NI TestStand™

Getting Started with TestStand

NATIONAL INSTRUMENTS™

Worldwide Technical Support and Product Information

`ni.com`

Worldwide Offices

Visit `ni.com/niglobal` to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

National Instruments Corporate Headquarters

11500 North Mopac Expressway   Austin, Texas 78759-3504   USA   Tel: 512 683 0100

For further support information, refer to the *Technical Support and Professional Services* appendix. To comment on National Instruments documentation, refer to the National Instruments Web site at `ni.com/info` and enter the Info Code `feedback`.

Important Information

IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS, SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE, INJURY, OR DEATH, THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM FAILURES, INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM NATIONAL INSTRUMENTS' TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE NATIONAL INSTRUMENTS PRODUCTS IN COMBINATION WITH OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY NATIONAL INSTRUMENTS, THE USER OR APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF NATIONAL INSTRUMENTS PRODUCTS WHENEVER NATIONAL INSTRUMENTS PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

Conventions

The following conventions are used in this manual:

»	The **»** symbol leads you through nested menu items and dialog box options to a final action. The sequence **Options»Settings»General** directs you to pull down the **Options** menu, select the **Settings** item, and select **General** from the last dialog box.
	This icon denotes a note, which alerts you to important information.
bold	Bold text denotes items that you must select or click on in the software, such as menu items and dialog box options. Bold text also denotes parameter names.
italic	Italic text denotes variables, emphasis, a cross-reference, or an introduction to a key concept. Italic text also denotes text that is a placeholder for a word or value that you must supply.
`monospace`	Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames, and extensions.

Contents

Chapter 3
Editing Steps in a Sequence

Chapter 4
Debugging Sequences

Chapter 5
Using Variables and Properties

Chapter 6
Using Callback Sequences

Chapter 7
Adding Users and Setting Privileges

Chapter 8
Interactive Executions

Chapter 9
Calling Sequences Dynamically

Chapter 10
Customizing Reports

Appendix A
Technical Support and Professional Services

Index

1

Introduction to TestStand

NI TestStand is a flexible and open test management framework for building, customizing, and deploying a full-featured test management system.

If you are using TestStand for the first time, National Instruments recommends that you complete the tutorials in this manual. These tutorials begin with a general introduction to the TestStand Sequence Editor and continue by teaching you how to build sequences, manage test data, and customize TestStand functionality. Because the steps of the tutorials depend on elements from previous tutorials, National Instruments recommends that you follow the chapter outline in order. Completed solution files are located in the `<TestStand Public>\Tutorial\Solution` directory.

Refer to the *NI TestStand Help* for more information about TestStand features covered in this manual. The TestStand documentation assumes you are familiar with the concepts and tutorials in this manual. Select **Start»All Programs»National Instruments»TestStand»Documentation»NI TestStand Help** or select **Help»NI TestStand Help** in the sequence editor to access the *NI TestStand Help*.

TestStand Overview

The *NI TestStand System and Architecture Overview Card* includes a complete architectural diagram of TestStand, descriptions of the various system components, and diagrams that illustrate sequence file execution flow, `Execution` object structure, and sequence file structure. Use this card to familiarize yourself with TestStand concepts and refer to the card while you review this manual. Select **Start»All Programs»National Instruments»TestStand»Documentation»Manuals»NI TestStand System and Architecture Overview Card** to access the *NI TestStand System and Architecture Overview Card*.

Use the *Guide to TestStand Documentation* topic in the *NI TestStand Help* to access all the TestStand documentation in electronic format. Select **Start»All Programs»National Instruments»TestStand»Documentation»Guide to Documentation** or select **Help»Guide To Documentation** in the sequence editor to access the *Guide to TestStand Documentation* topic in the *NI TestStand Help*.

Creating a TestStand-Based System

Creating a TestStand-based test solution is a process that includes designing the test system, developing test sequences for units under test (UUTs), customizing the TestStand framework, debugging the test sequences, and deploying the test system to test station computers.

National Instruments offers a variety of licenses for the different ways you can use TestStand in development and deployment applications. Refer to the *Licensing Options for TestStand Systems* section of this chapter for more information about each license.

Figure 1-1 shows the life cycle of developing a TestStand system.

Figure 1-1. TestStand Development Life Cycle
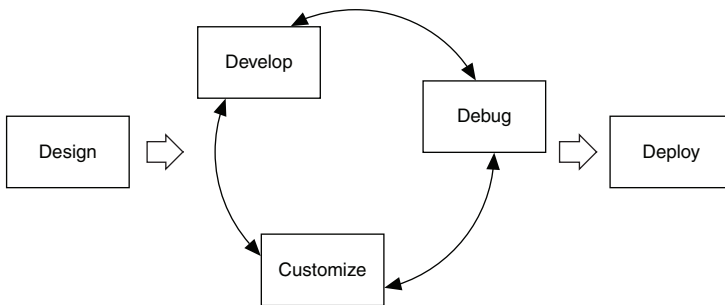
Design

Learn about the components of the TestStand framework, learn how to use those features, and understand when to customize that behavior. Plan the system architecture of the proposed solution. Determine the development environments to use to develop user interfaces and code modules in the solution. Consider how to manage the development and deployment of the solution, including how to organize files under development, how to deploy the files, and how to debug and update systems you deploy.

Refer to the *Major Software Components of TestStand* section of this manual and to the *NI TestStand Help* for more information about the components and features of TestStand and how to customize those features. Refer to the NI Developer Zone at ni.com/zone for more advanced concept and architecture information. For example, the *NI TestStand Advanced Architecture Series* is a suite of documents TestStand architects and developers created to provide more detailed information for experienced TestStand users with complex projects. Refer to the National Instruments website at ni.com/info and enter the Info Code rdtaas to locate the *NI TestStand Advanced Architecture Series*.

Develop

Write test sequences for use on a test station computer. A test sequence is a series of steps that initialize instruments, perform complex tests, or change the flow of executions. Use the sequence editor or a custom sequence editor to write and edit test sequences.

Customize

Edit the default behavior of the TestStand framework depending on the needs of the application you create. You can customize reporting, database logging, process models, callback sequences, and user interfaces to create a unique, robust test solution for an application.

Debug

Ensure that the test sequences and any customized features execute correctly before you deploy the test system. Use the TestStand Sequence Analyzer in the sequence editor or the stand-alone sequence analyzer application during development or before deployment to find errors and enforce custom development guidelines you establish.

TestStand provides multiple features in the sequence editor or a custom sequence editor for debugging sequences, including tracing, breakpoints, conditional breakpoints, stepping through code, and watch expressions. The TestStand development cycle is an iterative process, and you might have to debug an application multiple times.

Deploy

After you develop, customize, and debug a TestStand system, you can deploy the system to multiple test stations. The TestStand Deployment Utility simplifies the complex process of deploying a TestStand system by automating many of the steps involved, including collecting sequence files, code modules, configuration data for instruments, and support files for the test system. You can also use the deployment utility to create an installer.

Major Software Components of TestStand

The major software components of TestStand include the TestStand Engine, sequence editor, user interfaces, module adapters, process models, and deployment utility.

TestStand Engine

The TestStand Engine is a set of DLLs that exports an ActiveX Automation server API. The sequence editor and TestStand User Interfaces use the TestStand API, which you can call from any programming environment that supports access to ActiveX servers, including code modules you write in LabVIEW and LabWindows™/CVI™. The *NI TestStand Help* includes the detailed documentation for the TestStand API. The *NI TestStand API Reference Poster* and the *NI TestStand User Interface Controls Reference Poster* include an overview of the TestStand API.

TestStand Sequence Editor

The sequence editor is a development environment in which you create, edit, execute, and debug sequences and the tests sequences call. Use the sequence editor to access all features, such as step types and process models. Refer to the *Process Models* section of this chapter for more information about process models.

You can debug a sequence using the following techniques, similar to how you debug in application development environments (ADEs) such as LabVIEW, LabWindows/CVI (ANSI), and Visual Studio:

•   Setting breakpoints

•   Stepping into, out of, or over steps

•   Tracing through program executions

•   Displaying variables

•   Monitoring variables, expressions, and output messages during executions

•   Performing static analysis of sequence files to locate errors and enforce coding guidelines

In the sequence editor, you can start multiple concurrent executions. You can execute multiple instances of the same sequence, or you can execute different sequences at the same time. Each execution instance opens an Execution window. In Trace Mode, the Execution window shows the steps in the currently executing sequence. If the execution suspends, the Execution window shows the next step to execute and provides debugging options.

In the sequence editor, you can fully customize the pane and tab layout to optimize development and debugging tasks. You can also customize the menus, toolbars, and keyboard shortcuts.

User Interfaces

A TestStand User Interface is an application you deploy to a development system or a production station to provide a custom GUI for executing, debugging, or editing sequences. Simple user interfaces might only support running sequences, and custom sequence editors might support editing, running, and debugging sequences.

TestStand includes separate user interface applications developed in LabVIEW, LabWindows/CVI, Microsoft Visual Basic .NET, C#, and C++ (MFC). Because TestStand also includes the source code for each user interface, you can fully customize the user interfaces. You can create a custom user interface using any programming language that can host ActiveX controls or control ActiveX Automation servers.

With the user interfaces in Editor Mode, you can modify sequences and display sequence variables, sequence parameters, step properties, and so on. With the user interfaces in Operator Mode, you can start multiple concurrent executions, set breakpoints, and step through sequences.

Module Adapters

The TestStand Engine uses module adapters to invoke code modules that sequences call. Module adapters load and call code modules, pass parameters to code modules, and return values and status from code modules. The module adapters support the following types of code modules:

- LabVIEW VIs
- LabWindows/CVI functions in source files, object files, or library modules you create in LabWindows/CVI or other compilers
- C/C++ functions in DLLs
- .NET assemblies
- ActiveX Automation servers
- HTBasic subroutines

Adapters specific to an application development environment (ADE) can open the ADE, create source code for a new code module in the ADE, and display the source for an existing code module in the ADE. The adapters support stepping into the source code in the ADE while you execute the step from the TestStand Sequence Editor or a TestStand User Interface.

TestStand includes the following module adapters:

- **LabVIEW Adapter**—Calls LabVIEW VIs with a variety of connector panes. The VIs can exist in LLBs, or in LabVIEW packed project libraries. You can also call VIs in the context of a LabVIEW project or LabVIEW class member VIs.
- **LabWindows/CVI Adapter**—Calls C functions in an object file, library file, or DLL with a variety of parameter types. The function can also exist in a source file located in the project you use to execute steps out-of-process in the LabWindows/CVI development environment.
- **C/C++ DLL Adapter**—Calls C/C++ functions and static C++ class methods in a DLL with a variety of parameter types. You can call global static methods or static class methods in C++ DLLs. You can create the DLL code module with LabWindows/CVI, Microsoft Visual Studio, or any other environment that can create a C/C++ DLL.
- **.NET Adapter**—Calls .NET assemblies written in any .NET-compliant language, such as C# or Microsoft Visual Basic .NET.
- **ActiveX/COM Adapter**—Creates ActiveX/COM objects, calls methods, and accesses properties of those objects. When you create an object, you can assign the object reference to a variable or property for later use in other ActiveX/COM Adapter steps. When you call methods and access properties, you can specify an expression for each input and output parameter.
- **HTBasic Adapter**—Calls HTBasic subroutines without passing parameters directly to a subroutine. TestStand provides a library of CSUB routines that use the TestStand API to access TestStand variables and properties from an HTBasic subroutine.
- **Sequence Adapter**—Passes parameters when you make a call to a subsequence. You can call a subsequence in the current sequence file, in another sequence file, or a sequence file on a remote system. You can also make recursive sequence calls. For subsequence

parameters, you can specify a literal value, pass a variable or property by reference or by value, or use the default value the subsequence defines for the parameter.

Refer to the *Using LabVIEW and LabWindows/CVI with TestStand* manual for more information about using LabVIEW, the LabVIEW Adapter, LabWindows/CVI, and the LabWindows/CVI Adapter. Refer to Chapter 7, *Effectively Using LabVIEW with TestStand*, of the *Using LabVIEW and LabWindows/CVI with TestStand* manual for more information about how to best use LabVIEW features in a TestStand system.

Process Models

Testing a UUT requires more than just executing a set of tests. Usually, the test system must perform a series of operations before and after it executes the sequence that performs the tests. Common operations that define the testing process include identifying the UUT, notifying the operator of pass/fail status, logging results, and generating a report. The set of operations and the flow of execution is called a process model. A TestStand process model is a sequence file you can use to define standard testing operations so you do not have to re-implement the same operations in every sequence file you write.

TestStand includes a predefined Sequential model, Parallel model, and Batch model you can modify or replace. Use the Sequential model to run a test sequence on one UUT at a time. Use the Parallel and Batch models to run the same test sequence on multiple UUTs simultaneously.

You can modify an existing TestStand process model or you can create a custom process model.The ability to modify a process model is essential because the testing process can vary depending on production lines, production sites, or company systems and practices. You can edit a process model in the same way you edit other sequence files. You can also use client sequence files to customize various model operations by overriding the callback sequences process models define.

A process model defines a set of entry points, and each entry point is a sequence in the process model file that invokes a test sequence file. Defining multiple entry points in a process model gives the test station operator different ways to invoke a Main sequence or configure the process model.

Execution entry points in a process model provide different ways for the test station operator to invoke a Main sequence. Execution entry points handle common operations, such as unit under test (UUT) identification and report generation. For example, the default Sequential process model provides the Test UUTs and Single Pass Execution entry points. The Test UUTs Execution entry point initiates a loop that repeatedly identifies and tests UUTs. The Single Pass Execution entry point tests a single UUT without identifying it.

TestStand Deployment Utility

The TestStand Deployment Utility is a tool you can use to create a deployable image of a TestStand system and an optional installer contains the deployable image. The deployable image can contain sequence files, code modules and supporting files, process models and supporting files, user interface applications, configuration files, and step types and supporting files the TestStand system uses.

You can also use the deployment utility to include the TestStand Engine and supporting files, LabVIEW and LabWindows/CVI Run-Time Engines, and hardware drivers in the installer you create. The installer that the deployment utility creates can also register ActiveX servers, replace existing files on the target computer, and create program shortcuts. You can configure the deployment utility to remove VI block diagrams or to lock the VIs you deploy.

Licensing Options for TestStand Systems

National Instruments offers a variety of licenses for the different ways you can use TestStand in development and deployment applications. Refer to the *NI TestStand Help* or to the *Activating TestStand Licenses* section of the *NI TestStand Release Notes* for more information about the differences among the TestStand licensing options.

- **Designing, developing, and debugging sequences**—You must have one of the following licenses to write sequences:
  - TestStand Development System License
  - TestStand Custom Sequence Editor License

  You typically use the TestStand Debug Deployment Environment License to debug and fix deployed test systems.
- **Developing a user interface**—You must have the following license to develop a custom user interface, including a custom sequence editor:
  - TestStand Development System License
- **Creating a deployment**—You must have one of the following licenses to create a deployment:
  - TestStand Development System License
  - TestStand Custom Sequence Editor License
- **Deploying a TestStand system**—You must have one of the following licenses on the computer to which you deploy a test system, and each test station for a deployed test system must have its own copy of the license:
  - TestStand Base Deployment Engine License
  - TestStand Debug Deployment Environment License
- **Deploying a custom sequence editor**—You must have one of the following licenses on the computer to which you deploy a test system that includes a custom sequence editor, and each deployed test system must have its own copy of the license:
  - TestStand Custom Sequence Editor License
  - TestStand Development System License

You cannot activate and deactivate the TestStand Debug Deployment Environment License and reuse it on multiple computers. If you need to use a single debug license across multiple computers, contact National Instruments for more information about the Concurrent TestStand Debug Deployment Environment License.

You can use a TestStand Development System License on a computer to which you have deployed a test system to perform any operations you would perform on a development computer. You can also run TestStand on a test station computer in Evaluation Mode until the software evaluation period expires.

2

Loading and Running Sequences

The TestStand Sequence Editor includes menus, toolbars, windows, and panes you use to load workspace files and edit and run sequence files. Click the **Help Topic (F1)** button on the toolbar or select **Help»Help Topic** to launch the *NI TestStand Help* for specific information about the active window, tab, or pane in the sequence editor. Refer to the *NI TestStand Help* for more information about all other features covered in this chapter.

> **Note** Completed solution files are located in the `<TestStand Public>\Tutorial\Solution` directory.

Starting TestStand

Complete the following steps to launch the sequence editor.

1. Select **Start»All Programs»National Instruments»TestStand»Sequence Editor**. The sequence editor launches the main window and the Login dialog box.
2. Use the default user name, `administrator`, in the User Name ring control. Leave the Password field empty.
3. Click **OK**.

Menu Bar

By default, the menu bar contains the File, Edit, View, Execute, Debug, Configure, Source Control, Tools, Window, and Help menus. The menus are fully customizable. You can create menus with any TestStand Sequence Editor commands you want. Browse the menus in the sequence editor to familiarize yourself with the contents of each menu.

Toolbars

By default, the sequence editor contains the following toolbars with shortcuts to commonly used selections from the menu bar, as shown in Figure 2-1.

Figure 2-1. Sequence Editor Toolbars
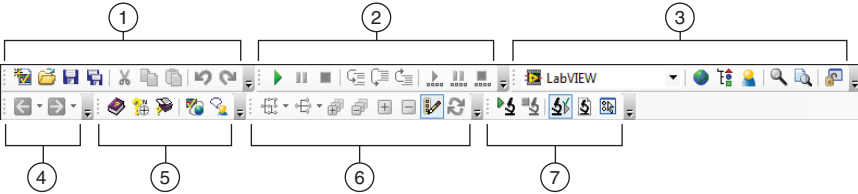
1	Standard Toolbar	5	Help Toolbar
2	Debug Toolbar	6	Sequence Hierarchy Toolbar
3	Environment Toolbar	7	Sequence Analyzer Toolbar
4	Navigation Toolbar		

- **Standard Toolbar**—Contains buttons for creating, loading, and saving sequence files, and for cutting, copying, and pasting. This toolbar also includes the **Undo** and **Redo** buttons.

- **Debug Toolbar**—Contains buttons for executing a sequence, stepping into, stepping over, stepping out of, terminating, and suspending executions.

- **Environment Toolbar**—Contains the following items:
  - Adapter ring control
  - Buttons for opening station globals, type palettes, and the TestStand User Manager
  - Buttons for performing search and replace operations
  - Button for unlocking and locking the user interface (UI) configuration to enable or disable customizing various aspects of the UI in the development environment

- **Navigation Toolbar**—Contains the **Back** and **Forward** buttons, which you use to show the previous or next view in the history list of the sequence file or Sequence Hierarchy window.

- **Help Toolbar**—Contains buttons to launch the *NI TestStand Help*, the TestStand support page of ni.com, and the TestStand Discussion Forum on ni.com.

- **Sequence Hierarchy Toolbar**—Contains buttons for using the Sequence Hierarchy window, which displays a graph of the sequence call and callback sequence relationships among sequence files and sequences to more easily visualize, navigate, and maintain test sequences.

- **Sequence Analyzer Toolbar**—Contains buttons for analyzing a sequence file or a TestStand Sequence Analyzer project to detect errors and for configuring the sequence analyzer options.

The toolbars are fully customizable. You can create toolbars with buttons for any environment commands you want.

Windows and Panes

The sequence editor contains tabbed windows and panes you can float, dock, resize, and hide. The sequence editor uses windows to show sequence files, executions, sequence hierarchies, the current sequence analyzer project, station globals, types, and the user manager. The sequence editor uses panes to show all other information.

You can customize the pane and tab layout to optimize development and debugging tasks. You can reset the user interface configuration state of the panes in the sequence editor by selecting **View»Reset UI Configuration**, which resets to a default configuration all modifications previously made to menus, toolbars, and the docking state of panes.

> **Note**   When you launch the sequence editor for the first time or when you reset the sequence editor configuration, the layout of the panes defaults to the Small Screen Example configuration or the Large Screen Example configuration, depending on the screen resolution. TestStand considers screens with a resolution of 1280×1024 or greater as large.
>
> Select **Configure»Sequence Editor Options** to launch the Sequence Editor Options dialog box and select a configuration. On the UI Configuration tab, select an example configuration in the **Saved Configurations** control and click the **Load Selected** button. Click **OK** to close the Sequence Editor Options dialog box.

The Insertion Palette pane in the TestStand Sequence Editor displays items you can insert into a sequence file. The pane contains a Step Types list and a Templates list. The Step Types list displays the step type menu, and the Templates list displays user-defined template sequences, steps, and variables.

Status Bar

The status bar displays the following information for the sequence editor:

- User name of the current user
- Name of the current process model file; double-click in this area to open the current process model
- Number of steps you have selected
- Total number of steps in the active sequence
- File path of the active report after execution

Organizing a TestStand System with Workspace and Project Files

Create a workspace to organize and access development files. Use workspaces early in development so you can easily keep track of files while you are developing. A workspace file (.tsw) contains references to any number of TestStand project files. A project file (.tpj) contains references to any number of other files of any type.

Use project files to organize related files and directories of files in the test system. You can insert any number of files into a project. For example, you can use a project file to organize a sequence file, code module source files, and other supporting files.

A workspace file opens on the Workspace pane, which displays the content of a workspace file. To open any file from the Workspace pane, double-click the file. To organize the files, use the options in the Edit menu to cut, copy, paste, and delete files in projects or project files in the workspace. You can also drag and drop files on the Workspace pane.

You can open only one workspace file at a time. If you have a workspace file open and you try to open or create another, TestStand prompts you to save the workspace file before opening or creating another file.

You must use workspace files to use the source code control (SCC) system features in TestStand. You can also use workspace files to deploy test systems with the TestStand Deployment Utility.

Loading a Workspace File

Complete the following steps to load and view a workspace file.

1. Select **File»Open File** and navigate to the `<TestStand Public>\Tutorial` directory. `<TestStand Public>` is located by default at `C:\Users\Public\Documents\ National Instruments\TestStand` on Microsoft Windows 7/Vista and at `C:\Documents and Settings\All Users\Documents\National Instruments\TestStand` on Windows XP.

2. Select `Tutorial.tsw` and click **Open**.

3. Expand the **Using TestStand** project, which contains all the sequence files used in the tutorials in this manual, including the `Solution` directory, as shown in Figure 2-2.

Figure 2-2. Workspace Pane with the Using TestStand Project Expanded
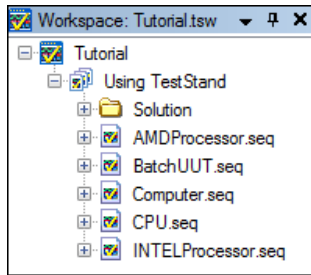
You can leave the workspace file open on the Workspace pane so you can directly open the sequence files for the tutorials in this manual, or you can close the Workspace pane and use the options in the File menu to open the sequence files. If you want to close the Workspace pane, select **File»Close Workspace File** and click **No** if TestStand prompts you to save the changes.

Creating Test Sequences and Sequence Files

A sequence consists of a series of steps. A step can perform many actions, such as initializing an instrument, performing a complex test, or controlling the flow of execution in a sequence.

A sequence file can contain one or more sequences. Sequence files can also contain global variables, which all sequences in the sequence file can access.The sequence editor color-codes sequences by type, such as Execution entry points, Configuration entry points, process model callback sequences, subsequences, and engine callbacks.

The Setup, Main, and Cleanup groups of the Steps pane display a list of the steps in the step group.TestStand executes the steps in the Setup step group first, the Main step group second, and the Cleanup step group last. The Setup step group typically contains steps that initialize instruments, fixtures, or a UUT. The Main step group typically contains the bulk of the steps in a sequence, including the steps that test the UUT. The Cleanup step group typically contains steps that power down or restore the initial state of instruments, fixtures, and the UUT.

A sequence can have any number of parameters and local variables. Use parameters to pass data to a sequence when you call the sequence as a subsequence. Use local variables for storing data relevant to the execution of the sequence, storing any other data needed only in the current sequence, maintaining counts, or holding intermediate values.

In the sequence editor, the Variables pane displays all the variables and properties the selected sequence has access to at run time. When you execute a sequence, the Variables pane displays the sequence context for the sequence currently selected on the Call Stack pane. The sequence context contains all the variables and properties accessible in the current execution. Use the Variables pane to examine and modify the values of these variables and properties when an execution is suspended.

The sequence editor opens each sequence file in a separate Sequence File window, which includes the Sequences pane, the Steps pane, and the Variables pane, as shown in Figure 2-1. The Sequences pane lists the name, comment, and requirements values for all sequences in the sequence file. Use the Sequences pane to select the active sequence to display on the Steps pane, to insert new sequences, and to delete existing sequences from a sequence file.

Figure 2-3. Sequence File Window
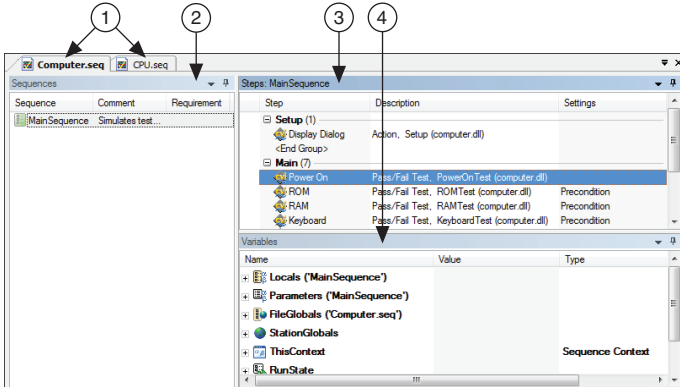
| 1 | Sequence File Windows | 2 | Sequences Pane | 3 | Steps Pane | 4 | Variables Pane |

Loading a Sequence File

Complete the following steps to load and view a sequence file.

1. Select **File»Open File** and navigate to the `<TestStand Public>\Tutorial` directory.

2. Select `Computer.seq` and click **Open**. The `Computer.seq` sequence file is a simulated test of a computer in which you can designate various hardware components to fail. The sequence runs tests implemented as functions in a DLL written with LabWindows/CVI.

> **Note**   If you left the `Tutorial.tsw` workspace file open from the *Loading a Workspace File* section of this chapter, you can open `Computer.seq` from the Workspace pane by double-clicking `Computer.seq` in the list of files or by right-clicking `Computer.seq` and selecting **Open** from the context menu.

3. Select `MainSequence` on the Sequences pane.

4. Browse the contents of each pane in the Sequence File window. Ensure that the Sequences pane and the Steps pane are visible when you finish.

Executing a Sequence

You can execute a sequence directly or execute a sequence using a process model Execution entry point. The process model entry point sequence contains a series of steps that specify the high-level flow of an execution.

Before the execution begins, by default, the TestStand Sequence Analyzer uses the rules and settings in the current sequence analyzer project to analyze the active sequence file and detect the most common situations that can cause run-time failures. The sequence analyzer prompts you to resolve the reported errors before you execute the sequence file. If no errors exist or if you select to continue execution with errors, the execution begins, and the sequence editor opens an Execution window. Click the **Toggle Analyze File Before Executing** button on the Sequence Analyzer toolbar or select **Debug»Sequence Analyzer»Toggle Analyze File Before Executing** to enable or disable this option. Click the **Analyze <*sequence filename*>** button on the Sequence Analyzer toolbar to analyze the currently displayed sequence file at any time during development.

The TestStand Sequence Editor launches a separate Execution window for each execution you start using the Execute menu. Use the Execution window to view, or trace, steps as they execute, to monitor the values of variables and properties, and to examine test reports when the execution completes.

Note You must have Internet Explorer 7.0 or later to view TestStand reports.

In Trace Mode, the Execution window shows the steps in the currently executing sequence and indicates the currently executing step. If the execution suspends, the Execution window shows the next step to execute and provides debugging options.

Executing a Sequence Directly

Complete the following steps to run `MainSequence` in the `Computer.seq` sequence file directly.

1. Select **Execute»Run MainSequence**. The sequence editor opens an Execution window to show the sequence as it executes. The first step in the Setup step group of `MainSequence` launches the Test Simulator dialog box, as shown in Figure 2-4. The Test Simulator dialog box prompts you to designate the computer component, if any, you want to fail during the execution.

Figure 2-4. MainSequence Execution Window
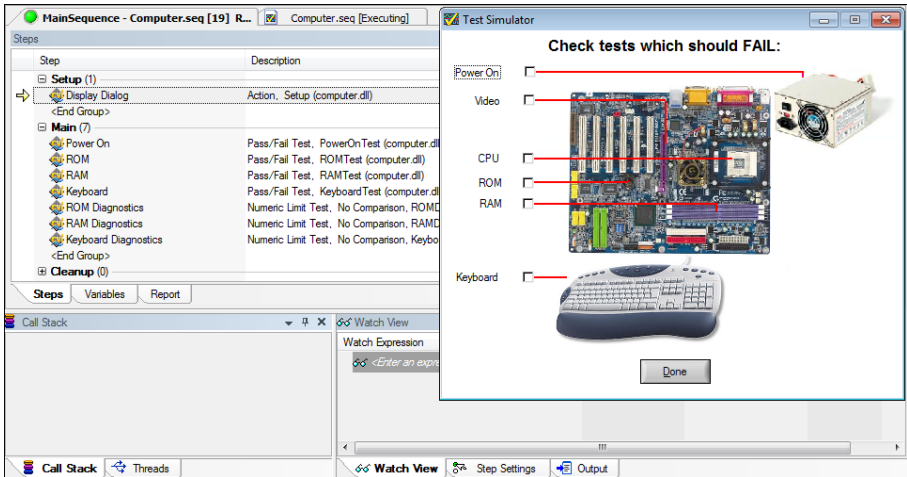
2. Enable the **RAM** test option.

3. Click **Done**.

4. Observe the Execution window as it traces through the steps TestStand runs. If you enable tracing, the TestStand Sequence Editor displays the progress of an execution with a yellow arrow icon located to the left of the icon for the currently executing step. This icon is called the execution pointer, as shown in the following figure.

After the execution completes, the Execution window dims, and the Status column of the `RAM` test contains the value `Failed`.

> 📝 **Note** If you do not see the execution pointer, select **Configure»Station Options** to launch the Station Options dialog box, in which you can enable tracing. On the Execution tab, enable the **Enable Tracing** and **Allow Tracing into Setup/Cleanup** options and use the **Speed** slider control to adjust the tracing speed to the level you want. Click **OK** to close the Station Options dialog box.

5. Right-click the Execution window tab and select **Close** from the context menu or select **Window»Close Completed Executions** to close the Execution window.

Executing a Sequence Using the Sequential Process Model

In addition to executing sequences directly, you can execute sequences using an Execution entry point, which is a sequence in a process model file that invokes a test sequence file, typically by calling the MainSequence callback in the client sequence file. Executing an Execution entry point performs a series of operations before and after calling `MainSequence` of the sequence file. Common operations of the process model include identifying the UUT, notifying the operator of pass/fail status, logging results, and generating reports.

The Sequential process model includes the following Execution entry points:

*   **Test UUTs Execution Entry Point**—The Test UUTs Execution entry point initiates a loop that repeatedly identifies and tests UUTs.
*   **Single Pass Execution Entry Point**—The Single Pass Execution entry point tests a single UUT without identifying it. Use the Single Pass Execution entry point when you want to debug tests or determine whether the sequence execution proceeds as you intended.

Complete the following steps to run `MainSequence` in the `Computer.seq` sequence file using the Test UUTs Execution entry point of the Sequential model.

1. Select **Configure»Station Options** to launch the Station Options dialog box.
2. Click the **Model** tab, select `SequentialModel.seq` from the Station Model ring control to select the Sequential model as the default process model, and click **OK**.
3. Select **Execute»Test UUTs**. Before executing the steps in `MainSequence`, the process model sequence launches a UUT Information dialog box that prompts you for a serial number.
4. Enter any serial number and click **OK**.
5. Use the options in the Test Simulator dialog box to select any test other than the Video or CPU tests to fail. You can also allow all the tests of the UUT to pass.
6. Click **Done**. Observe the Execution window as the sequence executes. After completing the steps in `MainSequence`, the process model shows a banner that indicates the result of the UUT.
7. Click **OK** to close the UUT Result banner. TestStand generates a report, displays the report after testing all the UUTs, and launches the UUT Information dialog box again.
8. Repeat steps 4 through 7 using several different serial numbers.

9. Click **Stop** in the UUT Information dialog box to stop the loop and complete the execution. After the execution completes, TestStand shows a full report on the Report pane of the Execution window for all the tested UUTs.

10. Examine the test report that includes the results for each UUT and close the Execution window.

Note   You must have Internet Explorer 7.0 or later to view TestStand reports.

Executing a Sequence Using the Batch Process Model

Use the Batch model to control a set of test sockets that test multiple UUTs as a group. For example, if you have a set of circuit boards attached to a common carrier, use the Batch model to ensure that you start and finish testing all boards at the same time. With the synchronization features of the Batch model, you can direct a step that applies to the batch as a whole to run only once per batch instead of once for each UUT. You can also specify whether certain steps or groups of steps cannot run on more than one UUT at a time or whether certain steps must run on all UUTs at the same time. The Batch model generates batch reports that summarize the test results for the UUTs in the batch.

Complete the following steps to run the `BatchUUT.seq` sequence file using the Test UUTs Execution entry point of the Batch model.

1. Open `<TestStand Public>\Tutorial\BatchUUT.seq` and examine the steps and comments in the sequence file to familiarize yourself with the functionality of the sequence.

Note   You do not need to change the default process model in the sequence editor for this tutorial. The `BatchUUT.seq` sequence file always uses the Batch model, regardless of the default process model of the sequence editor. Use the Advanced tab of the Sequence File Properties dialog box to specify the process model a sequence file uses.

2. Select **Configure»Model Options** to launch the Model Options dialog box.

3. In the Multiple UUT Settings section, change **Number of Test Sockets** to 2 and enable the **Tile Execution Windows** option. The Number of Test Sockets control specifies the number of UUTs to test in the batch.

4. In the Batch Settings section, select **Don't Synchronize** from the Default Batch Synchronization ring control and click **OK**.

5. Select **Execute»Test UUTs**. Before executing the steps in `MainSequence`, the process model sequence launches the UUT Information dialog box for the Batch model, which prompts you for a batch serial number and UUT serial numbers for each test socket. You can also disable test sockets in the UUT Information dialog box.

6. Enter any batch serial number and UUT serial numbers. Click **Go**. Review the information in the Batch Model Example dialog box, then click **OK**. TestStand launches several different front panels to indicate the progress of the executions. Click **OK** to continue.

   After completing the steps in MainSequence, the process model shows a banner that indicates the result of the UUTs. You can view the batch report and individual UUT reports or start the next batch.

7. Click the **View Batch Report** button. From the pop-up list, you can select Entire File to show all tested batches or Current Only to show the most recently tested batch.

   Note You must have Internet Explorer 7.0 or later to view TestStand reports.

8. Select **Current Only** from the pop-up list. TestStand launches the default application associated with the file extension of the report, such as Microsoft Internet Explorer for Automatic Test Markup Language (ATML), XML, and HTML reports and Microsoft Notepad for text reports.

   Note Enable Internet Explorer to run scripts if the browser launches a prompt.

   To use another application as an external viewer, select **Configure»External Viewers** and follow the instructions in the Configure External Viewers dialog box. Refer to the *Using External Report Viewers* section of Chapter 10, *Customizing Reports*, for more information about using external viewers.

9. Close Internet Explorer and click **Next Batch** in the Batch Results dialog box.

10. Repeat steps 6 through 9 for several different batches.

11. Click **Stop** in the UUT Information dialog box to complete the execution. After the execution completes, TestStand shows test reports on the Report pane of the Execution window for all batches and UUTs.

12. Examine the reports that include results for each batch and UUT and select **Window»Close All Windows**.

3

Editing Steps in a Sequence

You can add a step to a sequence, configure the step to call a code module or subsequence, and configure the properties of a step.

The Insertion Palette contains a set of predefined step types you can add to sequences. Step types define the standard behavior and a set of step properties for each step of that type. All steps of the same type have the same properties, but the values of the properties can differ.

When you build sequence files, you can also use the Templates list in the Insertion Palette. Use the Templates list to hold copies of steps, variables, and sequences you reuse during the development of sequence files. For example, you can add a step that calls a specific LabVIEW VI you typically use or a sequence that contains common setup steps, cleanup steps, and local variables.

Refer to the *NI TestStand Help* for more information about all the features covered in this chapter.

📝 **Note** Completed solution files are located in the `<TestStand Public>\Tutorial\Solution` directory.

Adding a New Step

Complete the following steps to add a Pass/Fail Test step to a sequence and configure the step to call a function in a LabWindows/CVI DLL code module by specifying the module adapter to use. TestStand does not need the source code to invoke a DLL code module. Instead, TestStand uses module adapters to determine the type of code module, how to call the code module, and how to pass parameters to the code module.

1. Open `<TestStand Public>\Tutorial\Computer.seq`.
2. Select **File»Save *<filename>* As** and save the sequence as `Computer2.seq` in the `<TestStand Public>\Tutorial` directory.

📝 **Note** When you must save a file, this manual specifies the suggested name. If other users will use the tutorial files on the same computer, save the files with unique filenames.

3. Click the **LabWindows/CVI** icon at the top of the Insertion Palette to specify the module adapter the step uses. You can also select adapters from the Adapter ring control on the Environment toolbar. The adapter you select applies only to the step types that can use the module adapter.

> **Note**   If you left the `Tutorial.tsw` workspace open in Chapter 2, *Loading and Running Sequences*, you might need to click the **Insertion Palette** tab at the bottom of the Workspace pane to show the Insertion Palette.

When you insert a step in a sequence, TestStand configures the step to use the adapter you selected from the Insertion Palette. The icon for the step reflects the adapter you selected. When you select <None> as the adapter and then insert a step, the new step does not call a code module. Use the General panel on the Properties tab of the Step Settings pane to change the associated adapter after you insert the step.

4. On the Insertion Palette, select **Tests»Pass/Fail Test** and drag the step below the RAM step on the Steps pane to add a Pass/Fail Test step. You can also select the RAM step and double-click the step on the Insertion Palette to add it, or you can right-click the RAM step and select **Insert Step** from the context menu to add a step. By default, the name of the new step is Pass/Fail Test.

    Use a Pass/Fail Test step to call a code module that returns a pass/fail determination. After the code module executes, the Pass/Fail Test step evaluates a Boolean expression to determine whether the step passed or failed.

5. Rename the new step `Video Test` by selecting the name on the Steps pane and entering the new name. You can also select the step on the Steps pane and press the <F2> key or change the Name field on the Properties tab of the Step Settings pane of the Pass/Fail Test step to rename the step.

Specifying the Code Module

A code module is a program module, such as a Microsoft Windows DLL or LabVIEW VI (`.vi`), that contains one or more functions that perform a specific test or other action.

Complete the following steps to specify the code module the step executes and to specify the parameters for a function in the code module for the `Computer2.seq` sequence file you created in the previous tutorial.

1. Select the **Video Test** step and click the **Module** tab of the Step Settings pane. You can also right-click the Video Test step and select **Specify Module** from the context menu to show the Module tab of the Step Settings pane.

2. Click the **Browse** button and select <TestStand Public>\Tutorial\ `computer.dll` and click **Open**. When you select a DLL, TestStand reads the type library or exported information of the DLL and lists the functions TestStand can call in the Function ring control.

3. Select **VideoTest** from the Function ring control. TestStand uses the prototype information stored in the type library or the exported information of the DLL to populate the Parameters Table.

4. In the Value Expression column of the **result** parameter, enter `Step.Result.PassFail`. You can also click the **Expression Browser** button or right-click in the Expression control and select **Browse** from the context menu to launch the Expression Browser dialog box, in

which you can interactively build an expression and create variables and parameters by selecting from lists of variables, properties, operators, functions, and the TestStand API. Click the **Check Expression for Errors** button to verify that the expression contains valid syntax.

When TestStand returns from calling the `VideoTest` function during execution, TestStand assigns the value from the **result** parameter to the `Step.Result.PassFail` property of the step.

5. Select **File»Save <*filename*>** to save the changes to the sequence file. Leave the sequence file open for the next tutorial.

Refer to Chapter 11, *Calling LabWindows/CVI Code Modules from TestStand*, of the *Using LabVIEW and LabWindows/CVI with TestStand* manual for more information about calling LabWindows/CVI code modules from TestStand.

Configuring Step Properties

Each step in a sequence contains properties. All steps have a common set of properties that determine the following attributes:

- When to load the code module for the step
- When to execute the step
- What information TestStand examines to determine the status of the step
- Whether TestStand executes the step in a loop
- What conditional actions occur after a step executes

Steps can also contain additional properties the step type defines. Use the Properties tab of the Step Settings pane to examine and modify the values of properties of a step.

Complete the following steps to examine and modify step properties in the `Computer2.seq` sequence file you created in the previous tutorial.

1. Select the **Video Test** step on the Steps pane and click the **Properties** tab of the Step Settings pane.

2. Click **Preconditions** on the Properties tab to show the Preconditions panel. A precondition is a set of conditions for a step that must be `True` for TestStand to execute the step during the normal flow of execution in a sequence.

3. Complete the following steps to define a precondition so the Video Test step executes only if the Power On step passes.

   a. Click the **Precondition Builder** button on the Preconditions panel to launch the Precondition Builder dialog box.

   b. In the Insert Step Status section, select the **Power On** step from the list of step names for the Main step group and click the **Insert Step Pass** button. The Conditions control now contains the string `PASS Power On`, which indicates that the step executes only if the Power On step passes.

c.   Click **OK** to close the Precondition Builder dialog box and confirm that the Preconditions panel matches the settings shown in Figure 3-1.

Figure 3-1. Preconditions Pane
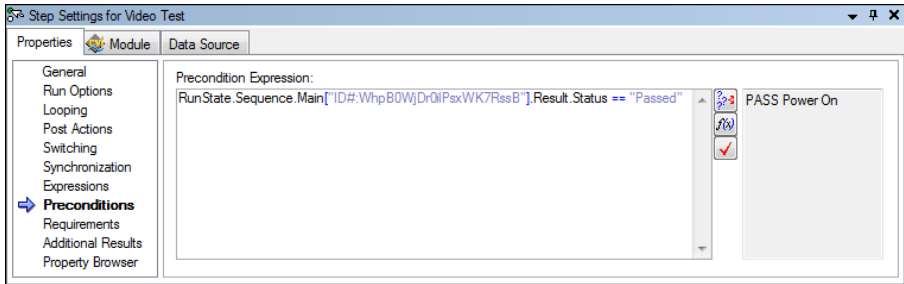
4.   Click **Post Actions** on the Properties tab of the Step Settings pane to show the Post Actions panel, on which you can specify what type of action occurs after the step executes. You can make the action conditional on the Pass/Fail status of the step or on any custom expression. You can enable the Use Custom Condition option, enter a custom expression that evaluates to `True` or `False`, and select the appropriate actions from the On Condition True and On Condition False ring controls.

5.   Select **Terminate execution** from the On Fail ring control.

6.   Click **Looping** on the Properties tab of the Step Settings pane to show the Looping panel, on which you can configure an individual step to run repeatedly in a loop when the step executes. Use the Loop Type ring control to select the type of looping for the step. TestStand determines the final status of the step based on the number of passes, failures, or loop iterations that occur.

7.   On the Looping panel, enter the following values into the corresponding controls. When you change the value of a property, TestStand shows the new value in bold to indicate that it differs from the default value.

   •   **Loop Type**—Fixed number of loops

   •   **Number of Loops**—`10`

   •   **Loop result is Fail if**—`<80%`

   Using these settings, TestStand executes the Video Test step 10 times and sets the overall status for the step to `Failed` if fewer than 8 of the 10 iterations pass.

8.   Confirm that the Settings column on the Steps pane of the Sequence File window shows that the Video Test step contains Loop, Precondition, and Post Action settings. Use the tooltip in the Settings column to verify the values for each setting.

9.   Save the changes to the sequence file.

10. Select **Execute»Single Pass** to run the sequence. Select the Video test to fail. The sequence immediately terminates after calling the Video Test step 10 times in a loop. TestStand records the result of each loop iteration in the report.

11. Close the Execution window.

Calling Subsequences

Use the Sequence Call step to call another sequence in the current sequence file or in another sequence file.

Complete the following steps to add a Sequence Call step to the `Computer2.seq` sequence file you created in the previous tutorial.

1. Insert a Sequence Call step after the Power On step on the Steps pane and rename the step `CPU Test`.

2. On the Module tab of the Step Settings pane for the CPU Test step, click the **Browse** button located to the right of the File Pathname control and select `<TestStand Public>\Tutorial\CPU.seq` to specify the sequence the step invokes.

3. Select **MainSequence** from the Sequence ring control on the Module tab of the Step Settings pane.

4. Save the changes to the `Computer2.seq` file.

5. Double-click the **CPU Test** step or right-click the step and select **Open Sequence** from the context menu to open the sequence and show the `MainSequence` of `CPU.seq` in a new Sequence File window.

6. Select **Execute»Run MainSequence**. Examine the execution of `CPU.seq` so you can recognize it later when you execute the `Computer2.seq` sequence file that calls `CPU.seq`.

7. Close the Execution window and the `CPU.seq` Sequence File window.

8. In the **Computer2.seq** Sequence File window, select **Execute»Single Pass**.

9. Select a test to fail and click **Done**. After the sequence executes, examine the test report. TestStand logs the results of the steps in the subsequence in addition to the steps from the parent sequence.

10. Close the Execution window.

Using Step Templates

The Insertion Palette contains the Step Types list and the Templates list. Use the Templates list to hold copies of steps, variables, and sequences you reuse during the development of sequence files. For example, you can add a step that calls a specific LabVIEW VI you typically use or a sequence that contains common setup steps, cleanup steps, and local variables.

Drag steps from the Steps pane, variables from the Variables pane, and sequences from the Sequences pane and drop them on the Templates list to add steps, variables, or sequences to the Templates list. Use the context menu to rename, copy, paste, delete, import, and export the items in the Templates list. You can drag and drop items to rearrange the list. Select **Insert Folder** from the context menu to add folders to the list. TestStand stores the settings for the Templates list, including any modifications to the list, in the `<TestStand Application Data>\Cfg\Templates.ini` file.

You must drag a sequence, step, or variable from the Templates list to a sequence file, make changes, drag the item back to the Templates list, and delete the original item from the Templates list to edit the item.

Inserting a Template Step

Complete the following steps to import a set of template steps to the Insertion Palette, insert a step from the Templates list in a sequence, and add a new step to the Templates list in the `Computer2.seq` sequence file you created in the previous tutorial.

1. Right-click the **Steps** folder in the Templates list of the Insertion Palette, select **Import** from the context menu, and navigate to the `<TestStand Public>\Tutorial` directory.

2. Select `Tutorial Templates.ini` and click **Open**. The sequence editor adds a Tutorial folder under the Steps folder.

3. Expand the **Tutorial** subfolder of the Steps folder to view the following imported template steps:

    • **Retry if Previous Step Fails**—A Message Popup step that prompts you to retry the previous step on failure.

    • **Open Notepad**—A Call Executable step configured to launch the Microsoft Notepad application.

    • **Output Message**—A statement step that logs an output message in which the logged text is the step name.

    • **Waveform Popup**—A step that calls the WaveformGraphPopup function in the LabWindows/CVI user interface library.

4. In the Templates list, select **Retry if Previous Step Fails** and drag the step below the Power On step on the Steps pane to add the Message Popup step to the sequence.

5. Rename the step `Retry Power On`.

6. Save the changes you made and select **Execute»Single Pass**.

    When the sequence editor launches the Test Simulator dialog box, select the **Power On** test to fail and click **Done** to close the dialog box. TestStand executes the Retry Power On step and launches the Step 'Power On' Failed dialog box because the precondition for the step determined that the previous Power On step failed.

7. Click **Retry** in the Step 'Power On' Failed dialog box to execute the Power On step again. The Power On step fails again, and TestStand launches the Step 'Power On' Failed dialog box again.

8. Click **Continue** in the Step 'Power On' Failed dialog box to continue the execution.

9. Review the report the execution generates. Notice that the Power On step executed twice, and the second call to Retry Power On continued the execution.

> **Note**   You must have Internet Explorer 7.0 or later to view TestStand reports.

10. Close the Execution window.

Creating a Template Step

Complete the following steps to update the Retry Power On step so that it automatically selects the **Continue** button if you do not respond to the prompt in the Computer2.seq sequence file you created in the previous tutorial.

1. In the Sequence File window, select the **Retry Power On** step on the Steps pane.

2. Click the **Text and Buttons** tab of the Step Settings pane. You can also right-click the step and select **Edit Text and Buttons** from the context menu to show the Text and Buttons tab of the Step Settings pane.

3. In the Button Options section on the Text and Buttons tab, select **Button 2** from the Timeout Button ring control.

4. Enter 20 in the Time To Wait control to instruct the step to wait for 20 seconds before continuing. This technique is useful if an operator is not present to acknowledge a non-critical message during testing.

5. Save the changes you made and select **Execute»Single Pass**.

   When the sequence editor launches the Test Simulator dialog box, select the **Power On** test to fail and click **Done** to close the dialog box. TestStand executes the Retry Power On step and launches the Step 'Power On' Failed dialog box. Do not take any action in this dialog box. When the timeout reaches zero, the execution continues as if you clicked Continue.

6. In the Sequence File window, drag the Retry Power On step into the Tutorial folder in the Templates list.

7. Rename the new template step Timeout Retry. You can use the new template step in subsequent development. The sequence editor automatically saves the templates list when you shut down the sequence editor.

8. Close the Computer2.seq sequence file and the Execution window.

9. You no longer need the Tutorial folder in the Templates list, and you can delete the folder if you want. Right-click the **Tutorial** folder in the Templates list and select **Delete** from the context menu to remove the folder.

4

Debugging Sequences

You can debug sequences by tracing, setting breakpoints and conditional breakpoints, stepping through code, and including watch expressions t. Refer to the *Using the Watch View Pane* section of Chapter 5, *Using Variables and Properties*, for more information about watch expressions.

Refer to the *NI TestStand Help* for more information about all the features covered in this chapter.

Note Completed solution files are located in the `<TestStand Public>\Tutorial\Solution` directory.

Step Mode Execution

Note Before you begin this tutorial, select **Configure»Station Options** to launch the Station Options dialog box, click the **Execution** tab, and confirm that the Enable Tracing and Allow Tracing into Setup/Cleanup options are enabled. Click **OK** to close the Station Options dialog box.

Setting a Breakpoint

Complete the following steps to set a breakpoint in a sequence file.

1. Open `<TestStand Public>\Tutorial\Computer2.seq`, which you created in the previous tutorial.

2. Select **File»Save <*filename*> As** and save the sequence as `Computer3.seq` in the `<TestStand Public>\Tutorial` directory.

3. Select **Execute»Break on First Step** to suspend an execution on the first step TestStand executes. A checkmark appears to the left of the menu option to indicate that you enabled this option.

4. Expand the **Cleanup** step group on the Steps pane. Steps in the Cleanup step group execute regardless of whether the sequence completes successfully or whether a run-time error occurs in the sequence. If a step in the Setup or Main step group causes a run-time error to occur or if the operator terminates the execution, the flow of execution stops and jumps to the Cleanup step group. Steps in the Cleanup group always run even when some of the steps in the Setup group do not run. When a step in the Cleanup group causes a run-time error, execution continues to the next step in the Cleanup group.

5. Insert a Message Popup step in the Cleanup step group and rename the step `Cleanup Message`.

6. On the Text and Buttons tab of the Step Settings pane, enter `"I am now in the Cleanup Step Group."`, including the quotation marks, in the Message Expression expression control.

> **Note**   You must enclose literal strings in double quotation marks ("…") in any TestStand expression field.

> **Note**   For this example, use the default values for all the settings on the Options tab and the Layout tab of the Step Settings pane.

7. Save the changes to the sequence file.

Stepping through a Sequence File

When you suspend an execution, you can step through the sequence using the Step Into, Step Over, and Step Out commands on the Debug toolbar, as shown in Figure 4-1.

Figure 4-1. Debug Toolbar Buttons
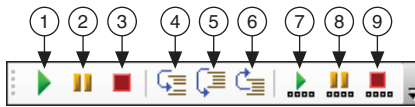
1	Resume	4	Step Into	7	Resume All
2	Break	5	Step Over	8	Break All
3	Terminate Execution	6	Step Out	9	Terminate All

You can find these and other debugging tools in the Debug menu of the sequence editor.

Complete the following steps to step during an execution through the Computer3.seq sequence file you created in the previous tutorial.

1. Select **Execute»Run MainSequence**. After the execution starts, the sequence editor immediately suspends at the first step of the sequence because you enabled the Break on First Step option in step 3 of the previous tutorial. The Execution window tab includes an execution pointer icon to indicate the running state of the execution.

   When the execution suspends, you can step through the sequence using the Step Into, Step Over, and Step Out commands on the Debug toolbar.

2. Click the **Step Over** button to execute the Display Dialog step, which launches the Test Simulator dialog box.

3. Select the **RAM** test to fail and click **Done**. After the Test Simulator dialog box closes, the sequence editor suspends the sequence execution at the end of the Setup step group on `<End Group>`.

4. Insert a breakpoint at the CPU Test step by clicking to the left of the step icon or right-clicking the step and selecting **Breakpoint»Toggle Breakpoint** from the context menu. In the Execution window, a dark red stop sign icon appears to the left of the CPU Test step to indicate the breakpoint.

✍ **Note** Right-click the stop sign icon and select **Breakpoint»Breakpoint Settings** to launch the Breakpoint Settings dialog box, in which you can configure a conditional breakpoint by specifying an expression that must evaluate to `True` to activate the breakpoint. Conditional breakpoints use a bright red stop sign icon in the Sequence File and Execution windows. Disabled breakpoints use a gray stop sign icon. When you globally disable all breakpoints, the breakpoints use a white stop sign icon. Refer to Chapter 5, *Using Variables and Properties*, for more information about expressions.

5. Click the **Resume** button to continue the execution. The sequence editor suspends the execution on the `CPU Test` step.

6. Click the **Step Into** button to step into the `MainSequence` subsequence in `CPU.seq`, and click the **Call Stack** pane. Figure 4-2 shows the Execution window Steps pane and Call Stack pane after you step into the subsequence.

Figure 4-2. Steps Pane and Call Stack Pane while Suspended in Subsequence
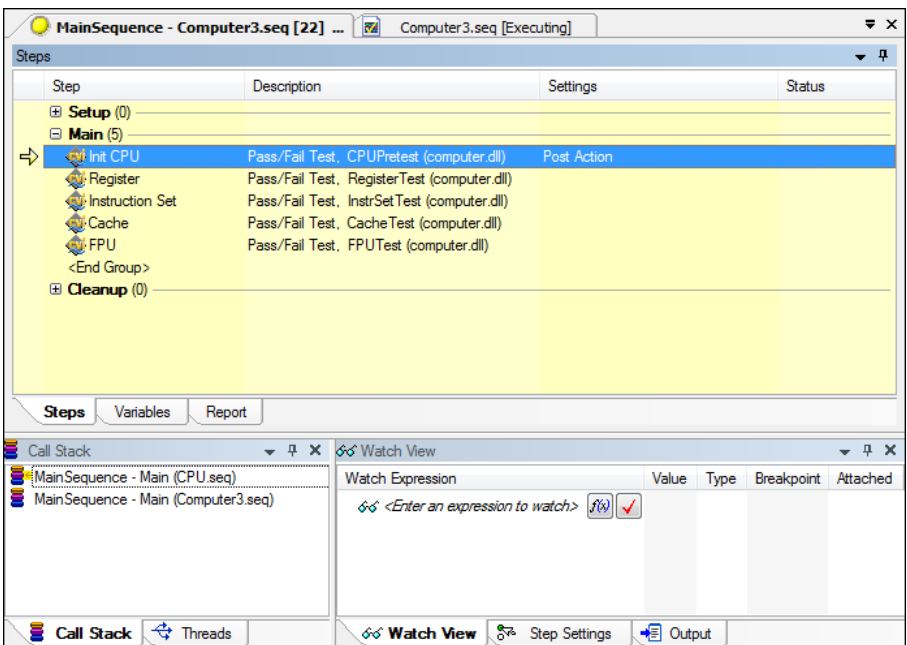
Sequence call steps are similar to subVIs in LabVIEW and function or method calls in C/C++. When a step invokes a subsequence, the sequence that contains the calling step waits for the subsequence to return. The subsequence invocation is nested in the invocation of the calling sequence. The sequence that is currently executing is the most nested sequence. The chain of active sequences that wait for nested subsequences to complete is called the call stack. The first item in the call stack is the most-nested sequence invocation.

The Call Stack pane displays the call stack for the execution thread currently selected on the Threads pane. A yellow pointer icon appears to the right of the most nested sequence invocation while the sequence executes, as shown in Figure 4-2.

7. Select each sequence on the Call Stack pane to view each sequence invocation. Return to the most nested sequence invocation in the call stack, CPU.seq.

8. Click the **Step Over** button to step through the CPU.seq subsequence one step at a time. Before you reach the end of the CPU.seq sequence, click the **Step Out** button. TestStand resumes the execution through the end of the current sequence and suspends the execution at the next step of the calling sequence or breakpoint, whichever comes first.

9. Continue stepping through the Computer3.seq sequence by clicking the **Step Over** button until the Cleanup Message step you added to the Cleanup step group executes. You must step over each of the 10 loops of the Video Test step.

10. Click **OK** to close the Cleanup Message dialog box and click the **Step Over** button to complete the execution. The Execution window dims when the execution completes. Do not close the Execution window.

11. Click the **Restart** button to rerun the execution. The Execution window must be the active window to restart the execution.

12. After the sequence editor suspends the execution on the first step, click the **Terminate Execution** button. TestStand launches the Cleanup Message dialog box even though you terminated the sequence execution. When an operator or run-time error terminates the execution, TestStand proceeds immediately to the steps in the Cleanup step group. Click **OK** to close the Cleanup Message dialog box. The Execution window dims when the execution completes.

13. Click the **Restart** button to rerun the execution. The Execution window must be the active window to restart the execution.

14. After the sequence editor suspends the execution on the first step, select **Debug»Abort All (no cleanup)**. The execution of the sequence immediately stops, and TestStand does not execute any steps in the Cleanup step group.

15. Close the Execution window, save the changes to the sequence file, and then close the sequence file.

5

Using Variables and Properties

You can create and use variables and properties and monitor the values of the variables and properties.

You can define the following types of variables to share data among steps of a sequence or among several sequences:

- Local variables are variables you can use to store data relevant to only the current sequence. Only steps within the sequence which defines the local can access these variables.
- Sequence file global variables are variables you can use to store data relevant to the entire sequence file. Each sequence and step in the sequence file can directly access these global variables.
- Station global variables are variables that persist across different executions. The TestStand Engine maintains the value of station global variables in a file on the computer on which TestStand is running.

In the sequence editor, the Variables pane displays all the variables and properties the selected sequence has access to at run time. In the Execution window, the Variables pane displays the sequence context for the sequence invocation currently selected on the Call Stack pane. The sequence context contains all the variables and properties the steps in the selected sequence invocation can access. Use the Variables pane to examine and modify the values of these variables and properties when an execution is suspended.

Refer to the *NI TestStand Help* for more information about all the features covered in this chapter.

📝 **Note** Completed solution files are located in the `<TestStand Public>\Tutorial\Solution` directory.

Creating Local Variables

Complete the following steps to create and use local variables. You can apply the concepts you learn in this tutorial to sequence file global and station global variables.

1. Open `<TestStand Public>\Tutorial\Computer2.seq`, which you created in Chapter 3, *Editing Steps in a Sequence*.
2. Select **File»Save <*filename*> As** and save the sequence as `Computer4.seq` in the `<TestStand Public>\Tutorial` directory.

3. Click the **Variables** pane in the Sequence File window. Expand the **Locals** item to view the local variables currently defined for `MainSequence`.

    Each sequence includes a ResultList local variable, which is an empty array of container properties. TestStand uses this variable to store step results for reporting and database logging.

4. On the Variables pane, right-click **Locals**, select **Insert Local»Number** from the context menu to insert a new numeric local variable, and rename the local variable `LoopIndex`.

5. Complete the following steps to insert and configure a For step.

    a. Click the **Steps** pane in the Sequence File window and insert a **Flow Control»For** step below the Power On step. Notice that TestStand also adds an End step below the For step.

    b. Drag the **End** step below the RAM step. TestStand automatically indents the Retry Power On, CPU Test, ROM, and RAM steps between the For and End steps.

    c. Click the **For** step and click the **For Loop** tab of the Step Settings pane. You can also right-click the For step and select **Edit For Loop** from the context menu to show the For Loop tab of the Step Settings pane.

    d. Enter 5 in the **Number of Loops** control.

    e. For the Loop Variable control, click the **Expression Browser** button or right-click in the Expression control and select **Browse** from the context menu to launch the Expression Browser dialog box, in which you can interactively build an expression by selecting from lists of variables, properties, operators, functions, and the TestStand API.

        TestStand supports all applicable expression operators and syntax you can use in C, C++, Java, and Visual Basic .NET. You can also call the TestStand API directly from within expressions.

        All TestStand controls that accept expressions provide context-sensitive editing features, such as drop-down lists, syntax checking, and expression coloring to help you create expressions.

    f. Expand the **Locals** item on the Variables/Properties tab of the Expression Browser dialog box. Each item in the top section of the Variables/Properties tab is a property or variable of TestStand.

    g. Select the **LoopIndex** variable under the Locals property and click the **Insert** button. The Expression Browser enters `Locals.LoopIndex` in the Expression control.

        To refer to a subproperty, use a period to separate the name of the property from the name of the subproperty. For example, reference the `LoopIndex` subproperty of the `Locals` property as `Locals.LoopIndex`.

h. Click the **Check Expression for Errors** button to verify that the expression contains valid syntax.

i. In the Expression Browser dialog box, click **OK** to return to the For Loop tab of the Step Settings pane. The Loop Variable control now contains the Locals.LoopIndex expression. Notice that the Custom Loop section shows the expressions TestStand uses when executing the For step when using a fixed number of loops.

6. Save the changes to the sequence file.

7. Select **Execute»Break on First Step** to remove the checkmark that appears to the left of the menu item and disable this option, which you enabled in the *Step Mode Execution* section of Chapter 4, *Debugging Sequences*, of this manual.

8. Select **Execute»Single Pass** to run the sequence. Click **Done** in the Test Simulator dialog box.

9. After the sequence executes, examine the test report, which shows that TestStand executed the steps within the loop (Retry Power On, CPU Test, ROM, and RAM) five times.

Note You must have Internet Explorer 7.0 or later to view TestStand reports.

10. Close the Execution window.

Using the Execution Window Variables Pane

Before executing the steps in a sequence, TestStand creates a run-time copy of the sequence to maintain separate local variable and step property values for each sequence invocation. When an execution completes, TestStand discards the run-time sequence copy.

For each active sequence, TestStand maintains a sequence context that contains references to the run-time copy of the sequence so you can access all the objects, variables, and properties that relate to the execution of the sequence.

The contents of the sequence context vary depending on the currently executing sequence and step, the location of the active sequence in the call stack, and the identity of the execution in which the active sequence resides. Depending on the current state of execution, sequence context subproperties might not exist. When a property exists, the contents of the property can vary.

The Variables pane displays the sequence context for the sequence invocation currently selected on the Call Stack pane. The sequence context contains all the variables and properties the steps in the selected sequence invocation can access. Use the Variables pane to examine and modify the values of these variables and properties when an execution is suspended.

Table 5-1 lists the first-level properties in the sequence context.

Table 5-1. First-Level Properties of the Sequence Context

Sequence Context Subproperty	Description
Locals	Run-time copy of the sequence local variables for the current sequence invocation.
Parameters	Run-time copy of the sequence parameters for the current sequence invocation.
FileGlobals	Run-time copy of the sequence file global variables for the current execution.
StationGlobals	Contains the station global variables for the engine invocation. TestStand maintains a single copy of the station global variables in memory.
ThisContext	Holds a reference to the current sequence context. You typically use this property to pass the entire sequence context as an argument to a subsequence or a step code module.
RunState	Contains properties that describe the state of execution in the sequence invocation.
Step	Run-time copy of the properties in the currently executing step of the current sequence invocation. The Step property exists only while a step executes. The property does not exist when the execution is between steps, such as at a breakpoint.

Complete the following steps to use the Variables pane of the Execution window to examine the value of the LoopIndex variable while TestStand executes the Computer4.seq sequence file you created in the previous tutorial.

1. Insert a breakpoint at the End step associated with the For loop you created in the previous tutorial.

2. Select **Execute»Single Pass** to run the sequence and click **Done** in the Test Simulator dialog box. The execution suspends on the End step.

3. Select the **Variables** pane of the Execution window to show the Variables pane and expand the **Locals** section.

4. Select the **LoopIndex** subproperty under the Locals property. The numeric value of LoopIndex is 0.

5. Select the Steps pane of the Execution window and click the **Resume** button on the Debug toolbar. The execution resumes and suspends at the End step again.

6. Click the **Variables** pane again. The value of Locals.LoopIndex is now 1. Leave the execution in the suspended state for the next tutorial.

Docking the Variables Pane

The sequence editor contains tabbed windows and panes you can float, dock, resize, and hide. Some panes, such as the Variables pane, must stay attached to the associated Sequence File or Execution window.

Complete the following steps to display the Steps and Variables panes at the same time.

1. In the Execution window, click the **Variables** pane tab and drag it to the right using the tab, not the titlebar. As you drag the pane from the current location, the sequence editor detaches the pane and displays docking guides. The docking guides show you where you can place the pane. As you move the mouse over a docking guide, the sequence editor highlights where the pane will relocate when dropped.

2. Drop the pane on the right-most docking guide, as shown in the following figure, so the pane appears to the right of the Steps and Report panes.

3. Click the **Steps** pane tab to hide the Reports pane so you can see the Steps pane and the Variables pane at the same time.

4. Click the **Resume** button. The execution resumes and suspends at the End step again. The value of `Locals.LoopIndex` on the Variables pane is now 2.

5. Drag the titlebar of the Variables pane and drop it on the center docking guide, as shown in the following figure, in the middle of the Steps pane to return the Variables pane to the previous location.

    Notice that the Variables tab now appears to the left of the Steps tab instead of to the right.

6. Click the **Variables** pane tab and drag the tab to the right and drop it on the Steps pane tab to adjust the order of the tabs. Leave the execution in the suspended state for the next tutorial.

Note Select **View»Reset UI Configuration** at any time to restore the panes to the original state.

Using the Watch View Pane

The Watch View pane of the Execution window displays the values of watch expressions you enter. The values in the Watch View pane update when execution suspends at a breakpoint. When you enable tracing, the sequence editor also updates the values also update after each step executes. Viewed and changed watch expression values are highlighted in red.

Enter watch expressions to monitor values of variables and properties as you trace or step through a sequence. You can drag individual variables or properties from the Variables pane to the Watch View pane. When you specify a container property, array property, or a PropertyObject reference as a watch expression, you can use the Watch View pane to expand the subproperties to view the values.

Complete the following steps to create a watch expression that uses the LoopIndex property.

1. Select the **LoopIndex** property on the Variables pane of the Execution window and drag the property to the Watch View pane. The value of the LoopIndex watch expression is 2.

2. Edit the LoopIndex watch expression directly in the Watch Expression column on the Watch View pane to change the expression to the following:

    ```
    Str (Locals.LoopIndex * 20) + "%"
    ```

    You can also right-click the Watch View pane and select **Insert Watch Expression** from the context menu or right-click the expression and select **Edit Watch Expression** from the context menu to launch the Watch Expression Settings dialog box, in which you can create and edit expressions.

3. Resume the execution. When the execution suspends on the End step again, the value of the watch expression changes from 40% to 60%.

4. Click the breakpoint icon to the left of the End step in the Execution window to remove the breakpoint. You can also right-click the step and select **Breakpoint»Toggle Breakpoint** from the context menu to remove the breakpoint.

5. Resume and complete the execution.

6. Close the Execution window.

7. Select **Debug»Breakpoints/Watches** to launch the Edit Breakpoints/Watch Expressions dialog box, in which you can review and edit the breakpoints and watch expressions associated with the current workspace. TestStand saves and reloads breakpoints and watch expressions using an options file associated with the current workspace or a default options file when no workspace is loaded.

8. Review the contents of the Breakpoints and Watch Expressions tabs in the Edit Breakpoints/Watch Expressions dialog box, delete the breakpoint and expression you created for this tutorial, and click **Done** to close the dialog box.

9. Close the sequence file.

6

Using Callback Sequences

Callbacks are sequences TestStand calls under specific circumstances. You can create new callbacks or you can override existing callbacks to customize the operation of the test station.

Refer to the *NI TestStand Help* for more information about all the features covered in this chapter.

📝 **Note** Completed solution files are located in the `<TestStand Public>\Tutorial\Solution` directory.

Process Model Callbacks

The process models contain sequences that define the operations TestStand performs before and after testing a unit under test (UUT). You can use an Execution entry point, such as Test UUTs or Single Pass, to invoke sequences within a process model to execute a sequence in a sequence file you create. The process models also contain Model callbacks, which are sequences you can override to customize the behavior of a process model without editing the process model directly.

The process models define a TestReport callback sequence that generates the test report for each UUT. In most cases, the TestReport callback in the process model file is sufficient because the TestReport callback handles many types of test results. You can, however, override the default TestReport callback with a custom TestReport callback in a sequence file you create. When you use a process model Execution entry point, such as Test UUTs or Single Pass, to run a sequence file, the process model calls the custom TestReport callback in the sequence file instead of the default TestReport callback in the process model.

Figure 6-1 shows the callback sequences the default Sequential process model calls and the order in which TestStand executes the callbacks within the Test UUTs and Single Pass Execution entry points.

Figure 6-1. Sequential Model Callbacks
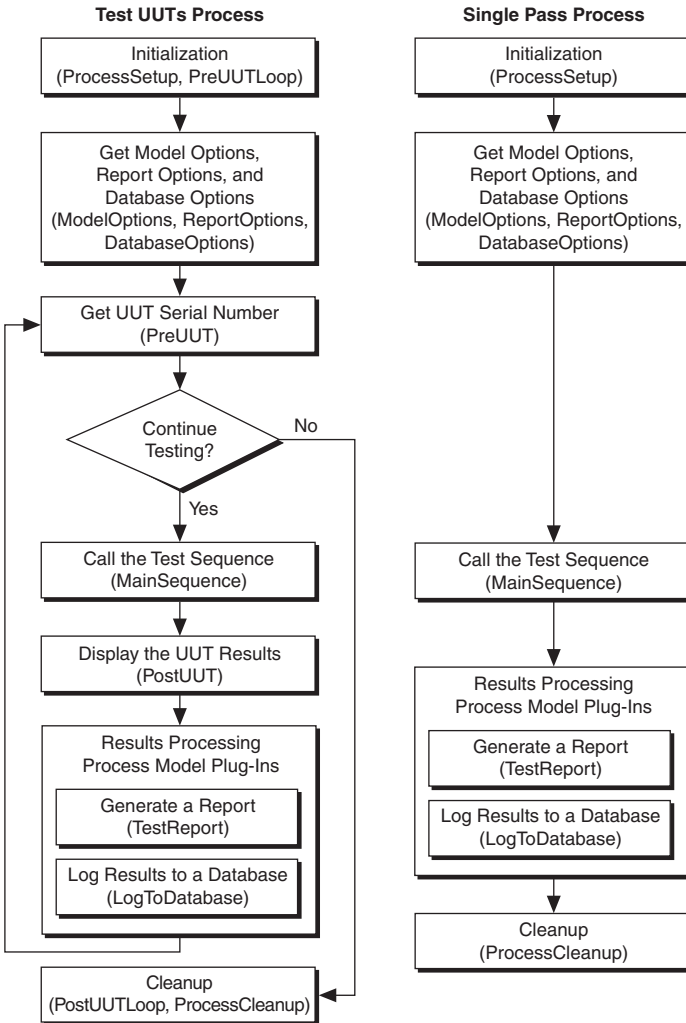
You can modify the process models or replace the models entirely to alter the behavior of the process models for all sequences.

📝 **Note** To modify the installed process models or to create a new process model, copy all the process model files from the `<TestStand>\Components\Models\TestStandModels` directory to the `<TestStand Public>\Components\Models\TestStandModels` directory and make changes to the copy. When you copy installed files to modify, rename the files after you modify them if you want to create a separate custom component. You do not have to rename the files after you modify them if you only want to modify the behavior of an existing component. If you do not rename the files and you use the files in a future version of TestStand, changes National Instruments makes to the component might not be compatible with the modified version of the component. Storing new and customized files in the `<TestStand Public>` directory ensures that installations of the same version of TestStand do not overwrite the customizations and ensures that uninstalling TestStand does not remove the files you customize.

Viewing Process Model Callbacks

Complete the following steps to examine the Model callbacks in the Sequential process model, which is the default process model TestStand uses to execute sequences.

1. Open `<TestStand>\Components\Models\TestStandModels\SequentialModel.seq`. The Model section of the status bar displays the process model the current sequence uses. You can double-click the Model section of the status bar to open that process model file.

2. On the Sequences pane, select **Test UUTs**, which is the Execution entry point TestStand executes when you select **Execute»Test UUTs**. The Main step group of the Test UUTs sequence calls several callback sequences, including the following:

   – PreUUTLoop callback
   – PreUUT callback
   – MainSequence callback
   – PostUUT callback
   – PostUUTLoop callback

   The report generator process model plug-in and the database logger process model plug-in invoke the TestReport callback and the LogToDatabase callback when the process model calls the Model Plug-in - UUT Done entry point and the Model Plugin - Batch Done entry point.

3. Double-click the **PreUUT Callback** step, not the PreUUTLoop Callback step, to view the PreUUT callback sequence, which calls the DoPreUUT sequence. The DoPreUUT sequence includes an IdentifyUUT step and a Set Serial Number step.

4. Open the DoPreUUT sequence and dismiss the DoPreUUT Properties dialog box.

5. Right-click the **IdentifyUUT** step, and select **Run Selected Steps** from the context menu to launch the UUT Information dialog box, which is similar to the dialog box you see when you execute a sequence using the Test UUTs Execution entry point. You can override the

PreUUT callback with a custom callback to change the way TestStand obtains a UUT serial number, such as reading the serial number from a bar code instead.

6.   Click **OK** in the UUT Information dialog box.

7.   Close the Execution window.

8.   In the Test UUTs sequence, double-click the **PreUUTLoop Callback** step to view the PreUUTLoop callback sequence, which is empty because it is a placeholder. If you want to add steps that execute before the UUT loop, override this callback.

9.   Close `SequentialModel.seq` and decline any prompts to save the changes.

Overriding a Process Model Callback

Complete the following steps to override the default PreUUTLoop callback sequence in the Sequential model by creating a PreUUTLoop callback sequence in a client sequence file. Use this technique when you want to perform a task only once before operating on multiple UUTs, such as initializing hardware.

1.   Open `<TestStand Public>\Tutorial\Computer4.seq`, which you created in Chapter 5, *Using Variables and Properties*.

2.   Select **File»Save <*filename*> As** and save the sequence as `Computer5.seq` in the `<TestStand Public>\Tutorial` directory.

3.   Select **Edit»Sequence File Callbacks** to launch the Sequence File Callbacks dialog box and select the **PreUUTLoop** callback.

4.   Click **Add**. The value in the Present column changes from `no` to `yes`. Click **OK** to close the Sequence File Callbacks dialog box. The sequence editor creates a new empty callback sequence in the sequence file. Now, when you start an execution using an Execution entry point, TestStand calls the callback in the sequence file instead of the callback sequence in the Sequential model.

     The PreUUTLoop callback sequence becomes the selected sequence on the Sequences pane and opens in the Sequence File window.

5.   Insert a Message Popup step in the Main step group of the PreUUTLoop callback sequence and rename the new step `Pre UUT Loop Callback Message`.

6.   Enter the literal string `"Now in the Pre UUT Loop Callback"` in the Message Expression control.

7.   Save the changes to the sequence file.

8.   Select **Execute»Test UUTs** to execute the sequence. TestStand launches the Pre UUT Loop Callback Message dialog box.

9.   Click **OK** to close the Pre UUT Loop Callback Message dialog box. TestStand launches the UUT Information dialog box from the PreUUT callback sequence in the Sequential model. Enter a serial number and click **OK**.

10. Run through several iterations of the sequence. TestStand launched the Pre UUT Loop Callback Message dialog box only once at the very beginning of the execution because the

PreUUTLoop callback executes before the loop, and the PreUUT callback executes within the loop.

11. Click **Stop** in the UUT Information dialog box.

12. Close all the windows in the sequence editor.

The Parallel and Batch process models include similar callbacks that you can override.

7

Adding Users and Setting Privileges

Use the TestStand User Manager to maintain the list of users, user names, user passwords, user privileges, groups, group privileges, and members of groups. TestStand can limit the functionality of the TestStand Sequence Editor and User Interfaces depending on the privilege settings you specify in the user manager for the current user and the groups to which the user belongs.

Refer to the *NI TestStand Help* for more information about all the features covered in this chapter.

📝 **Note** Completed solution files are located in the `<TestStand Public>\Tutorial\Solution` directory.

The user manager helps you implement policies and procedures that concern the use of test stations. The user manager is not a security system, and it does not inhibit or control the operating system or third-party applications. Use the system-level security features the operating system provides to secure test station computers against unauthorized use.

Adding a New User

📝 **Note** This tutorial assumes you are currently logged in to TestStand as the `administrator` user. If you are not logged in as administrator, select **File»Login** and select **administrator** from the User Name ring control. Leave the Password field empty and click **OK**.

Complete the following steps to add a new user.

1. Select **View»User Manager** or click the **User Manager** button on the Environment toolbar to show the User Manager window, which shows all the users and groups configured on the test station.

2. Right-click the **Users** item, select **Insert User** from the context menu, and enter a name to create a new user.

3. Right-click the new user you just added, select **Properties** from the context menu to launch the *<User>* Properties dialog box, and complete the following steps.

    a. Confirm that the **User Name** control displays the new name you entered.

    b. Enter the full name in the **Full Name** control.

    c. Enter a password in the **Password** and **Confirm Password** controls.

    d. Select **Operator** in the Group Privileges control.

    e. Click **OK** to close the User Properties dialog box.

4. Save the changes to the user manager.

5. Select **File»Login** to launch the Login dialog box. The User Name ring control now includes the new user you just added.

6. Select the user you just created, enter the appropriate password, and click **OK**.

7. Open `<TestStand Public>\Tutorial\Computer.seq`.

8. Select the **Execute** menu and notice that the Single Pass and Run MainSequence options of the Execute menu are disabled because the user you just created does not have the privileges to execute them.

9. Right-click the **Steps** pane to insert a new step. The Insert Step menu command is also disabled because the user privileges have changed.

10. Select **File»Login** and select **administrator** from the User Name ring control. Leave the Password field empty and click **OK**.

Creating a New Group

You can use the user manager to modify the default groups and to create new groups that define a combination of appropriate privileges.

The default Operator, Technician, Developer, and Administrator groups define a set of privilege settings for the new user to inherit. By default, the Operator group grants a user the privilege to execute, terminate, and abort sequences but does not grant the privilege to create or debug sequences.

Complete the following steps to create a new group.

1. Launch the User Manager window if the window is not open.

2. Expand the **Groups** item to show the four default groups.

3. Right-click the **Operator** group and select **Copy** from the context menu.

4. Right-click the **Groups** item and select **Paste** from the context menu.

5. Rename the new group `Senior Operator`. The new group is identical to the Operator group except for the name.

Modifying Privileges

The TestStand User Manager stores user and group privileges as Boolean properties and organizes the privileges in the following categories:

- **Operate**—Privileges for executing sequences and terminating and aborting executions.
- **Debug**—Privileges for controlling execution flow, executing manual and interactive executions, and editing station global variables and run-time variables.
- **Develop**—Privileges for editing and saving sequence files, editing and saving workspace files, and using source code control.
- **Configure**—Privileges for editing process model files and configuring station options, users, adapters, application settings, and report, database logging, and model options.
- **Custom**—Custom privileges you define. Customize the `NI_UserCustomPrivileges` data type to add new privileges.

You can grant all privileges in a specific category for each user or group in the user manager, and you can grant specific privileges for each user or group. In addition, when you add a user as a member of a group, TestStand grants the user all the privileges of the group. TestStand grants a privilege to a user or group when the property value for the privilege is `True` or when the value of the `GrantAll` property in any enclosing parent privilege category is `True`. For example, a user has the privilege to terminate an execution when one of the following properties is `True`:

- `<User>.Privileges.Operate.Terminate`
- `<User>.Privileges.Operate.GrantAll`
- `<User>.Privileges.GrantAll`
- `<Group>.Privileges.Operate.Terminate`
- `<Group>.Privileges.Operate.GrantAll`
- `<Group>.Privileges.GrantAll`

📝 **Note** TestStand stores the privilege categories as subproperties of the `Privileges` property. The `Privileges` property also includes a Boolean `GrantAll` subproperty. The property `Privileges.GrantAll` applies to all privilege categories. When you set the `GrantAll` property to `True`, the user or group has all privileges. You must set the `GrantAll` property to `False` to honor privilege settings within each privilege category.

Complete the following steps to modify the default privileges for the group you created in the previous section of this tutorial.

1. Expand the **Senior Operator** item and expand the **Privileges** property.
2. Expand the **Debug** item, which is a property that contains Boolean subproperties. Use the **Value** column ring control to change the `SinglePass` property under Debug to `True`.

3. Complete the following steps to add the user you created in the *Adding a New User* section
    of this tutorial to the Senior Operator group.

    a. Right-click the user you previously created under Users and select **Properties** from
        the context menu to launch the <*User*> Properties dialog box.

    b. Disable **Operator** in the Group Privileges control and enable **Senior Operator**
        instead.

    c. Click **OK** to close the User Properties dialog box.

4. Save the changes to the user manager.

5. Select **File»Login** to launch the Login dialog box. Select the user you previously created,
    enter the appropriate password, and click **OK**.

6. Select the Computer.seq window.

7. Select the **Execute** menu. Notice that the Single Pass option is now available, but the Run
    MainSequence option is still disabled because the user you created does not have the
    privilege to execute sequences without a Model entry point.

8. Close all the windows in the sequence editor.

9. Select **File»Login** and select **administrator** from the User Name ring control. Leave the
    Password field empty and click **OK**.

Refer to the CreateDeleteUsers.seq example located in the <TestStand Public>\
Examples\CreateDeleteUsers directory for information about how to use the
TestStand API to add and remove users programmatically.

8

Interactive Executions

When you run steps in interactive mode, you can execute specific steps in a sequence. Use the Interactive Executions section on the Execution tab of the Station Options dialog box to control if an interactive execution records results, runs steps in the Setup and Cleanup step groups, and evaluates preconditions. The controls in the Interactive Executions section also determine how TestStand handles step failures, errors, and branching during interactive executions.

Refer to the *NI TestStand Help* for more information about all the features covered in this chapter.

📝 **Note** Completed solution files are located in the `<TestStand Public>\Tutorial\Solution` directory.

Running Selected Steps as a Separate Execution

Complete the following steps to run selected steps from a Sequence File window as a separate execution.

1. Open `<TestStand Public>\Tutorial\Computer.seq`.

2. Insert a breakpoint at the Power On step.

3. Press <Ctrl> and select the **Power On**, **ROM**, and **ROM Diagnostics** steps.

4. Select **Execute»Run Selected Steps** to start a root interactive execution.

   By default, when you run selected steps from a Sequence File window, TestStand also executes the Setup and Cleanup step groups. You enable or disable the Run Setup and Cleanup option on the Execution tab of the Station Options dialog box to control whether TestStand runs the Setup and Cleanup step groups as part of the root interactive execution.

5. Click **Done** in the Test Simulator dialog box. The execution stops at the Power On step breakpoint. The execution pointer for the interactive execution is a narrow yellow arrow. The sequence editor uses a full yellow arrow as an execution pointer for a normal execution.

6. Step over twice to step through the execution until you reach the ROM Diagnostics step. Only the steps you selected execute. TestStand dims the other steps.

7. Resume and complete the execution.

8. Close the Execution window after the execution completes.

9. Ensure that the Power On, ROM, and ROM Diagnostics steps are selected. Repeat steps 4 through 8, but select **Execute»Run Selected Steps Using»Single Pass** in step 4. TestStand executes the steps you selected using the Single Pass Execution entry point, which generates a unit under test (UUT) report.

10. Close the Execution window after the execution completes.

Running Selected Steps During an Execution

Complete the following steps to interactively execute selected steps in a sequence while suspended at a breakpoint during an execution.

1. Ensure that the Power On, ROM, and ROM Diagnostics steps are selected and select **Execute»Single Pass** to start a new execution.

2. Select the **ROM** test to fail in the Test Simulator dialog box, and click **Done**. The execution stops at the breakpoint on the Power On step.

3. Step through the execution until you reach the RAM Diagnostics step. Notice that the ROM step failed.

4. Place a breakpoint at the ROM step in the Execution window and select the **ROM** and **ROM Diagnostics** steps.

5. Right-click the **ROM Diagnostics** step and select **Loop on Selected Steps** from the context menu to launch the Loop on Selected Steps dialog box. Enter 100 in the Loop Count control and click **OK**.

   TestStand starts an interactive execution of 100 loops for the steps you selected and enters a suspended state at the breakpoint for the ROM step. The execution pointer for the normal execution remains on the ROM Diagnostics step, and an execution pointer for the new interactive execution points to the ROM step.

6. Step through the interactive execution. The interactive execution toggles only between the ROM step and the ROM Diagnostics step. The ROM step continues to fail.

7. Click the **Terminate Interactive Execution** button on the Debug toolbar. TestStand returns the execution to a suspended state on the RAM Diagnostics step.

8. Complete the following steps to force the execution to continue from a step other than the currently suspended step.

   a. Select the **ROM** step so it is the only highlighted step.

   b. Right-click the **ROM** step and select **Set Next Step to Cursor** from the context menu. The execution pointer moves from the RAM Diagnostics step to the ROM step. The execution continues from the ROM step when you resume or step through the sequence.

   c. Step over once. The ROM step executes instead of the RAM Diagnostics step.

9. Resume and complete the execution. The report contains entries for each step you executed interactively.

Note You must have Internet Explorer 7.0 or later to view TestStand reports.

10. Remove the breakpoint from the Power On step in the Sequence File window.

11. Close all the windows in the sequence editor and do not save any changes to the sequence file.

9

Calling Sequences Dynamically

You can add to a sequence a step that dynamically runs one of two sequences, and you can pass parameters to the sequence you call.

Refer to the *NI TestStand Help* for more information about all the features covered in this chapter.

📝 **Note** Completed solution files are located in the `<TestStand Public>\Tutorial\Solution` directory.

Dynamically Specifying a Sequence to Run

Complete the following steps to open an existing sequence, add steps to prompt the operator for a CPU type and the number of CPUs to test, and add a step to call one of two different sequences depending on the type of CPU the user specifies.

1. Open `<TestStand Public>\Tutorial\Computer.seq`.

2. Select **File»Save *<filename>* As** and save the sequence as `Computer6.seq` in the `<TestStand Public>\Tutorial` directory.

3. Click the **Variables** pane in the Sequence File window, right-click the **Locals** item, select **Insert Local»String** from the context menu, and name the local variable `CPUType`.

4. Click the **Steps** pane in the Sequence File window to show the steps in the `MainSequence`.

5. Insert a Message Popup step below the Power On step, rename the new step `Select CPU Type`, and complete the following steps to configure the Select CPU Type step.

   a. On the Text and Buttons tab of the Step Settings pane, enter the following values in the corresponding controls:

      • **Message Expression**—`"Please select the CPU type for the UUT."`

      • **Button 1**—`"INTEL CPU"`

      • **Button 2**—`"AMD CPU"`

      • **Cancel Button**—`None`

   b. On the Layout tab of the Step Settings pane, enable the **Make Modal** option in the Dialog Options section. Enabling this option prevents the sequence editor from hiding the Select CPU dialog box and prevents you from interacting with the sequence editor until you close the Select CPU dialog box.

    c.   On the Properties tab of the Step Settings pane, click **Expressions** to show the Expressions panel.

    d.   In the Post-Expression control or in the Expression Browser dialog box, enter the following expression to assign the string value `"AMD"` or `"INTEL"` to the local variable depending on the button users click:

```
Locals.CPUType = ((Step.Result.ButtonHit == 2) ? "AMD" :
"INTEL")
```

        Click the **Check Expression for Errors** button to verify that the expression contains valid syntax.

6.   On the Steps pane, insert a Message Popup step below the Select CPU Type step, rename the new step `Specify Number of CPUs`, and complete the following steps to configure the Specify Number of CPUs step.

    a.   On the Text and Buttons tab of the Step Settings pane, enter the following values in the corresponding controls:

        •   **Message Expression**—`"Please select the number of CPUs installed for the UUT."`

        •   **Button 1**—`"1"`

        •   **Button 2**—`"2"`

        •   **Button 3**—`"3"`

        •   **Button 4**—`"4"`

        •   **Cancel Button**—None

    b.   On the Layout tab of the Step Settings pane, enable the **Make Modal** option in the Dialog Options section.

7.   Insert a Sequence Call step below the Specify Number of CPUs step, rename the step `CPU Test`, and complete the following steps to configure the CPU Test step.

    a.   On the Module tab of the Step Settings pane, enable the **Specify By Expression** option.

    b.   Enter `Locals.CPUType + "Processor.seq"` in the File Path or Reference control and `"MainSequence"` in the Sequence control.

    c.   Click the **Load Prototype** button to launch the Load Sequence Prototype dialog box, in which you can select the prototype for the Sequence Call step.

    d.   Click the **Browse** button in the Load Sequence Prototype dialog box and select `<TestStand Public>\Tutorial\AMDProcessor.seq`.

    e.   Click **OK** to close the Load Sequence Prototype dialog box. TestStand populates the Parameters Table on the Module tab with the parameter list for the sequence.

f.   Click in the Value column of the **CPUsInstalled** parameter and enter the following expression:

```
RunState.Sequence.Main["Specify Number of
CPUs"].Result.ButtonHit
```

Click the **Check Expression for Errors** button to verify that the expression contains valid syntax.

You can also use the Expression Browser dialog box to locate the property.

8.   Save the changes to the sequence file.

Running a Sequence Dynamically

Complete the following steps to run a sequence dynamically.

1.   Place a breakpoint at the CPU Test step.

2.   Select **Execute»Single Pass** and click **Done** in the Test Simulator dialog box.

3.   Click the **INTEL CPU** button in the Select CPU Type dialog box and click the **2** button in the Specify Number of CPUs dialog box.

4.   When execution suspends at the breakpoint on the CPU Test step, click the **Step Into** button on the Debug toolbar to step into the MainSequence sequence in INTELProcessor.seq.

5.   Click the **Call Stack** pane. The call stack list shows INTELProcessor.seq at the top of the sequence call stack.

6.   Click the **Variables** pane and expand the **Parameters** item. The value of the **CPUsInstalled** parameter equals the value on the button you clicked in the Specify Number of CPUs dialog box. The MainSequence sequence in the INTELProcessor.seq sequence file also requires a **ModelName** parameter. The Sequence Call step you created did not specify the **ModelName** parameter, so the TestStand Engine initializes the parameter to the default name.

7.   Resume and complete the execution. When the execution completes, review the report but do not close the Execution window.

📝   **Note**   You must have Internet Explorer 7.0 or later to view TestStand reports.

8.   Click the **Restart** button on the Debug toolbar to restart the execution and click **Done** in the Test Simulator dialog box. The Execution window must be the active window to restart the execution.

9.   Click the **AMD CPU** button in the Select CPU Type dialog box and click the **3** button in the Specify Number of CPUs dialog box.

10. When the execution suspends at the breakpoint on the CPU Test step, step into the MainSequence sequence in AMDProcessor.seq. The Call Stack pane lists AMDProcessor.seq at the top of the call stack.

11. Resume and complete the execution and review the report.

12. Close all the windows in the sequence editor.

10

Customizing Reports

You can customize report generation within TestStand. Using callback sequences as discussed in Chapter 6, *Using Callback Sequences*, you can create a TestReport callback sequence to generate reports in any format. TestStand also provides several options to configure the format of the test report without creating a callback.

> **Note** You must have Internet Explorer 7.0 or later to view TestStand reports.

Refer to the *NI TestStand Help* for more information about all the features covered in this chapter.

> **Note** Completed solution files are located in the `<TestStand Public>\Tutorial\Solution` directory.

Configuring Test Report Options

Complete the following steps to configure the test report options.

1. Open `<TestStand Public>\Tutorial\Computer.seq`.

2. Select **Configure»Result Processing** to launch the Result Processing dialog box.

3. Click the icon in the Options column for the built-in Report model plug-in to launch the Report Options dialog box and complete the following steps.

    a. On the Contents tab, select **ATML 5.00 Standards Report Document** from the Report Format ring control.

    b. Enable the **Include Step Results** option and confirm the following step result settings:

        • Enable the **Include Test Limits** option.

        • Enable the **Include Measurements** option and select **Insert Graph** from the Include Arrays ring control.

    c. Click the **Edit Format** button to launch the Numeric Format dialog box, in which you specify the format TestStand uses to display the value of a numeric variable or property. By default, TestStand configures the numeric format to report numbers with 13 digits of precision.

    d. Change **Maximum Number of Significant Digits** to 2 and click **OK** to close the Numeric Format dialog box.

4. Click the **Report File Pathname** tab, which you use to specify the report file pathname.
    You can specify a fixed pathname to use for all report files or you can specify options the
    report generator uses to generate report file pathnames. Use the default values for the
    options on this tab and click **OK** to close the Report Options dialog box.

5. Click **OK** to close the Result Processing dialog box.

6. Select **Execute»Test UUTs** to execute the sequence. Run through several iterations of the
    sequence, selecting components other than the Video and CPU tests to fail.

7. Click **Stop** in the UUT Information dialog box to stop sequence execution. The test report
    contains failure chain information for UUTs that fail. The failure chain shows the step
    failure that caused the UUT to fail and shows the Sequence Call steps through which
    the execution reached the failing step. Each step name in the failure chain links to the
    section of the report that shows the result for the step.

8. Close the Execution window.

9. Select **Configure»Result Processing** to launch the Result Processing dialog box.

10. Click the icon in the Options column for the built-in reporting plug-in to launch the Report
    Options dialog box and complete the following steps.

    a. Click the **Contents** tab and select **ASCII Text File** from the Report Format ring
        control.

    b. Select **Exclude Passed/Done/Skipped** from the Result Filtering Expression ring
        control for TestStand to apply a filtering expression to determine what steps appear in
        the report.

        In this example, you configure TestStand to record only the results of the steps that do
        not pass or steps that complete without any status. TestStand also changes the value of
        the Include Arrays ring control to `Insert Table` because ASCII reports do not
        support graphs.

11. Click **OK** to close the Report Options dialog box, and click **OK** to close the Result
    Processing dialog box.

12. Repeat steps 6 through 7 and examine the text version of the test report. Close the
    Execution Window when you are done.

Using External Report Viewers

You can view the test report in external applications more suited for showing and editing text,
such as Microsoft Word or Microsoft Excel.

Complete the following steps to use an external report viewer to view the test report.

1. Select **Configure»External Viewers** to launch the Configure External Viewers dialog box.

2. Click the **Add** button.

3. Click the **Browse** button located to the right of the **Viewer** control and navigate to the
    application you want to use to view the report, such as Word, and click **OK**.

4. Select **txt** from the Format pull-down menu.

5. Enable the **Automatically Launch Viewer** option.

6. Click **OK** to close the Configure External Viewers dialog box.

7. Select **Execute»Test UUTs** to execute the sequence and run through several iterations of the sequence, selecting components other than the Video and CPU tests to fail.

8. Click **Stop** in the UUT Information dialog box to stop sequence execution. TestStand generates the text report and launches the external viewer application you configured in step 3 to show the test report.

9. Examine the test report and close the external report viewing application.

10. Close the Execution window.

11. Complete the following steps to change the report settings back to the default settings.

    a. Launch the Result Processing dialog box.

    b. Enable the **Show More Options** option.

    c. Click the **Reset to Defaults** button.

    d. Click **OK** in the Reset Configuration prompt.

    e. Click **OK** to close the Result Processing dialog box.

12. Complete the following steps to change the external viewer settings back to the default settings.

    a. Launch the Configure External Viewers dialog box.

    b. Select the external viewer application you configured in step 3 and click the **Delete** button.

    c. Click **OK** to close the Configure External Viewers dialog box.

13. Close all the windows in the sequence editor.

Adding Additional Results to Reports

Use the Additional Results panel to add and configure additional results, which are values TestStand adds to the result list of a step when the step executes. An additional result can be a module parameter or a custom additional result in which you specify the name and value of the result. You can configure TestStand to automatically include additional results when generating a report or when logging results to a database. The default TestStand report generator style sheets do not display additional results for skipped steps.

Complete the following steps to create a step that calls a DLL code module to return a numeric array and add the numeric array and other values to the report.

1. Select **File»New»Sequence File** to open a new sequence file.

2. Save the sequence file as `CustomReport.seq` in the `<TestStand Public>\Tutorial` directory.

3. Complete the following steps to create a local variable.

    a. Click the **Variables** pane in the Sequence File window, right-click the **Locals** item and select **Insert Local»Array of»Number** from the context menu to launch the Array Bounds dialog box, in which you can modify the array bounds.

    b. Enter 49 in the Upper Bounds control and click **OK**.

    c. Rename the local variable NumArray.

4. On the Insertion Palette, click the **C/C++ DLL** adapter icon. You can also select the C/C++ DLL Adapter from the Adapter ring control on the Environment toolbar.

5. Click the **Steps** pane of the Sequence File window, add an Action step to the Main step group, and complete the following steps to configure the Action step.

    a. On the Module tab of the Step Settings pane, click the **Browse** button located to the right of the Module control, navigate to <TestStand Public>\Tutorial\ NumericArray.dll to specify the code module for the step, and click **Open**.

    b. Select **GetNumericArray** from the Function ring control.

    c. In the Value Expression column of the **measurements** parameter in the Parameters Table, enter Locals.NumArray to copy the value from the numeric array output parameter to the Locals.NumArray variable when TestStand returns from calling the GetNumericArray function. Click the **Check Expression for Errors** button to verify that the expression contains valid syntax.

    d. In the Value column of the Dim 1 Size property in the Parameter Details Table located to the right of the Parameters Table, enter –1 to specify that TestStand passes all elements of the Locals.NumArray property to the code module.

6. On the Properties tab of the Step Settings pane, complete the following steps to add the numeric array and other values to the report.

    a. Click **Additional Results** to show the Additional Results panel.

    b. Place a checkmark in the **measurements [Out]** parameter checkbox to log the output value of the parameter.

    c. Click the **Add Custom Result** button to add a new empty row to the list of the results.

    d. Enter "Time", including the quotation marks, in the Name column and enter Time() in the Value to Log column. The Time TestStand expression function returns the current time.

    e. Click the **Add Custom Result** button again.

    f. Enter "Date", including the quotation marks, in the Name column and enter Date() in the Value to Log column. The Date TestStand expression function returns the current date.

7. Save the changes you made and select **Execute»Single Pass** to execute the sequence and review the report that includes the array data, the time, and the date.

8. Close all the windows in the sequence editor.

Adding to Reports Using Callbacks

Complete the following steps to add a logo to the header of the HTML report using a Report callback in the process model.

1. Open `<TestStand Public>\Tutorial\Computer.seq`.

2. Select **File»Save *<filename>* As** and save the sequence as `Computer7.seq` in the `<TestStand Public>\Tutorial` directory.

3. Select **Configure»Result Processing** to launch the Result Processing dialog box.

4. Click the icon in the Options column for the built-in reporting plug-in to launch the Report Options dialog box and complete the following steps.

    a. On the Contents tab, select **HTML Document** from the Report Format ring control.

    b. Click **OK** to close the Report Options dialog box.

    c. Click **OK** to close the Result Processing dialog box.

5. Select **Edit»Sequence File Callbacks** to launch the Sequence File Callbacks dialog box, select the **ModifyReportHeader** callback, and click **Add** to add the callback to the sequence file.

6. Click **Edit** to close the Sequence File Callbacks dialog box and edit the new ModifyReportHeader callback sequence in the Sequence File window, which TestStand automatically opens.

7. Click the **Variables** pane, right-click the **Locals** item, select **Insert Local»String** from the context menu, and name the local variable `AddToHeader`. Click in the Value column of the AddToHeader variable and enter the following text:

    ```
    <img alt='Logo Goes Here' src='Logo.jpg'><br><br>
    <a href='http://www.ni.com'>Visit Our Web Site</a><br><br>
    ```

8. Click the **Steps** pane, insert a Statement step in the Main step group, and rename the step `Add Custom Logo`.

9. On the Expression tab of the Step Settings pane, enter the following expression in the Expression control:

    ```
    Parameters.ReportHeader = Locals.AddToHeader +
    Parameters.ReportHeader
    ```

    Click the **Check Expression for Errors** button to verify that the expression contains valid syntax.

10. Save the changes to the sequence file.

11. Select **Execute»Single Pass** and click **Done** in the Test Simulator dialog box. Review the report after the execution completes and notice the new logo image at the top of the UUT report.

12. Complete the following steps to change the report settings back to the default settings.

    a.   Launch the Result Processing dialog box.

    b.   Enable the **Show More Options** option.

    c.   Click the **Reset to Defaults** button.

    d.   Click **OK** in the Reset Configuration prompt.

    e.   Click **OK** to close the Result Processing dialog box.

13. Close all the windows in the sequence editor.

A

Technical Support and Professional Services

Log in to your National Instruments `ni.com` User Profile to get personalized access to your services. Visit the following sections of `ni.com` for technical support and professional services:

- **Support**—Technical support at `ni.com/support` includes the following resources:

  - **Self-Help Technical Resources**—For answers and solutions, visit `ni.com/support` for software drivers and updates, a searchable KnowledgeBase, product manuals, step-by-step troubleshooting wizards, thousands of example programs, tutorials, application notes, instrument drivers, and so on. Registered users also receive access to the NI Discussion Forums at `ni.com/forums`. NI Applications Engineers make sure every question submitted online receives an answer.

  - **Standard Service Program Membership**—This program entitles members to direct access to NI Applications Engineers via phone and email for one-to-one technical support, as well as exclusive access to eLearning training modules at `ni.com/elearning`. All customers automatically receive a one-year membership in the Standard Service Program (SSP) with the purchase of most software products and bundles including NI Developer Suite. NI also offers flexible extended contract options that guarantee your SSP benefits are available without interruption for as long as you need them. Visit `ni.com/ssp` for more information.

    For information about other technical support options in your area, visit `ni.com/services`, or contact your local office at `ni.com/contact`.

- **Training and Certification**—Visit `ni.com/training` for training and certification program information. You can also register for instructor-led, hands-on courses at locations around the world.

- **System Integration**—If you have time constraints, limited in-house technical resources, or other project challenges, National Instruments Alliance Partner members can help. To learn more, call your local NI office or visit `ni.com/alliance`.

You also can visit the Worldwide Offices section of `ni.com/niglobal` to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

Index