The AString variable appears below whichever variable that is highlighted at the time. In fact, whenever you insert an item, it will appear below the highlighted item. With a variable highlighted, use the Up and Down menu icons shown circled in [Figure 8](#) on page 20, and alphabetize the variables.

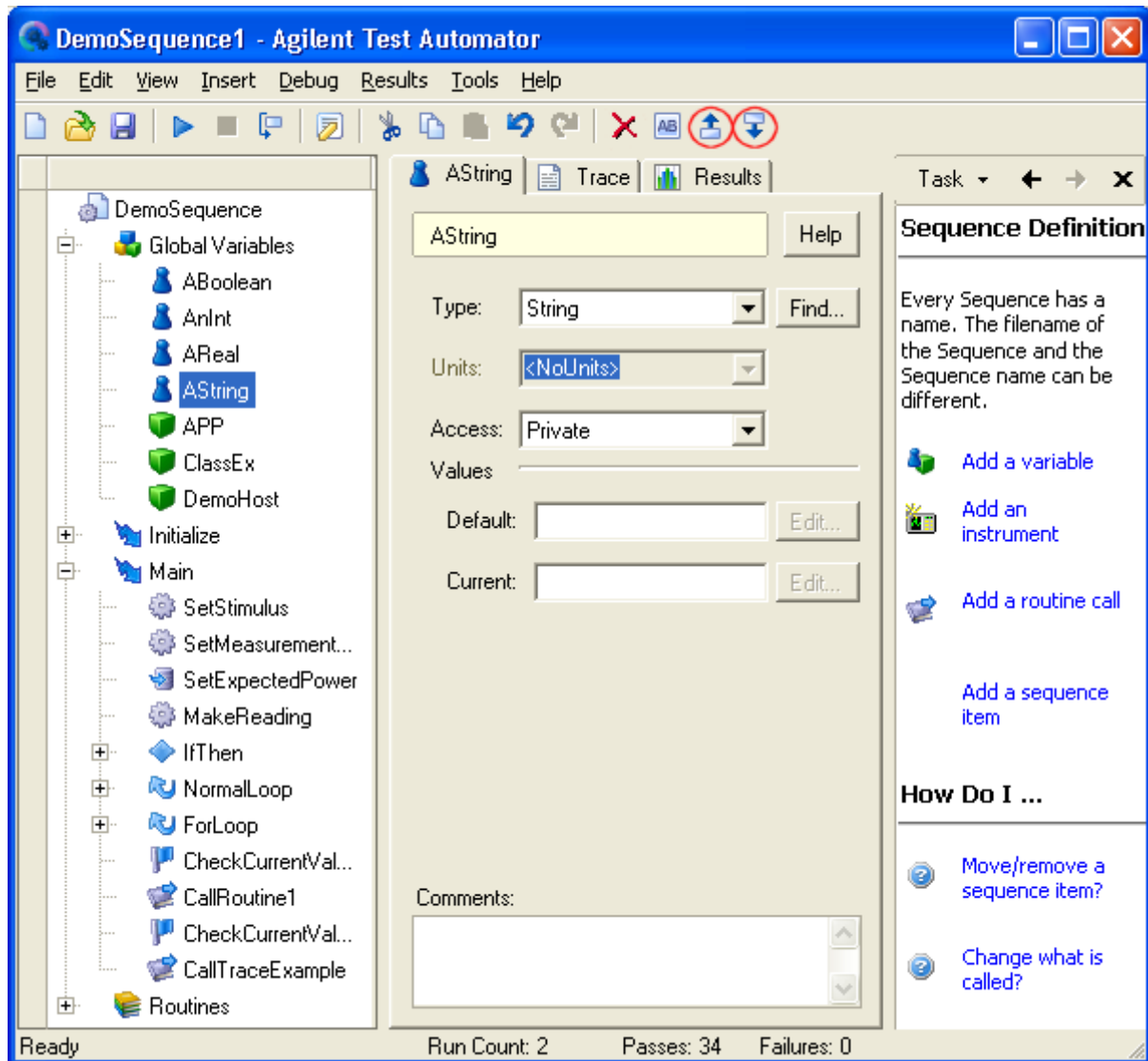


Figure 8 Empty Sequence with Sorted Variables

You need to check the default value on the AReal variable. Select it from the Sequence Tree and its details appear on the right. If the default value is not zero, edit the field now. Whatever you enter in the Default Value field must be compatible with the variable's data type, or you get an error message. Default values are saved with the Sequence Definition. The Current Value is set to the default value when a Sequence Definition is loaded and changes as the sequence definition is run. However, it is not saved with the Sequence Definition.

Variables that are Objects When you add a variable that is an object of a class, the Initialize Sequence section is automatically updated to contain a corresponding item to create that variable object. Sometimes this item has parameters that you need to set before running the sequence, but usually it does not.

Initialize Sequence

When you run your Sequence Definition the first time, the items in the Initialize Sequence section run before Main Sequence. Subsequent runs of the Sequence Definition *do not* run any of the items in the Initialize Sequence section unless there is a change in the Sequence Definition (an item is added or deleted). The initialize feature must be used carefully because it is only intended for the initial, first-run preparation of the sequence definition.

An example is useful. If you had a sequence definition you were going to run 50 times, and you were using the Initialize Sequence section to initialize an instrument as part of the process, it would only be initialized once. This initialization would occur before the first run of the sequence and would not happen again during all of the remaining runs.

While Initialize Sequence can contain several different things, it is most often used for objects of classes and instruments. Create another variable and name it AClassExample.

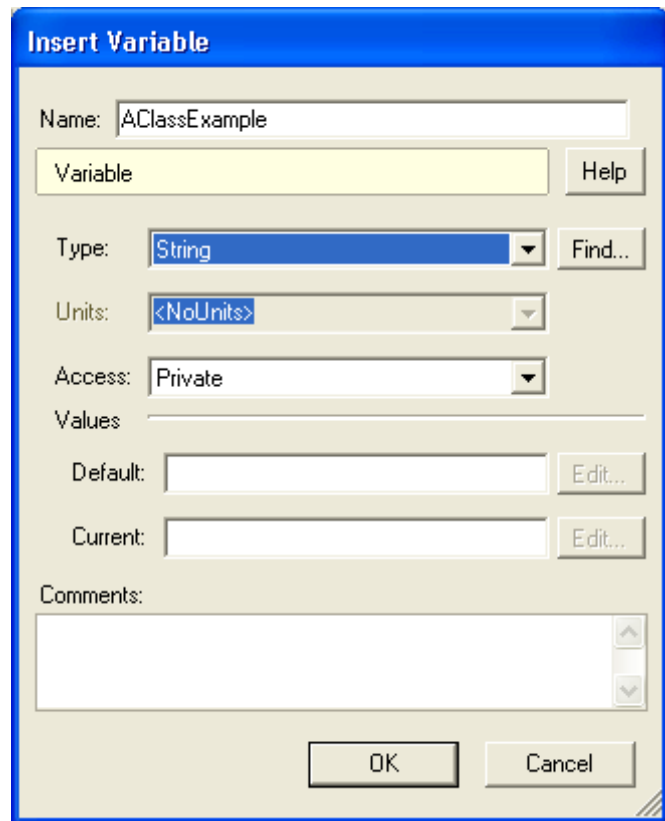


Figure 9 Creating the AClassExample Variable

You have to set the data type now. Open the drop-down box for Type and see if ClassExample is a default type. As you can see, it is not. You have to use the Find button to locate the ClassExample type.

Select the Find button. The general structure of the tree that you see is Assembly, Namespace, and then the class or interface, or enumeration, and so on. So, expand the Sample Types Assembly entry; now expand the Sample Types Namespace. You can see the ClassExample class displayed. Anything at this level of the tree can be used as a Test Automator type. (Alternatively, since you know the name of

the class you are looking for, you can use the **Find** box to locate it.) Select **OK** to complete the creation of the type and select it again to complete the creation of the variable.

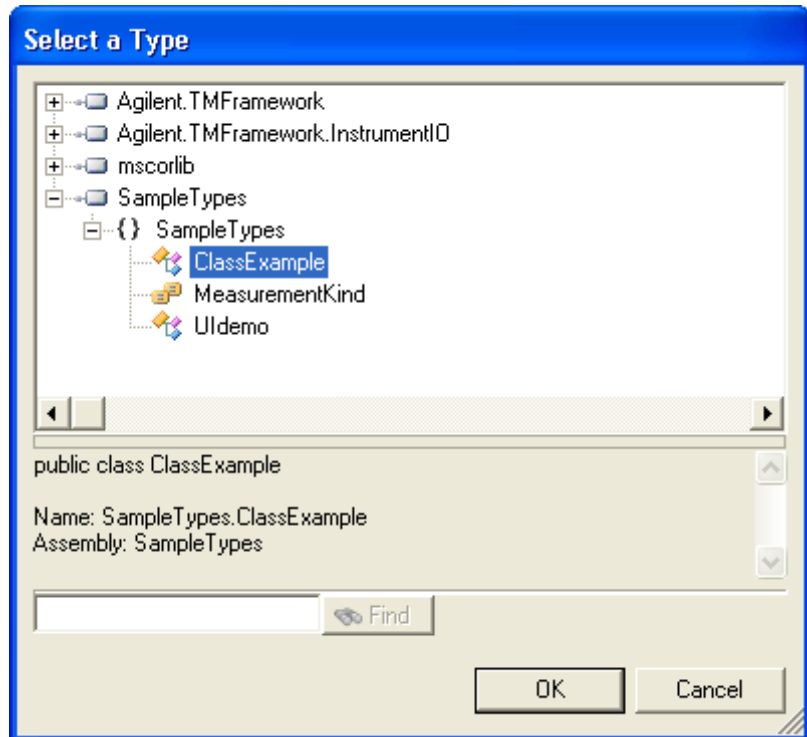



Figure 10 Class Example Type Selection

To match the GettingStarted sequence file, you need to rename this variable to ClassEx. Highlight the AClassExample variable and use the Rename icon () to rename it to ClassEx. You can also right click on the sequence item and select Rename from the context menu that appears.

Notice that Initialize acquires another entry. This is because the ClassEx variable is an object and must be initialized before the Main Sequence runs. When a variable is added,

Test Automator decides if it is a simple type that needs no initialization, such as integer, real, and boolean, or one that needs to be initialized, such as an object or instrument. If it needs to be initialized, an item is automatically added to the Initialize Sequence section.

When you add a variable that is an object of a class, the Initialize Sequence section is automatically updated. Objects have to be initialized before they can be used. While this is often not the case, some objects have parameters that have to be set. You may need to set these parameters in the Initialize Sequence section before the object will initialize properly or run.

Main Sequence

EmptySequence has nothing in it at present. You can start by adding method calls to the sequence.

- Collapse Global Variables and Initialize. Then highlight Main Sequence and select **Insert > Method Call** from the main menu. Name this method call SetStimulus. Select **@ClassEx** from the drop-down list of objects; leave the type as ClassExample; select **SetStimulus(Double)** from the drop-down list of methods; and enter 50 for the Output parameter. This sets an output of some kind at the level of 50.

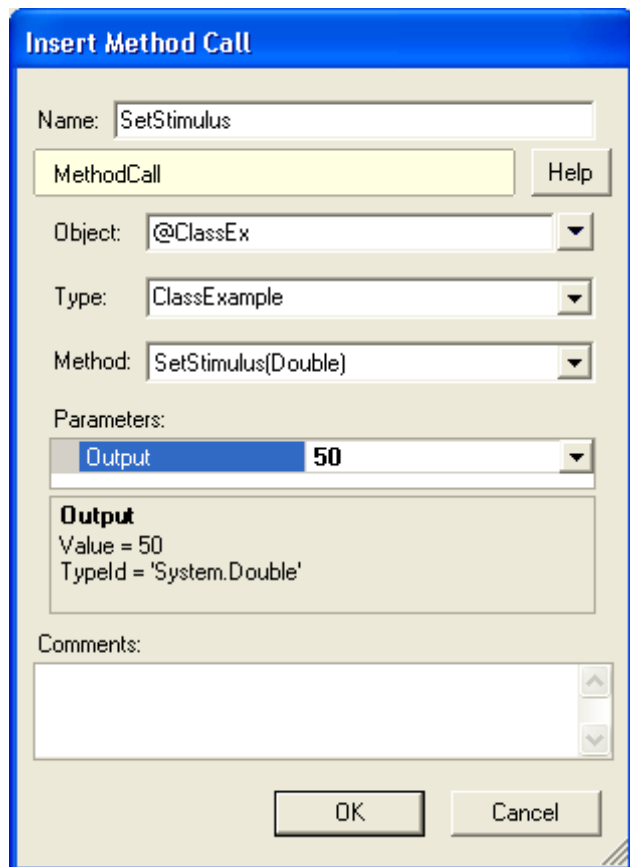


Figure 11 Insert SetStimulus Method Call

Now you need to set the kind of stimulus, and you want this to occur after the SetStimulus. Do this by adding a second method call, named SetMeasurementKind. Highlight **SetStimulus** and select **Insert > Method Call** from the main menu. Select **@ClassEx** from the drop-down list of objects; leave type as ClassExample; select **InitMeasurement(MeasurementKind)** from the Method drop-down list, and set Kind to **Ohms**.

Now you can experiment with an assignment statement. Highlight `SetMeasurementKind` and select **Insert > Assignment** from the main menu. Choose **@ClassEx** from the drop-down list in **Result**. To see the properties that are defined for `@ClassEx`, highlight `@ClassEx` and click the drop-down arrow to its right. `ExpectedPower` is the only property available, so select it from the list that appears. Set **Expression** to `5.4`. See [Figure 12](#) on page 26. Rename this assignment `SetExpectedPower`.

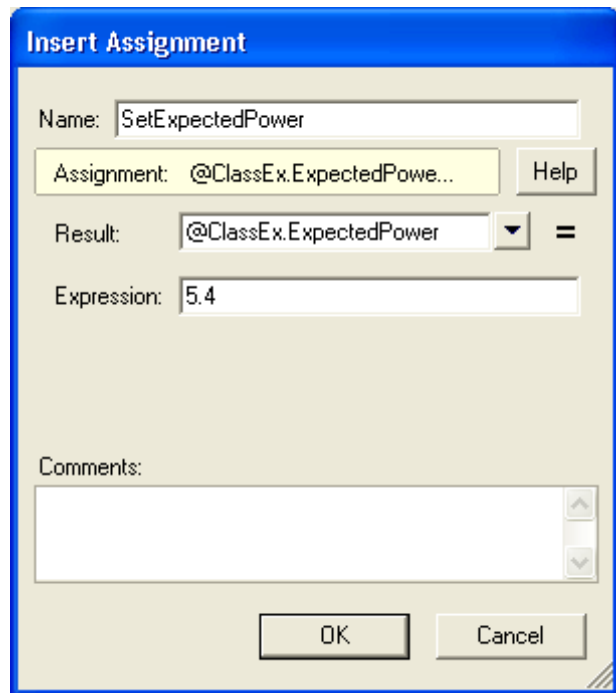


Figure 12 Assignment Statement Creation

Now add another method call that takes the measurement. Notice in [Figure 13](#) on page 27 that the method call has two parameters and a return value. Name the method call `MakeReading`. The Object is **@ClassEx** again, the Type is `Class Example`, and the Method is `MakeReading(Int64, Double&)`. Set **MakeReading_Return** equal to `@ABoolean`, the **Timeout** to

100, and the **Reading** to @AReal. Note you can set these values equal to the value of a variable by selecting that variable's symbol name. When MakeReading runs, it sets the value of @ABoolean based on the MakeReading return value.

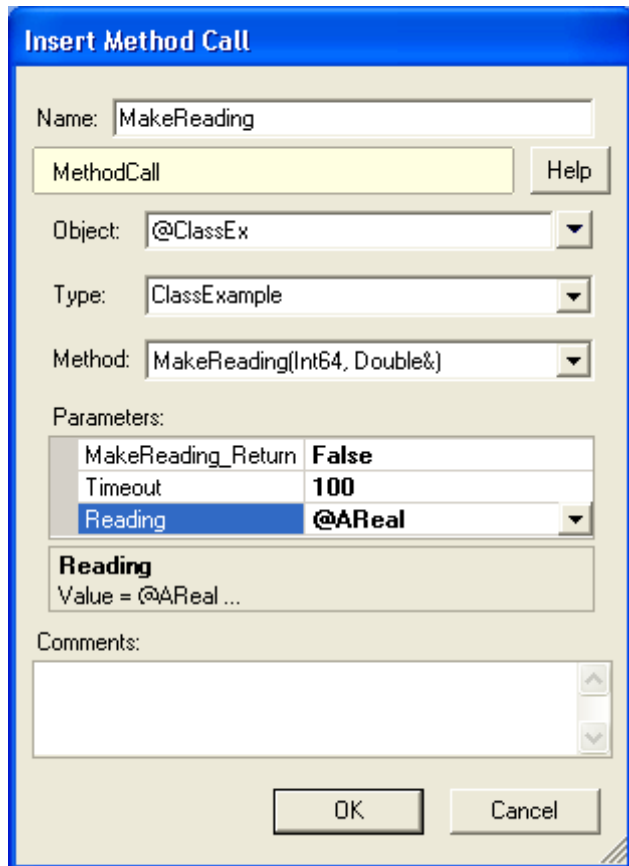


Figure 13 Method Call with Return Value and Parameters

You can add an If/Then flow control at this point. Highlight MakeReading and select **Insert > Flow Control > If..Then** from the main menu. Set the Condition equal to @ABoolean. In this way, the 'ThenItems' run depending upon the value of 'ABoolean' set by the prior MakeReading method call.

For the sequence to progress into the ThenItems part of the If/Then statement, @ABoolean has to be true, which means a measurement was taken. Now you want to know if the measurement is within the limits that you set for acceptable performance.

Within the ThenItems part of the If/Then statement, add a result check. Highlight ThenItems and, from the main menu, select **Insert > Result Check**. The first thing you have to select is the Result. In this case, you are checking the value of @AReal. This is typical, as normally you are loading the results of a measurement into a variable. Set the Minimum and Maximum as shown in [Figure 14](#) on page 29. Name this Result Check CheckReading.

Note one thing of interest regarding limits. Single-ended result checks are equality checks, and automatically show up for string and boolean data types. For integers, setting both minimum and maximum to the same value is equivalent to an equality check. For reals, equality checks are *not* recommended.

NOTE

This is only appropriate for variables of the type integer, boolean, and string. Double values should not use an equality result check.

Insert Result Check

Name:

Result Check:

Publish Result
 Limit check and publish Result

Minimum:

Result:

Maximum:

Publishing

total digits

Comments:

Figure 14 Setting a Result Check

The Analysis Tab is discussed in the next chapter, and it needs multiple data points. Using a For Loop, you can generate data points for this purpose. Highlight the If/Then

statement and choose **Insert > Flow Control > For..Loop** from the main menu. This inserts the For Loop after the If/Then statement.

NOTE

If you have the If/Then statement open, the For Loop may be inserted within it. Remember, you can use the Up/Down icons to quickly change the position of an item in the sequence definition.

Setup the loop by setting the CurrentValue to zero, the ValueStart to one, the ValueStop to thirty, and the ValueStep to one. Name the For Loop *NormalLoop*.

For Loops use a loop counter (usually one) to iterate through the looping operation. ValueStart is the starting value for the For..Loop operator. ValueStop is the ending value for the For..Loop operator. ValueStep is the incrementing value of the operator for each iteration of the For..Loop. CurrentValue is the current loop count; its value changes as the loop iterates.

The NormalLoop needs one method call and a result check to generate the information needed for the Analysis Tab demonstration. The method call comes first. Highlight NormalLoop and insert a method call. Using [Figure 15](#) on page 31 as a template, set up the method call. Name the method call ReadNormalValue.

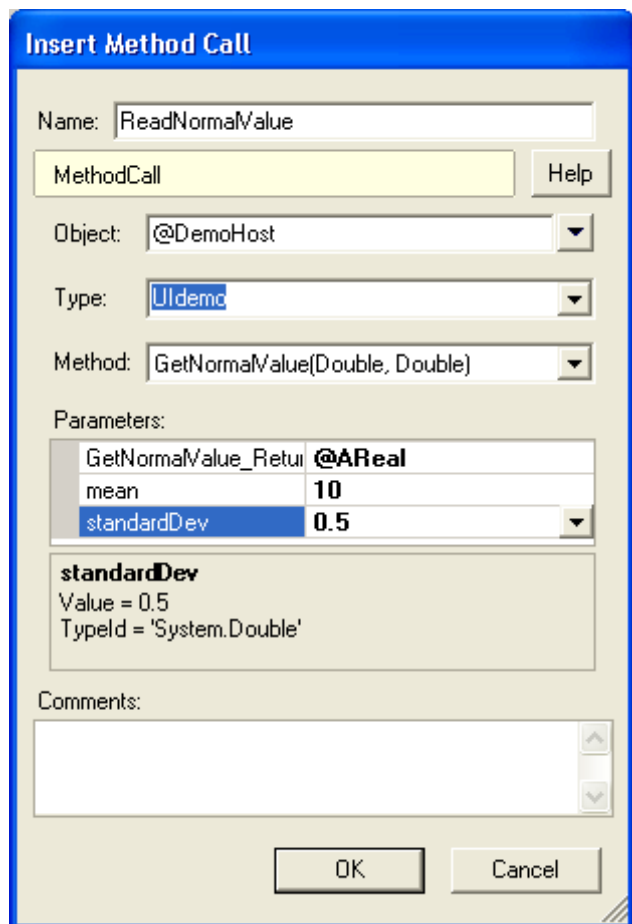


Figure 15 Method Call with Standard Deviation and Mean

Create a result check following the method call you just defined using [Figure 16](#) on page 32 as a template. The result check examines the value of AReal to see if it is between 8 and 12. Name the result check CheckNormal.

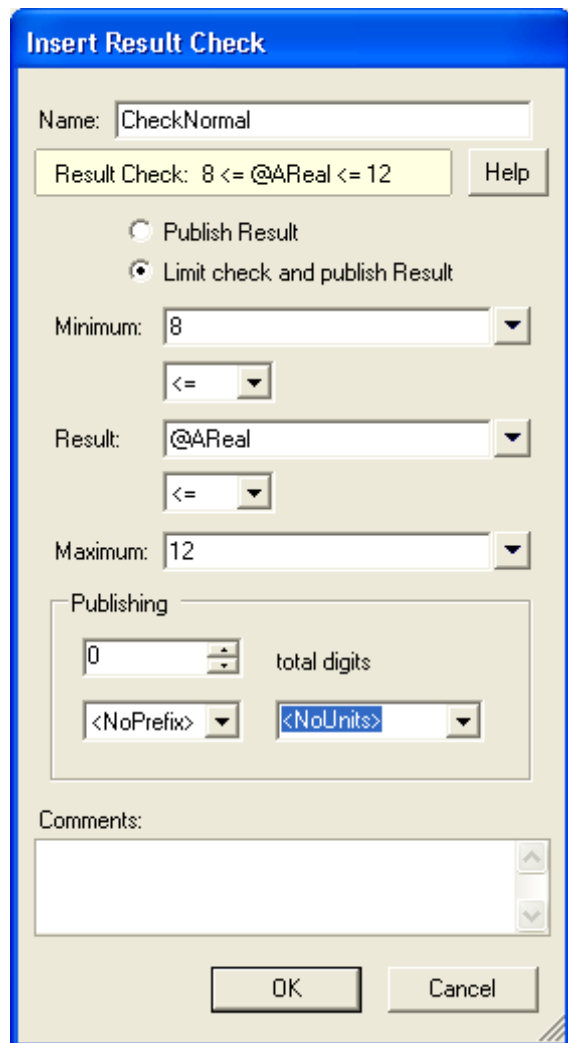





Figure 16 Result Check to Ensure the For Loop Completed

Routines

Before you can proceed to the next step, calling routines, you must build the routines. So, you are going to define a local and external routine and then call them into your sequence definition. The instructions are going to be briefer and more to the point. If you find yourself struggling with anything, refer back to earlier portions of the chapter.


Go to the Routines section () and highlight it. From the **Insert > Routine** on the main menu, select **Local Routine**. Note that a local routine icon () appears with its own set of variables (). Rename the Routine to Routine1. Define the following local variables in the Variables section: AnInt as an Int64 with a default value of 1 and LocalAReal as a double with a default value of 98.7. Both variables are private.

NOTE

Variables defined in the Local Routine are local variables. As such, they are only available to the defined local routine. If you need to create a variable for use in the local routine as well as elsewhere in your sequence definition, create it as a global variable.

Highlight Routine1 and add an assignment statement. Name it SetLocalAnInt. Set the Result to @AnInt and Expression to 5. Then add a result check and name it CheckLocalAnInt. Set the Result to @AnInt and create an equality with a value of 5 (refer to the previous result check documentation on [page 28](#)).

Create an assignment statement. Name it SetGlobalAReal. Set the Result to @AReal and Expression to @LocalAReal. Now add the final item, a result check. Rename it CheckGlobalAReal. Set result to @AReal, Minimum to 98 and Maximum to 99 (<= for both comparators).

Highlight Routines. From **Insert > Routine** choose External Routine (). Name the external routine TraceExample. Select the Sequence drop-down and a similar list appears:

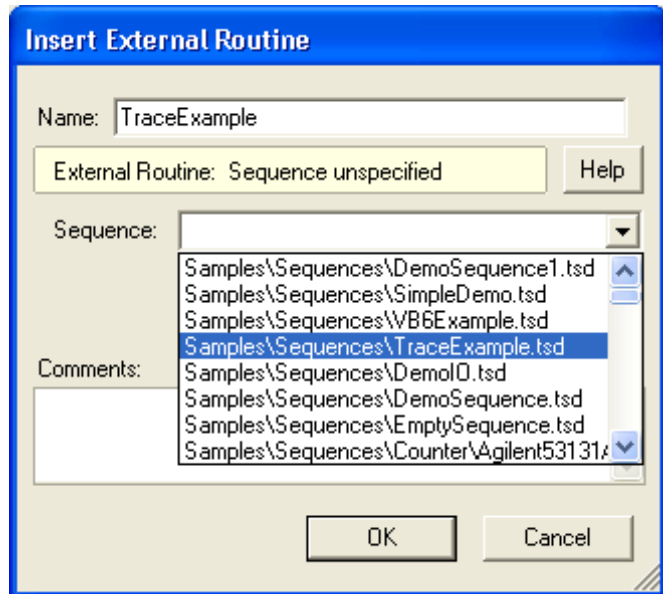


Figure 17 Selecting a Remote Sequence

Select the external sequence `TraceExample.tsd` and select **OK**. From this point on, you will use the **Insert > Routines > Routine Call** to call local and external routines into your sequence definition.

In the final steps of creating the `GettingStarted` sequence, you have to insert a local and an external routine call. Highlight `NormalLoop` and insert a routine call (**Insert > Routine Call**) to a local routine (). Name the local routine call `Routine1`, as in the following graphic.

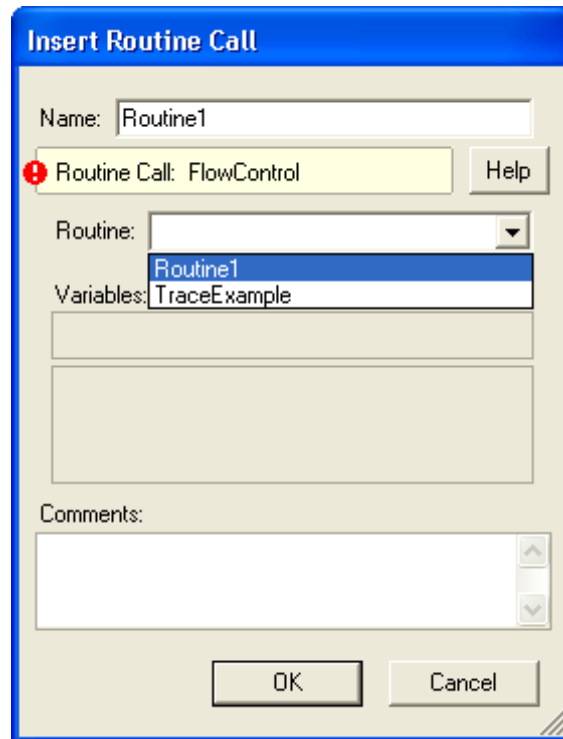


Figure 18 Adding a Routine Call

You need to add a call to an external sequence. Select **Insert > Routine Call** and pick TraceExample from the Routine drop-down box. Name this external routine call CallTraceExample.

Finally, you disable the two sequence calls by right clicking on them and selecting Disable from the context menu. When you disable an item, it is skipped when the sequence runs. You have now finished the sequence definition.

Run the sequence definition and note the following:

- 1 Result checks are always pass/fail with pass being a green ball icon (●) and fail being a red box X icon (⊠). If the result check is embedded in a flow control, the flow

control shows the noted indicator until it is expanded and the result check becomes visible.

- 2 Sequence items not described in #1 are shown as complete by a check mark (✔).
- 3 All variables, whether they are global or local, now have a current value. This value is not reset until the sequence definition file is reloaded or the sequence is run again.
- 4 The Trace Tab contains the transaction tracking information from the latest sequence run.
- 5 The Results Tab data table is populated with the results of the sequence run. These results are cumulative.

Trace Tab

The Trace Tab contains detailed information about the progress of the sequence.

```

Start Sequence "EmptySequence" [12:26:26.205 PM; Id = 1]
Location = "/EmptySequence"
  Start Sequence "Main" [12:26:26.502 PM; Id = 2] [RunCount = 1]
  Location = "/EmptySequence/Main"
    Start Sequence "Main.IfThen" [12:26:26.595 PM; Id = 3]
    Location = "/EmptySequence/Main/IfThen"
      Start Sequence "ThenItems" [12:26:26.611 PM; Id = 4]
      Location = "/EmptySequence/Main/IfThen/ThenItems"
        Measurement "CheckReading" [12:26:26.689 PM]
        Value = 5 : 4 <= Value <= 5.5 >>> PASS <<<
        Location = "/EmptySequence/Main/IfThen/ThenItems/CheckReading"
      End Sequence "ThenItems" [12:26:26.814 PM; Id = 4] >>> PASS <<<
    End Sequence "Main.IfThen" [12:26:26.861 PM; Id = 3] >>> PASS <<<
  Start Sequence "Main.NormalLoop" [12:26:26.892 PM; Id = 5] ←
  Location = "/EmptySequence/Main/NormalLoop"
    Start Sequence "Iteration[1]" [12:26:26.939 PM; Id = 6] [CurrentValue = 1]
    Location = "/EmptySequence/Main/NormalLoop/Iteration"
      Measurement "ResultCheck" [12:26:27.064 PM]
      Value = 11.0867018239045 : 8 <= Value <= 12 >>> PASS <<<
      Location = "/EmptySequence/Main/NormalLoop/ResultCheck"
    End Sequence "Iteration[1]" [12:26:27.095 PM; Id = 6] >>> PASS <<<
    Start Sequence "Iteration[2]" [12:26:27.095 PM; Id = 7] [CurrentValue = 2]
    Location = "/EmptySequence/Main/NormalLoop/Iteration"
      Measurement "ResultCheck" [12:26:27.189 PM]
      Value = 10.5192016195595 : 8 <= Value <= 12 >>> PASS <<<
      Location = "/EmptySequence/Main/NormalLoop/ResultCheck"
    End Sequence "Iteration[2]" [12:26:27.205 PM; Id = 7] >>> PASS <<<

```

Figure 19 Partial Display of the Trace Tab Results

As you look at the output, you can see where the “NormalLoop” starts at 12:26:26:892 PM. This is the For Loop that creates the output for the Analysis demonstration in the next chapter. The two iterations shown are good examples of the depth and breadth of detail you get with the Trace Tab.

The Results Tab is the next point of discovery and is discussed in Chapter 3.

2 Using Test Automator

3


Results Management

Results Management	40
Results Presentation	46

Results Management

Select the Results tab. The data gathered from the Sequence Definition is placed in a data results table shown at the bottom of this tab. This table has many features that simplify data management such as sorting, filtering, column expand and compress, and various other options. Before reviewing these features, size the data results table vertically to about 15 rows.

Columns

When you manage the table's columns using the columns icon () you can select which columns are displayed in the data table. You can also select the order in which they are displayed. See [Figure 20](#) on page 41.

The Use column is not listed amongst the columns. It is a special column that cannot be removed from the data table and is the *only* way to control whether a row stays within the results data set or not. By unchecking a row, you remove it from further consideration for printing, export, saving to a file, or graphing. Even though it is visible to you, it is treated as invisible by the results data set.

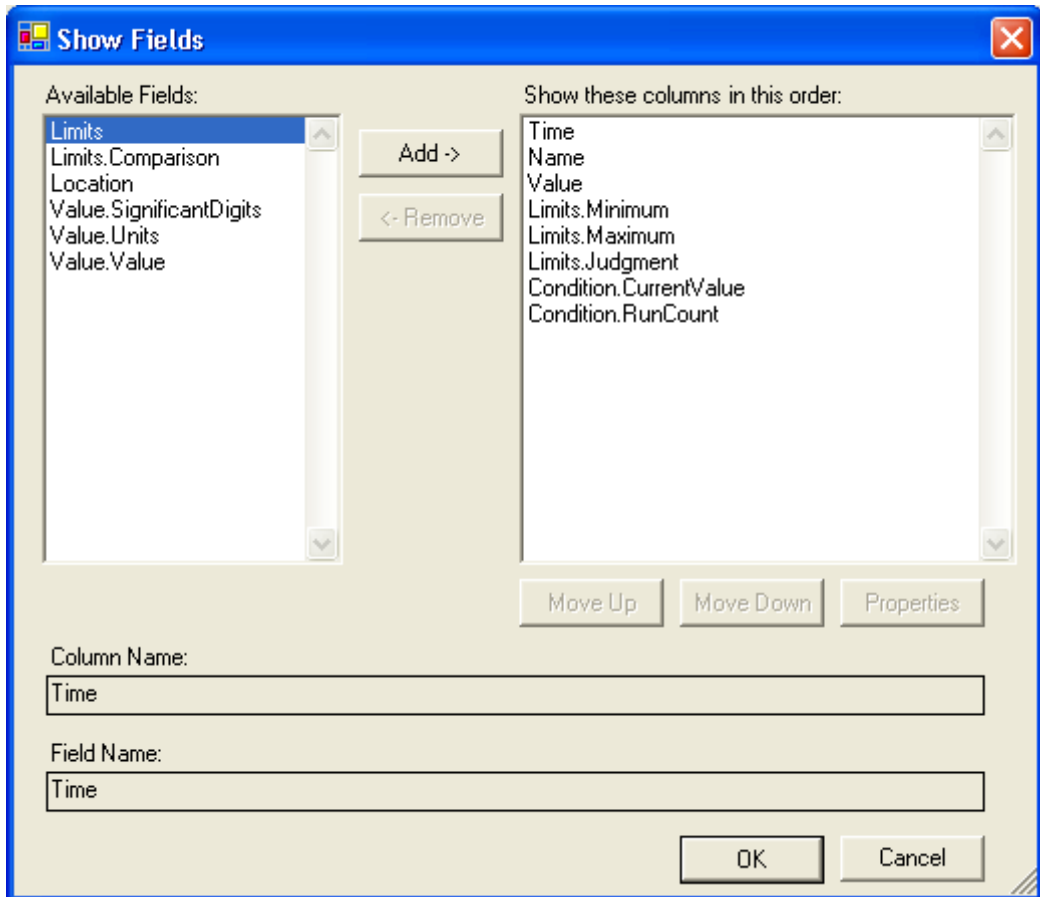



Figure 20 Columns Selection Dialog

You can also select the down arrow next to Columns to get a quick glance at the currently selected fields. Click one of the field entries to add it or remove it from the active display.

Notice some fields, such as Condition.CurrentValue, have sub-fields (illustrated by a period (.) and a name). The expand and compress commands are used with these sub-fields.

Properties

To change the Column Properties, select the Properties icon () and the dialog in [Figure 21](#) on page 43 appears. The Column Properties dialog has several options.

- Column Name: The name of the currently selected column
- Column Width: The width of the currently selected column shown in characters.
- Use Default Formatting: Use the default formatting associated with this data item. If the Use formatting hints from the Value field checkbox is checked, formatting is based upon a format “hint” that the listener for that data item has assigned.

NOTE

Listeners are beyond the scope of this tutorial. Suffice it to say that they are an integral link in the transfer of information to the data results table. The data path is: Result Check to Publisher to Listener to Results Table to Graphs. You can also refer to the online help where Listeners are covered in more detail.

-
- Use a format string: You can force a formatting style by selecting it from the drop-down list.
 - Use abbreviations: Sometimes you do not want to clutter up your display with the full length format descriptor such as milli, so you can check this box to abbreviate the entry to m.
 - Alignment: Left, center, or right justified.

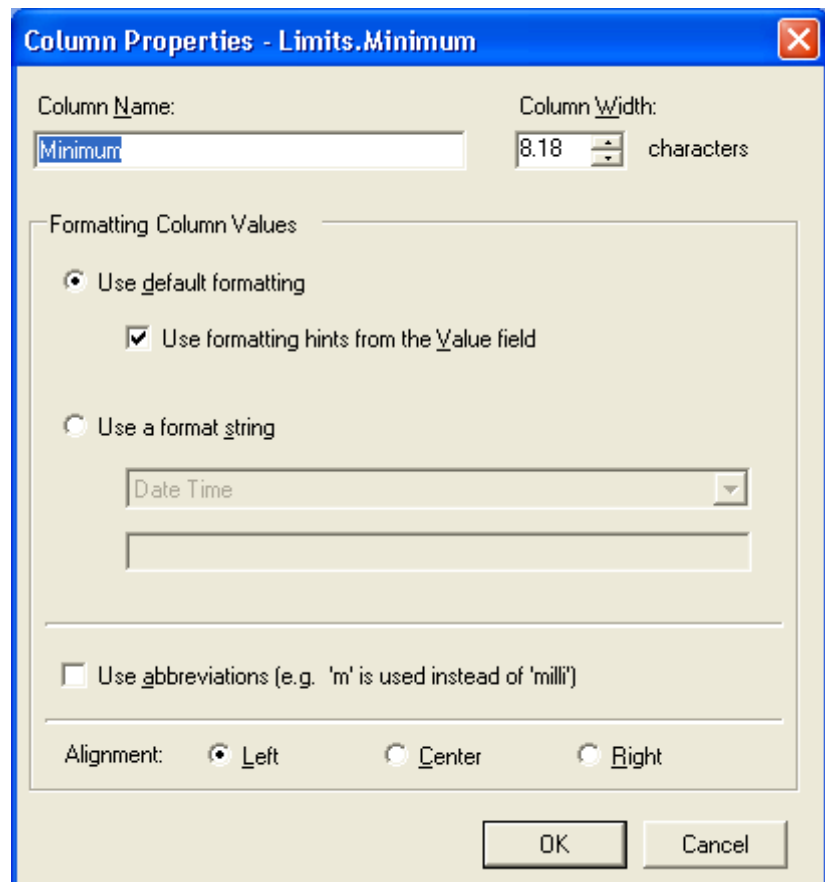




Figure 21 Column Properties Dialog


Sort

Select the Sort icon () to sort the rows of the data table based upon the currently selected column. By selecting the arrow to the right of the icon, you can also do multi-column sorts. The number of columns supported for multi-column sorting is set in ToolBar Options but cannot exceed 10.


Export

The table data can be exported as a comma separated values (CSV) file, a text file, or as an Excel file (the default). Choose the Export icon () and select one of the available options.


Options

Select the Options icon () to adjust the data table options. There are options for the ToolBar and the Table. The options are well explained on the forms, and you should experiment with them. In particular, note that you can turn off the options button. If you do and you want to change an option later, you can reactivate the options button from the right click context menu.


Expand

If a field has sub-fields, you can expand these sub-fields on the data table by highlighting the parent field and then selecting Expand (). This presents all of the data in one column.

Collapse

If a field has sub-fields, you can collapse these sub-fields on the data table by highlighting the parent field and then selecting Collapse (). This presents the data as individual columns.


Remove

If you want to remove a column from the data table, highlight the column and select the Remove icon ()

NOTE

Data is never destroyed when you do a remove. It is only hidden from the data results table. When the data is hidden, it is no longer available for the services provided to other columns in the data results table, such as printing and graphing.


Rename

If you want to rename a column, highlight the column and select the Rename icon ().

Setup

Setup your printer by selecting the setup icon ().

Preview

Preview your print job by selecting the preview icon ().

Print

Select the Print icon () to print your data file.

ToolBar Options

Almost all of the Toolbar characteristics are configurable. The visibility of all buttons on the Toolbar is optional. Tool tips are also optional and multi-column sort is configurable up to 10 columns.

Table Options


You can modify table colors and column header behavior with this set of options.

Results Presentation

Test Automator provides some basic graphs and simple statistics that are intended to be indicative of trend information. The goal of this approach is to help you decide what is important for further study in a more advanced tool such as Excel.

Before playing with the Graphs window, resize the data results table vertically to three rows so you have a full view of the Graphs you can create.

Graphs

Test Automator supports three different graphs: a Normalized Progress Chart, a Histogram, and a Limit Chart. Select Graphs () and pick a graph to display. After you have viewed each one, start with the Histogram and cycle through them as they are described in the following pages.

Histogram

The Histogram presents a histogram of the selected column and statistics about it also.

Histogram Statistics Only the Histogram provides statistics. The statistics offered are: mean, standard deviation, C_{pk} , normality, bar range, and samples.

The Mean is the average value of the sample set (n) as calculated by the sum of n divided by n .

The Standard Deviation is the square root of the variance. That is, the standard deviation of a collection of numbers is the square root of (the difference between the mean of the squares of the numbers and the square of the mean of the numbers).

$$\sqrt{\frac{n \sum x^2 - (\sum x)^2}{n(n-1)}} = S$$

Standard Deviation

(1)

C_{pk} : You can think of the process capability index C_p in three ways:

- 1 C_p measures the capability of a process to meet its specification limits. It is the ratio between the required and actual variability.
- 2 More mathematically, the C_p is the ratio of the Specification difference (upper - lower) divided by 6-sigma, which is the spread of a normal curve.
- 3 Graphically, think of positioning a normal curve centered between the specifications. Now look at the tail areas that exceed the specifications. The smaller the area, the larger the C_p . In this sense, it is equivalent to looking at the popular PPM measure (parts-per-million), which gives the area of the normal curve that exceeds the specifications.

C_{pk} is the same as C_p , but it compensates for a histogram that is not centered. As C_{pk} approaches 1, its value as a predictor of the curve's normality increases. If $C_{pk} \geq 1$, then 99.7% of the products of the process are within specification limits. If $C_{pk} < 1$, then more non-conforming products are being made the lower C_{pk} goes.

Normality is a measure of how closely the curve approaches a normal distribution. This measure is important because, beyond the visual clues of the histogram itself, it helps you determine how useful the C_{pk} measure is.

Bar Range is the width, on the X axis, of a bar.


Samples is the number of samples in the sample set (n).

Normalized Progress Chart

The Normalized Progress Chart plots dissimilar measurement values normalized to a fixed high and low limit line. If no limits are provided, the value is always plotted in the center. This is the chart to use if you are putting different values types, such as hertz and megahertz, in the same column.


Limit Chart

The Limit Chart plots the values as they are collected for a given column of data along with any available limits. The chart is scaled to display all values.

Graphics Properties Select the Graphics Properties icon () to launch the graphics properties dialog. There are two tabs in this dialog, one for the Appearance and one for the Values.


The Appearance tab is straightforward. It should be noted that you do control the formatting of the values on the X axis, Y axis, and the statistics. See Appendix B for tables of the different formatting codes.


The Values tab is also straightforward with the exception of “Scale histogram to include limits when present”. By default, the histogram is presented in a viewable format that provides the most visual value. This is often not the true representation of the histogram. By selecting this checkbox, you can see the histogram’s data distribution in a direct relationship to the limits of the graph.


Graphics Export Select the Graphics Export icon () to export your graph in one of the following formats: .bmp, .gif, .jpeg, .png, and .tiff.


NOTE

There is one limitation to the printing of graphics from Test Automator. Test Automator only prints what is visible in the graphics window. If you have a normalized progress chart, and you have scrolled beyond the right margin, only the graph between the left and right boundaries prints.

Graphics Print Setup You can setup your printer by selecting the printer setup icon ()

Graphics Print Preview You can preview your print job by selecting the print preview icon ()

Graphics Print You can select the Graphics Print icon () to print your graph.

Graphics Options Select the Graphics Options icon () to launch the graphics options dialog. This dialog is self-explanatory.

3 Using Test Automator

Appendix A

Test Automator Configuration File 52

The Contents of the Configuration File 53

Test Automator Configuration File

The Test Automator is configured using the XML file, Test Automator.exe.config. From your perspective, this file defines just about everything about how Test Automator works. Since this configuration file is in an XML format, here are a few pointers for the use of XML:

- All statements begin with '<' and end with '>'. For example `<add key="Type2" value="System.Math" />`.
- Groupings, such as `system.diagnostics`, begin with `<>` and end with `</>` (note the forward slash). For example, `<system.diagnostics> statements ... statements </system.diagnostics>`
- Comments begin with `<!--` and end with `-->`.
- You must create a section to hold the elements you define.

The Contents of the Configuration File

In the following table, the different parts of the configuration file are noted and defined. Use these explanations, and the TestAutomator.exe.config file that ships in the samples\config directory to review possible configuration file content.

Table 1 TestAutomator.exe.Config file description

Configuration Item	Description
configSections	A boilerplate section that must not be changed.
policies	While other policy switches can be defined, the default policies are normal (used for normal testing), errorRecovery (use during error recovery and results are not collected), and debug (use during debugging sessions to include more information).
resultsManagement	There is a direct relationship between the resultsManagement section and the results data table. There are three predefined categories: <ol style="list-style-type: none"> 1 Listeners: The DefaultReportListener forwards reporting text to System.Trace and any trace listeners. You may also add your own listeners here. 2 Units: Define the different unit base names that are available.
resultsVisualization	There is a direct relationship between resultsVisualization and the graphing subsystem. There are two predefined categories: <ol style="list-style-type: none"> 1 resultsTableViewer: Used to define the characteristics of the initial column look and feel. 2 resultsControlViewer: Used to define which graphs are available for presentation
Sequencing	<ol style="list-style-type: none"> 1 References: used to define the assemblies to load. 2 variableTypes: controls the initial set of types available for variables. Types can be from referenced assemblies or standard .NET Framework types.

Table 1 TestAutomator.exe.Config file description

Configuration Item	Description
contentKinds	<p>Menu options. There are several different options here.</p> <ol style="list-style-type: none"> 1 add name: The name of the contentKind. 2 menuPosition: The position on the menu. Small numbers appear earlier. 3 caption: The way the textual name appears on the Insert menu. 4 category: Used for the title of a sub-menu. 5 definitionType: Examples of definition types are - Instrument, DirectIO, AssignExpression, and FlowControl. 6 editorType: The editor that is presented as a dialog box when you select one of the sequence items. 7 runtimeType: Every sequence item has three aspects - an editor, a definition, and a runtime. This is where the runTime is defined. 8 hostObjectType: Sometimes the runtime information is inadequate and Test Automator needs more information. For example, a flow control is a runtimeType but there are multiple types of flow control (for loops, if/ then, groups, and so forth). In these cases, Test Automator needs more information so it relies on the hostObjectType for further clarification. 9 xmlType: Sometimes the XML in the TestAutomator.exe.config file cannot be read by the Test Automator application. In these instances, a hint is provided to the configuration file reader by adding information to the xmlType category. 10 imageResourceName and disabledImageResourceName: Paths to the images you have associated with the menu option.

Appendix B

Numeric Formats 56

Numeric Formats

Standard numeric format strings are used to format common numeric types. A standard numeric format string takes the form *Axx* where *A* is a single alphabetic character called the format specifier, and *xx* is an optional integer called the precision specifier. The format specifier must be one of the built-in format characters listed in [Table 2](#). The precision specifier ranges from 0 to 99 and controls the number of significant digits or zeros to the right of a decimal. The format string cannot contain spaces. The numeric formats that are supported are:

Table 2 Microsoft® Documents their Numeric Formatting Specifiers as:

Specifier	Name	Description
C or c	Currency	The number is converted to a string that represents a currency amount. The conversion is controlled by the currency format information of the <code>NumberFormatInfo</code> object used to format the number. The precision specifier indicates the desired number of decimal places. If the precision specifier is omitted, the default currency precision given by the <code>NumberFormatInfo</code> is used.
D or d	Decimal	This format is supported for integral types only. The number is converted to a string of decimal digits (0-9), prefixed by a minus sign if the number is negative. The precision specifier indicates the minimum number of digits desired in the resulting string. If required, the number is padded with zeros to its left to produce the number of digits given by the precision specifier.
E or e	Scientific (exponential)	The number is converted to a string of the form "-d.ddd...E+ddd" or "-d.ddd...e+ddd", where each 'd' indicates a digit (0-9). The string starts with a minus sign if the number is negative. One digit always precedes the decimal point. The precision specifier indicates the desired number of digits after the decimal point. If the precision specifier is omitted, a default of six digits after the decimal point is used. The case of the format specifier indicates whether to prefix the exponent with an 'E' or an 'e'. The exponent always consists of a plus or minus sign and a minimum of three digits. The exponent is padded with zeros to meet this minimum, if required.
F or f	Fixed-point	The number is converted to a string of the form "-ddd.ddd..." where each 'd' indicates a digit (0-9). The string starts with a minus sign if the number is negative. The precision specifier indicates the desired number of decimal places. If the precision specifier is omitted, the default numeric precision given by the <code>NumberFormatInfo</code> is used.

Table 2 Microsoft® Documents their Numeric Formatting Specifiers as:

G or g	General	<p>The number is converted to the most compact of either fixed-point or scientific notation, depending on the type of the number and whether a precision specifier is present. If the precision specifier is omitted or zero, the type of the number determines the default precision, as indicated by the following list.</p> <p>Byte or SByte: 3 Int16 or UInt16: 5 Int32 or UInt32: 10 Int64 or UInt64: 19 Single: 7 Double: 15 Decimal: 29</p> <p>Fixed-point notation is used if the exponent that would result from expressing the number in scientific notation is greater than -5 and less than the precision specifier; otherwise, scientific notation is used. The result contains a decimal point if required and trailing zeroes are omitted. If the precision specifier is present and the number of significant digits in the result exceeds the specified precision, then the excess trailing digits are removed by rounding. If scientific notation is used, the exponent in the result is prefixed with 'E' if the format specifier is 'G', or 'e' if the format specifier is 'g'. The exception to the preceding rule is if the number is a Decimal and the precision specifier is omitted. In that case, fixed-point notation is always used and trailing zeroes are preserved.</p>
N or n	Number	<p>The number is converted to a string of the form "-d,ddd,ddd.ddd...", where each 'd' indicates a digit (0-9). The string starts with a minus sign if the number is negative. Thousand separators are inserted between each group of three digits to the left of the decimal point. The precision specifier indicates the desired number of decimal places. If the precision specifier is omitted, the default numeric precision given by the NumberFormatInfo is used.</p>
P or p	Percent	<p>The number is converted to a string that represents a percent as defined by the NumberFormatInfo.PercentNegativePattern property or the NumberFormatInfo.PercentPositivePattern property. If the number is negative, the string produced is defined by the PercentNegativePattern and starts with a minus sign. The converted number is multiplied by 100 in order to be presented as a percentage. The precision specifier indicates the desired number of decimal places. If the precision specifier is omitted, the default numeric precision given by NumberFormatInfo is used.</p>

Table 2 Microsoft® Documents their Numeric Formatting Specifiers as:

R or r	Round-trip	The round-trip specifier guarantees that a numeric value converted to a string is parsed back into the same numeric value. When a numeric value is formatted using this specifier, it is first tested using the general format, with 15 spaces of precision for a Double and 7 spaces of precision for a Single . If the value is successfully parsed back to the same numeric value, it is formatted using the general format specifier. However, if the value is not successfully parsed back to the same numeric value, then the value is formatted using 17 digits of precision for a Double and 9 digits of precision for a Single . Although a precision specifier can be appended to the round-trip format specifier, it is ignored. Round trips are given precedence over precision when using this specifier. This format is supported by floating-point types only.
X or x	Hexadecimal	The number is converted to a string of hexadecimal digits. The case of the format specifier indicates whether to use uppercase or lowercase characters for the hexadecimal digits greater than 9. For example, use 'X' to produce "ABCDEF", and 'x' to produce "abcdef". The precision specifier indicates the minimum number of digits desired in the resulting string. If required, the number is padded with zeros to its left to produce the number of digits given by the precision specifier. This format is supported for integral types only.

A

Alignment for columns 42
Assignment 26

B

Bar range 47

C

Call 24
Caption 36
Category 36
Collapse a column 44
Columns
 collapse 44
 expand 44
 formatting 42
 icon 40
 name 42
 options 45
 print 45
 properties 42
 remove 44
 rename 45
 sorting 43
 Use column 40
 width 42
Completed sequence items 36
Config section
 contentKinds 36
 policies 35
 resultsManagement 35
 resultsVisualization 35
contentKinds 36
CP_k 47
Creating a Sequence Definition 16

D

Data
 analysis 46
 CSV export 44
 Excel export 44
 text file export 44

Data table options 44
Debugging a sequence 37
definitionType 36

E

editorType 36
Expand a column 44
Export to a file 44
External routine 33

F

For loop 29
Formatting columns 42

G

Global versus local variables 33
Graphics
 export 48
 options 49
 print 49
 print preview 49
 print setup 49
 properties 48

H

Histogram
 statistics 46
hostObjectType 36

I

Icons for Pass/Fail 35
If/Then/Else 27
imageResourceName 36
Insert an item below the current item 19
Inserting a call to a routine 34
Installation 12

L

Limit check 28
Local routine 33

M

Main category 24
Mean 46
menuPosition 36

N

Normality 47

O

Object variables 24
Options
 columns 44
 data table 44
 graphics 49
 table 45
 toolbar 45
Output tab 37
Overview of Test Automator 2

P

Pass/fail status 35
Policies 35
Print
 graphics 49
Print preview 45
 graphics 49
Print setup
 graphics 49
Printer setup 45
Process capability index 47
Product Key 12
Properties for a column 42

R

Record sorting 43
Redo an action 18
Remove 44
 columns 44
Rename
 a variable 23
 columns 45

- Resetting a variable value 36
- Routines 33
- Run Initialize once 21
- runtimeType 36

S

- Samples 47
- Save a file 44
- Saving graphics 48
- Selecting data records 40
- Sequence item pass/fail status 35
- Sequence output 37
- Standard deviation 46
- Statistics 46
- Sub-fields 41
 - collapse 44
 - expand 44
- System Requirements 13

T

- Table options 45
- Test Automator screen layout 4
- Text file export 44
- The Initialize category 21
- Toolbar options 45

U

- Undo an action 18
- Use column 40
- Using variable with calls 26

V

- Variables 16
 - current value 36
 - that are objects 21

X

- XML syntax 34
- xmlType 36