NI TestStand™ 2: Framework Development Exercises

Course Software Version 2012
May 2013 Edition
Part Number 324607J-01

Copyright
© 2001–2013 National Instruments. All rights reserved.
Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

National Instruments respects the intellectual property of others, and we ask our users to do the same. NI software is protected by copyright and other intellectual property laws. Where NI software may be used to reproduce software or other materials belonging to others, you may use NI software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

End-User License Agreements and Third-Party Legal Notices
You can find end-user license agreements (EULAs) and third-party legal notices in the following locations:
• Notices are located in the <National Instruments>_Legal Information and <National Instruments> directories.
• EULAs are located in the <National Instruments>_Shared\MDP\Legal\license directory.
• Review <National Instruments>_Legal Information.txt for more information on including legal information in installers built with NI products.
**Trademarks**

Refer to the *NI Trademarks and Logo Guidelines* at ni.com/trademarks for more information on National Instruments trademarks.

ARM, Keil, and µVision are trademarks or registered of ARM Ltd or its subsidiaries.

LEGO, the LEGO logo, WEDO, and MINDSTORMS are trademarks of the LEGO Group. ©2013 The LEGO Group.

TETRIX by Pitsco is a trademark of Pitsco, Inc. ©2013

FIELDBUS FOUNDATION™ and FOUNDATION™ are trademarks of the Fieldbus Foundation.

EtherCAT® is a registered trademark of and licensed by Beckhoff Automation GmbH.

CANopen® is a registered Community Trademark of CAN in Automation e.V.

DeviceNet™ and EtherNet/IP™ are trademarks of ODVA.

Go!, SensorDAQ, and Vernier are registered trademarks of Vernier Software & Technology. Vernier Software & Technology and vernier.com are trademarks or trade dress.

Xilinx is the registered trademark of Xilinx, Inc.

Taptite and Trilobular are registered trademarks of Research Engineering & Manufacturing Inc.

FireWire® is the registered trademark of Apple Inc.

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

Handle Graphics®, MATLAB®, Real-Time Workshop®, Simulink®, Stateflow®, and xPC TargetBox® are registered trademarks, and TargetBox™ and Target Language Compiler™ are trademarks of The MathWorks, Inc.

Tektronix®, Tek, and Tektronix, Enabling Technology are registered trademarks of Tektronix, Inc.

The Bluetooth® word mark is a registered trademark owned by the Bluetooth SIG, Inc.

The ExpressCard™ word mark and logos are owned by PCMCIA and any use of such marks by National Instruments is under license.

The mark LabWindows is used under a license from Microsoft Corporation. Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

Other product and company names mentioned herein are trademarks or trade names of their respective companies.

Members of the National Instruments Alliance Partner Program are business entities independent from National Instruments and have no agency, partnership, or joint-venture relationship with National Instruments.

**Patents**

For patents covering National Instruments products/technology, refer to the appropriate location: Help>Patents in your software, the patents.txt file on your media, or the National Instruments Patent Notice at ni.com/patents.
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
To comment on National Instruments documentation, refer to the National Instruments Web site at ni.com/info and enter the Info Code feedback.
## Contents

**Student Guide**

A. NI Certification........................................................................................................................................................................................vii  
B. Course Description ..................................................................................................................................................................................vii  
C. What You Need to Get Started ..............................................................................................................................................................viii  
D. Installing the Course Software ...............................................................................................................................................................ix  
E. TestStand Directories .............................................................................................................................................................................ix  
F. Course Goals .........................................................................................................................................................................................x  

**Lesson 1**  
**Test Frameworks**

<table>
<thead>
<tr>
<th>Exercise</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Reviewing a Framework Requirements Document</td>
<td>1-2</td>
</tr>
</tbody>
</table>

**Lesson 2**  
**TestStand API**

<table>
<thead>
<tr>
<th>Exercise</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>Calling the TestStand API</td>
<td>2-2</td>
</tr>
<tr>
<td>2-2</td>
<td>Dynamically Constructing a Sequence</td>
<td>2-14</td>
</tr>
</tbody>
</table>

**Lesson 3**  
**Custom Steps**

<table>
<thead>
<tr>
<th>Exercise</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-1</td>
<td>Creating a Custom Step Type</td>
<td>3-2</td>
</tr>
<tr>
<td>3-2</td>
<td>Merging and Using Custom Step Types</td>
<td>3-21</td>
</tr>
</tbody>
</table>

**Lesson 4**  
**Process Models**

<table>
<thead>
<tr>
<th>Exercise</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-1</td>
<td>Adding Entry Points</td>
<td>4-2</td>
</tr>
<tr>
<td>4-2</td>
<td>Customizing Process Model Prompts</td>
<td>4-10</td>
</tr>
<tr>
<td>4-3</td>
<td>Customizing Data Collection and Report Generation (Optional)</td>
<td>4-30</td>
</tr>
<tr>
<td>4-4</td>
<td>Creating a Custom Process Model Plug-in (Optional)</td>
<td>4-36</td>
</tr>
</tbody>
</table>
## Lesson 5
### User Interfaces

<table>
<thead>
<tr>
<th>Exercise 5-1 A</th>
<th>Creating a LabVIEW User Interface</th>
<th>5-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exercise 5-1 B</td>
<td>Creating a LabWindows/CVI User Interface</td>
<td>5-11</td>
</tr>
<tr>
<td>Exercise 5-2 A</td>
<td>Using UI Messages in LabVIEW</td>
<td>5-21</td>
</tr>
<tr>
<td>Exercise 5-2 B</td>
<td>Using UI Messages in LabWindows/CVI</td>
<td>5-33</td>
</tr>
</tbody>
</table>

### Appendix A

<table>
<thead>
<tr>
<th>Exercise A-1</th>
<th>Logging Additional Results</th>
<th>A-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exercise A-2</td>
<td>Querying the Database</td>
<td>A-22</td>
</tr>
</tbody>
</table>

### Appendix B

#### Challenge Exercises

<table>
<thead>
<tr>
<th>Exercise B-1</th>
<th>Dynamic Sequence Calling</th>
<th>B-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exercise B-2</td>
<td>Creating a Simple Process Model</td>
<td>B-3</td>
</tr>
<tr>
<td>Exercise B-3</td>
<td>Passing Parameters to New Executions</td>
<td>B-4</td>
</tr>
<tr>
<td>Exercise B-4</td>
<td>Time Delay Custom Step Type</td>
<td>B-5</td>
</tr>
<tr>
<td>Exercise B-5</td>
<td>Multithreading and Synchronization</td>
<td>B-6</td>
</tr>
<tr>
<td>Exercise B-6</td>
<td>Creating a Sequence Builder Tool</td>
<td>B-8</td>
</tr>
</tbody>
</table>
4

Process Models

Exercises

<table>
<thead>
<tr>
<th>Exercise</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exercise 4-1</td>
<td>Adding Entry Points</td>
</tr>
<tr>
<td>Exercise 4-2</td>
<td>Customizing Process Model Prompts</td>
</tr>
<tr>
<td>Exercise 4-3</td>
<td>Customizing Data Collection and Report Generation (Optional)</td>
</tr>
<tr>
<td>Exercise 4-4</td>
<td>Creating a Custom Process Model Plug-in (Optional)</td>
</tr>
</tbody>
</table>
Exercise 4-1  Adding Entry Points

Goal
Add a Configuration entry point and an Execution entry point to a process model.

Scenario
The NI TestStand 2 CD Test sequence currently uses the default sequential process model. This process model is acceptable, but creating a custom process model to handle general CD tests would make the NI TestStand 2 CD Test sequence more usable and maintainable.

Design
The new process model should provide the following features:

- **Configuration Entry Point**—A configuration dialog box to specify the settings for the test. Currently, you change the test settings by directly editing a station global variable. This is inconvenient and counter-intuitive for the user. In addition, only users with permission to edit station global variables can change test settings. Create a configuration entry point to display the configuration dialog box. The provided CD Test Settings Dialog VI displays an example of this dialog box. If the user does not cancel the dialog box, apply the settings to the station global variable and save the StationGlobals.ini file using a TestStand API call.

- **Execution Entry Point**—An Execution entry point customized for the actions necessary to test a general CD. This allows you to remove several steps from your CD-specific test sequence and reduce unnecessary items in the report. It also makes it easier to create sequences for new CDs. In this exercise, you create the Execution entry point and modify it in subsequent exercises to customize it for testing CDs.

Implementation
1. Create a Configuration entry point.
   - Open `<TestStand>\Components\Models\TestStandModels\SequentialModel.seq`.
   - Tip You can open the process model used by any sequence by double-clicking the name of the process model in the status bar at the bottom of the Sequence Editor window.
   - Save the file as `<Exercises>\TestStand 2\CD Test\CD Test Model.seq`.
   - Right-click in the Sequences pane and select **Insert Sequence**.
   - Name the sequence **Configure CD Test Settings**.
☐ Right-click the Configure CD Test Settings sequence and select **Sequence Properties**.

☐ Enable the **Disable Result Recording for All Steps** checkbox.

☐ Select the **Model** tab.

☐ Select **Configuration Entry Point** from the Type pull-down menu.

☐ Enter "**CD Test Settings**" in the Entry Point Name Expression textbox.

☐ Enable the **Entry Point Ignores Client File** checkbox.

☐ Enable the **Hide Entry Point Execution** checkbox.

☐ Disable the **Save Modified Sequence Files Before Execution** checkbox.
☐ Enable the **Show Entry Point for All Windows** checkbox.

![Figure 4-1. Configure CD Test Settings Properties](image)

☐ Click **OK** to close the Configure CD Test Settings Properties dialog box.
2. Create a simulation dialog box.

- Select **Configure CD Test Settings** in the Sequences pane.
- Add a LabVIEW Action step to the Main step group.
- Enter **Display Test Settings Dialog** as the step name.
- Configure the module tab on the Step Settings pane, as shown in Figure 4-2.

**Figure 4-2. Display Test Settings Dialog Module Settings**

**Tip** Use the Variables pane or the Expression Browser to create a `DialogCancelled` local variable with a Boolean type. In the Expression Browser, right-click **Locals**, select **Insert Boolean**, and name the variable `DialogCancelled`.
Configure a Post-Expression for the step, as shown in Figure 4-3.

**Figure 4-3.** Display Test Settings Dialog Post-Expression

The Post-Expression checks the `DialogCancelled` local variable that was set by the parameter from the code module. If the user did not cancel the dialog box, the expression calls the `Engine.CommitGlobalsToDisk` method, which saves the `PropertyObjectFile` that contains the global variables to disk. If the user cancelled the dialog box, the global variables did not change, so a False constant tells the expression to do nothing.

3. Create an Execution entry point.
   - Right-click the Test UUTs sequence in the Sequences pane and select **Copy**.
   - Right-click the Test UUTs sequence in the Sequences pane and select **Paste**.
   - Name the new sequence **CD Test**.
   - Right-click the new sequence and select **Sequence Properties**.
   - Delete the text in the Comment textbox.
   - Select the **Model** tab.
   - Select **Execution Entry Point** from the Type pull-down menu.
Enter "CD Test" in the Entry Point Name Expression textbox.

**Figure 4-4. CD Test Properties**

Click **OK** to close the Configure CD Test Settings Properties dialog box.
4. Configure the NI TestStand 2 CD Test to use the new process model.
   □ Save the process model.
   □ Open `<Exercises>\TestStand 2\CD Test\TS2 CD Test.seq` if it is not already open.
   □ Select `Edit»Sequence File Properties`.
   □ Select the `Advanced` tab.
   □ Select `Require Specific Model` from the Model Option pull-down menu.
   □ Click the `Browse` button and select `<Exercises>\TestStand 2\CD Test\CD Test Model.seq` in the Model File textbox.
   □ Click `OK` to close the dialog box.

5. Save the sequence file.

Testing
1. Test the Configuration entry point.
   □ Expand TestSettings Station Globals variable from the Variables Pane.
   □ Notice that the `LogIgnoreErrors` is set to `False`.
   □ Select `Configure»CD Test Settings`.
   □ Verify that the CD Test Settings dialog box launches.
   □ Enable the `Include Ignored Errors in Results` checkbox.
   □ Click `OK` to close the dialog box.
   □ Verify that `LogIgnoreErrors` is set to `True`.
   □ Select `Configure»CD Test Settings`.
   □ Verify the `Include Ignored Errors in Results` checkbox is enabled.
- Disable the **Simulate Tests** checkbox.
- Click **Cancel** to close the dialog box.
- Expand the **TestSettings** Station Global variable from the Variables pane.
- Verify that **Simulate** is set to **True**.

2. Test the Execution entry point.
   - Select **Execute»CD Test**.
   - Verify that the CD Test entry point behaves like the Test UUTs entry point.

**End of Exercise 4-1**
Exercise 4-2  Customizing Process Model Prompts

Goal
Customize an Execution entry point to reflect the operation of a particular type of test and to include prompts to the user. Identify items that are not specific to a UUT and move them to the process model.

Scenario
The CD Test system has several operations in the test sequence that apply to general CD tests rather than specific to testing a particular CD. This causes two difficulties.

• Additional operations are entered into the test results, thus including unnecessary information. You could change the settings of each of the steps, but it is preferable to move them to the test framework where the issue will not arise.

• Operations must be written for each type of CD you want to test. They must also be maintained for each of the test sequences. As an example, consider what would happen if you wrote many CD test sequences in the current format and then decided that the prompt telling the operator to place the CD under the camera is not specific enough. You would have to modify each sequence to correct the prompt.

Move any operations that are not UUT-specific to the test framework to improve the maintainability and usability of the test system.

Design
Identify the operations that are UUT-specific and those that apply to testing all CDs. Move operations that are not UUT-specific to the process model. Keep the following in mind.

• If two or more UUTs share an operation, it does not mean that all UUTs share the same operation. For example, the NI TestStand 1 and NI TestStand 2 course CDs perform the same tests with different limits to verify the CD label. However, other CDs may have different labels with different test requirements.

• Some operations are already customized by the process model for the current CD, such as generating custom error codes or generating the serial number programmatically. These operations represent specific customizations for this particular type of CD.
Table 4-1 describes the various operations performed by the current CD Test sequence and identifies those operations which apply to a specific CD and those which apply to testing CDs in general.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display simulation settings dialog box</td>
<td>General</td>
</tr>
<tr>
<td>Prompt the operator to insert CD in the drive</td>
<td>General</td>
</tr>
<tr>
<td>Tests to verify the contents of the CD</td>
<td>UUT-Specific</td>
</tr>
<tr>
<td>Prompt the operator to place the CD under the camera</td>
<td>General</td>
</tr>
<tr>
<td>Tests to verify the CD label</td>
<td>UUT-Specific</td>
</tr>
<tr>
<td>Create test settings on load</td>
<td>General</td>
</tr>
<tr>
<td>Convert custom errors to error messages</td>
<td>UUT-Specific</td>
</tr>
<tr>
<td>Programmatically determine serial number</td>
<td>UUT-Specific</td>
</tr>
</tbody>
</table>

Move each of the general operations to the process model. Use a callback sequence for each operation, so that you can override the process model functionality for specific CDs if necessary.

The prompt to place the CD under the camera is part of a subsequence called from the MainSequence. To move this prompt to the process model, you must disconnect the CD label test from the CD contents test, which is the current MainSequence. You can convert the MainSequence callback to a normal sequence and include two callbacks within that sequence to process the two separate tests. This is appropriate because the CD label test is not a subset of the CD contents test, but rather a separate set of operations to perform in the process of testing a CD.

In Part A: Replace MainSequence Callback with Contents Test and Label Test Callbacks, you replace the MainSequence callback with Contents Test and Label Test callbacks. In Part B: Move Simulation Settings to the Process Model, you move the Simulation Settings dialog to the process model because it applies to testing CDs in general. In Part C: Move Remaining General Operations to the Process Model, you move the rest of the general testing operations, such as the user prompts, to the process model.
Part A: Replace MainSequence Callback with Contents Test and Label Test Callbacks

Implementation
In the previous exercise, you created a new process model and created entry points within that model. In Part A of this exercise, you will begin laying the foundation of your customized test framework.

1. Open <Exercises>\TestStand 2\CD Test\TS2 CD Test.seq if it is not already open.
2. Open <Exercises>\TestStand 2\CD Test\CD Test Model.seq if it is not already open.
3. Copy the CDSimulationSettings type to the process model.
   - Click the Types button to view the Types pane.
   - Select TS2 CD Test.seq from the View Types For list.
   - Right-click the CDSimulationSettings type under Custom Data Types and select Copy.
   - Select MyTypes.ini from the View Types For list.
   - Right-click Custom Data Types and select Paste.
   - Click OK in the Paste Type dialog box.
     Because the process model is not inside the TS2 CD Test.seq sequence file, you must move this type to a Type Palette so that both sequence files can use it.
   - Save MyTypes.ini.
   - Close the Types pane.
4. Create a local variable in the process model to transfer the simulation settings from the PreUUT callback to MainSequence.
   - Select CD Test from the Sequences pane in the CD Test Model.seq sequence file.
   - Select the Variables pane.
   - Right-click Locals and select Insert Local»Type»CDSimulationSettings.
1. Name the new local variable `SimulationSettings`.
2. Set `Locals»SimulationSettings»FilesDirectory` to `<Exercises>\TestStand 2\CD Test\Simulated CD Drive\Passing CD 1`.
3. Set `Locals»SimulationSettings»ImageFile` to `<Exercises>\TestStand 2\CD Test\Images for Simulation\Passing\Pass 001.jpg`. This will be the initial value of the `ImageFile` local variable when the CD Test Execution entry point begins execution.
4. Confirm that your variables look similar to Figure 4-5.

**Figure 4-5. CD Test Execution Entry Point Local Variables**

5. Create a parameter in the PreUUT callback to transfer the simulation settings.
   - In the Sequences pane, select the PreUUT callback.
   - In the Variables pane, right-click Parameters and select Insert Parameter»Type»CDSimulationSettings.
   - Name the parameter `SimulationSettings`.

   **Note** The client TS2 CD Test sequence file overrides this PreUUT callback. Later, you update the PreUUT callback in the client TS2 CD Test sequence file to pass the value of the simulation settings back to the CD Test Execution entry point using the parameter you created in this step.

6. Use the local variable in the PreUUT callback in the process model
   - In the Sequences pane, select the CD Test Execution entry point.
   - In the Steps pane, select the PreUUT Callback step.
In the Module tab, click the icon to reload the sequence prototype, which updates the PreUUT Callback step parameters to include the SimulationSettings parameter.

In the Module tab, set the SimulationSettings parameter to Locals.SimulationSettings.

In your client sequence file, there are two main organizations of steps for testing the contents of the CD and the printing on the CD’s face. Therefore, there is not one set of steps that you can refer to as the Main Sequence. So, next you will configure the process model to treat the MainSequence as a normal sequence that calls more specific sequence callbacks called “Contents Test” and Label Test”.

7. Change the MainSequence callback to a normal subsequence.
   - In the Sequences pane, right-click MainSequence and select Sequence Properties.
   - Select the Model tab.
   - Select Normal from the Type pull-down menu and click OK.

8. Create a parameter in the MainSequence subsequence to receive the simulation settings from the CD Test execution entry point.
   - Verify that MainSequence is selected in the Sequences pane.
   - In the Variables pane, right-click Parameters and select Insert Parameter » Type » CDSimulationSettings.
   - Name the parameter SimulationSettings.

9. Call the MainSequence from the process model instead of the client sequence file.
   - In the Sequences pane, select the CD Test Execution entry point.
   - In the Steps pane, select the MainSequence Callback step.
Figure 4-6. MainSequence Callback Step

Select the **Module** tab from the Step Settings pane.

Enable the **Use Current File** checkbox.

Click the **Load Prototype** button .

In the Load Sequence Prototype window, set **Sequence** to **MainSequence** and click **OK**.

Disable the **Specify By Expression** checkbox.
Select the **MainSequence** from the **Sequence** pull-down menu.

Verify that the **SimulationSettings** parameter appears in the Module tab.

Set the value of the **SimulationSettings** parameter to `Locals.SimulationSettings`.

This parameter allows you to pass the value of the **SimulationSettings** local variable in the CD Test Execution entry point in the process model to the **MainSequence** subsequence in the CD Test Execution entry point.

Select the **Properties** tab from the Step Settings pane.

Select the **Run Options** category.

Select **Disable tracing in sequence** from the Sequence Call Trace Setting pull-down menu.

10. Check for client server files.

In the Sequences pane, select the **CD Test Execution Entry point**.

In the Variables pane, set the value of **Parameters.Sequence** to **Contents Test**.

This parameter is used by the process model and other support files to determine the currently running client sequence. By setting the default value to “Contents Test” here, you are instructing TestStand to verify, during the Initialize Entry Point step, that the client sequence file includes the Contents Test sequence before executing the entry point.

In the Steps pane, right-click the **Initialize Entry Point** step and select **Open Sequence**.

This opens **ModelSupport.seq** in a new window. **ModelSupport.seq** contains utility sequences that process models call. In this case, **ModelSupport.seq** contains a Statement that we’ll need in the CD Test process model.

In the Steps pane, right-click the **Check For Target Sequence** step and select **Copy**.

Close the **ModelSupport.seq** window.

In the CD Test Model.seq window, select the Steps pane.

Right-click the **Initialize Entry Point** step and select **Paste**.

Name the new step **Check for Label Test Sequence**.
Select the **Check for Label Test Sequence** step.

Select the **Properties** tab in the Step Settings pane.

Select the **Preconditions** category from the Properties tab.

Change `Parameters.sequence` in the Precondition Expression to "Label Test".

Select the **Expression** tab from the Step Settings pane.

Change `Parameters.sequence` in the first line of the expression to "Label Test".

This expression verifies that the client sequence file includes a Label Test sequence before executing the entry point.
11. Add a Contents Test callback.
   □ Right-click **MainSequence** in the Sequences pane and select **Insert Sequence**.
   □ Name the new sequence **Contents Test**.
   □ Right-click **Contents Test** in the Sequences pane and select **Sequence Properties**.
   □ Select the **Model** tab.
   □ Select **Callback** from the Type pull-down menu and click **OK**.
   □ Verify that **Contents Test** in the Sequences pane is selected.
   □ In the Variables pane, right-click **Parameters** and select **Insert Parameter»Type»CDSimulationSettings**.
   □ Name the parameter **SimulationSettings**.
   This parameter will allow the **MainSequence** subsequence to pass the simulation settings to the **Contents Test** callback.

   **Note** If there is a sequence named Contents Test in the client sequence file, it will override this sequence.

12. Add a Label Test callback.
   □ Right-click **Contents Test** in the Sequences pane and select **Copy**.
   □ Right-click **Contents Test** in the Sequences pane and select **Paste**.
   This places a copy of the sequence directly below the first one.
   □ Name the new sequence **Label Test**.

13. Create the **MainSequence**.
   □ Select **MainSequence** from the Sequences pane.
   □ Insert a Sequence Call step in the Main step group.
   □ Label the step **Contents Test Callback**.
Enable the **Use Current File** checkbox on the Module tab.

Select **Contents Test** from the Sequence pull-down menu.

Set the value of the **SimulationSettings** parameter to `Parameters.SimulationSettings`.

Select the **Properties** tab from the Step Settings pane.

Select the **Run Options** category from the Properties tab.

Select **Use initial execution setting** from the Sequence Call Trace Setting pull-down menu.

Insert a Sequence Call step below the Contents Test Callback step.

Label the step **Label Test Callback**.

On the Module tab, enable the **Use Current File** checkbox.

Select **Label Test** from the Sequence pull-down menu.

Set the value of the **SimulationSettings** parameter to `Parameters.SimulationSettings`.

Select the **Properties** tab from the Step Settings pane.

Select the **Run Options** category from the Properties tab.

Select **Use initial execution setting** from the Sequence Call Trace Setting pull-down menu.

14. Save **CD Test Model.seq**.

15. Modify the client test sequence to reflect the changes in the process model.

   Open `<Exercises>\TestStand 2\CD Test\TS2_CD Test.seq` if it is not already open.

   Observe that the color of the **MainSequence** icon in the Sequences pane has changed from green to blue, indicating that it is no longer a callback.

   Right-click **MainSequence** in the Sequences pane and select **Rename**.
Rename the sequence Contents Test.
Observe that the icon color changes back to green.

Verify that the Contents Test sequence is selected in the Sequences pane.

In the Variables pane, drag the SimulationSettings variable from the FileGlobals container to the Parameters container.

In the Variables pane, right-click Parameters»SimulationSettings and select Pass By Reference to enable this option.

Note Passing the SimulationSettings parameter by reference allows the Contents Test callback to change the actual value of the corresponding property in a sequence that calls the Contents Test callback.

Right-click CD Image Recognition in the Sequences pane and select Rename.

Rename the sequence Label Test.

Select Contents Test from the Sequences pane.

Delete the Call CD Image Recognition step from the Steps pane.

16. Update the client test sequence file to get the simulation settings from parameters rather than file global variables.

In the Sequences pane, select the Contents Test sequence.

In the Steps pane, select the Display Simulation Dialog step.

In the Step Settings pane, select the Module tab.
Replace the SimulationSettings file global variables with **SimulationSettings** parameters in the step settings of the Display Simulation Dialog step, as shown in Figure 4-8.

**Figure 4-8. Settings for Display Simulation Dialog**

- In the Steps pane, select **Volume Label Test**.
- In the Step Settings pane, select the **Module** tab.
  This expression checks the value of the Simulate process model setting to determine whether the user is currently simulating the test. If the user is simulating the test, this expression passes the file directory configured in the simulation dialog box. If the user is not simulating the test, this expression passes the drive letter from the process model settings.
- In the Steps pane, select **File Count Test**.
- In the Step Settings pane, select the **Module** tab.
17. Update the PreUUT callback in the client test sequence file to get the simulation settings from parameters rather than file global variables.

- In Sequences pane, select **PreUUT**.
- In the Variables pane, right-click **Parameters** and select **Insert Parameter»Type»CDSimulationSettings**.
- Name the parameter **SimulationSettings**.
- In the Steps pane, select the **Read Serial Number** step.
- In the Step Settings pane, select the **Module** tab.
- Change the value of the **CD Files Directory** parameter to **Parameters.SimulationSettings.FilesDirectory**

18. Save **TS2 CD Test.seq**.

**Testing**

1. Verify that the Contents Test and Label Test sequences from **TS2 CD Test.seq** execute as they are called from the process model.

- Open `<Exercises>\TestStand 2\CD Test\CD Test Model.seq` if it is not already open.
- Select **MainSequence** from the Sequences pane.
- Right-click the **Contents Test Callback** step and select **Breakpoint»Toggle Breakpoint**.
- Right-click the **Label Test Callback** step and select **Breakpoint»Toggle Breakpoint**.
- Open `<Exercises>\TestStand 2\CD Test\TS2 CD Test.seq` if it is not already open.
- Select **Execute»CD Test**.

**Note**  The CD Test entry point will execute until the execution reaches the first breakpoint you have set inside of **MainSequence**.

- When the execution reaches the first breakpoint inside of **MainSequence**, select **Debug»Resume**. Verify that the Contents Test sequence from **TS2 CD Test.seq** executes.
- When the execution reaches the second breakpoint inside of **MainSequence**, select **Debug»Resume**. Verify that the Label Test sequence from **TS2 CD Test.seq** executes.
Part B: Move Simulation Settings to the Process Model

In Part A, you created the basic structure of your customized test framework. Now, in Part B, you will start functionally organizing your framework. The first step will be to move some of the non-UUT specific functions to the process model instead of the client sequence file.

1. Create a Simulation Settings Dialog callback in the process model.
   - Select the CD Test Model.seq sequence file.
   - Right-click the PreUUT sequence in the sequences pane and select Insert Sequence.
   - Name the new sequence Simulation Settings Dialog.
   - Right-click the sequence and select Sequence Properties.
   - Enable the Disable Result Recording for All Steps option.
   - Select the Model tab.
   - Select Callback from the Type pull-down menu.
   - Click OK to exit the dialog box.

2. Create a parameter in the Simulation Settings Dialog callback to receive the simulation settings from the CD Test Execution entry point.
   - Verify that Simulation Settings Dialog in the Sequences pane is selected.
   - In the Variables pane, right-click Parameters and select Insert Parameter Type CDSimulationSettings.
   - Name the parameter SimulationSettings.

3. Move the Simulation Settings dialog box from the client sequence file to the process model.
   - Select the TS2 CD Test.seq sequence file.
   - Select Contents Test on the Sequences pane.
   - Right-click the Display Simulation Dialog step and select Cut.
Select the CD Test Model.seq sequence file.

Select the Simulation Settings Dialog sequence on the Sequences pane.

Right-click the Main Step group and select Paste.

Select MainSequence on the Sequences pane.

Drag a Sequence Call step from the Insertion Palette to the Setup step group.

Name the step Simulation Settings Dialog Callback.

Select the Module tab from the Step Settings pane.

Enable the Use Current File checkbox.

Select Simulation Settings Dialog from the Sequence pull-down menu.

Set the value of the SimulationSettings parameter to Parameters.SimulationSettings.

The value of this parameter will be used by the Display Simulation Dialog step in the Simulation Settings Dialog callback.

Select the Properties tab on the Step Settings pane.

Select the Run Options category.

Set the Result Recording Option to Disabled.

Figure 4-9. Simulation Settings Dialog Callback

4. Save the sequence file.
Testing

1. Verify that the Simulation Settings Dialog callback executes as it is called from the process model.

   - Open `<Exercises>\TestStand 2\CD Test\CD Test Model.seq` if it is not already open.
   - Select MainSequence on the Sequences pane.
   - Right-click the Simulation Settings Dialog Callback step and select Breakpoint»Toggle Breakpoint.
   - Open `<Exercises>\TestStand 2\CD Test\TS2 CD Test.seq` if it is not already open.
   - Select Execute»CD Test.
   - When the execution reaches the first breakpoint inside of MainSequence, select Debug»Resume. Verify that the CD Test Simulation Settings dialog box launches, and click OK.

2. Verify that the rest of the sequences execute without errors.

   - When the execution reaches the second breakpoint inside of MainSequence, select Debug»Resume. Verify that the Contents Test sequence from TS2 CD Test.seq executes.
   - When the execution reaches the third breakpoint inside of MainSequence, select Debug»Resume. Verify that the Label Test sequence from TS2 CD Test.seq executes.

3. Remove the breakpoints and save the sequence file.

   - Open `<Exercises>\TestStand 2\CD Test\CD Test Model.seq` if it is not already open.
   - Select MainSequence on the Sequences pane.
   - Right-click the Simulation Settings Dialog Callback step and select Breakpoint»Toggle Breakpoint.
   - Right-click the Contents Test Callback step and select Breakpoint»Toggle Breakpoint.
   - Right-click the Label Test Callback step and select Breakpoint»Toggle Breakpoint.
   - Save the sequence file.
Part C: Move Remaining General Operations to the Process Model

In Part B, you began the process of better organizing and modularizing your test framework. You did this by moving non-UUT specific simulation steps to a new callback within the process model. Now, you will continue this process by creating additional model callbacks for the remaining dialogs and non-UUT specific test steps.

1. Move the Insert CD in Drive Prompt to the process model.
   - Select the CD Test Model.seq sequence file.
   - Right-click the Simulation Settings Dialog sequence in the Sequences pane and select Insert Sequence.
   - Name the new sequence CD in Drive Prompt.
   - Right-click the sequence and select Sequence Properties.
   - Enable the Disable Result Recording for All Steps option.
   - Select the Model tab.
   - Select Callback from the Type pull-down menu and click OK.
   - Select the TS2 CD Test.seq sequence file.
   - Select Contents Test on the Sequences pane.
   - Right-click the Insert CD in Drive Prompt step and select Cut.
   - Select the CD Test Model.seq sequence file.
   - Select the CD in Drive Prompt sequence on the Sequences pane.
   - Right-click the Main step group and select Paste.
   - Select MainSequence on the Sequences pane.
   - Drag a Sequence Call step from the Insertion Palette to just before the Contents Test Callback step.
   - Name the step CD in Drive Prompt Callback.
- Select the **Module** tab from the Step Settings pane.
- Enable the **Use Current File** checkbox.
- Select **CD in Drive Prompt** from the Sequence pull-down menu.
- Select the **Properties** tab on the Step Settings pane.
- Select the **Run Options** category.
- Select **Disabled** in the **Result Recording Option** drop-down menu.

2. Move the Place CD Under Camera Prompt to the process model.
   - Right-click the **CD in Drive Prompt** sequence in the Sequences pane and select **Insert Sequence**.
   - Name the new sequence **CD Under Camera Prompt**.
   - Right-click the sequence and select **Sequence Properties**.
   - Enable the **Disable Result Recording for All Steps** checkbox.
   - Select the **Model** tab.
   - Select **Callback** from the Type pull-down menu and click **OK**.
   - Select the **TS2 CD Test.seq** sequence file.
   - Select **Label Test** from the Sequences pane.
   - Right-click the **Place CD Under Camera Prompt** step and select **Cut**.
   - Select the **CD Test Model.seq** sequence file.
   - Select the **CD Under Camera Prompt** sequence from the Sequences pane.
   - Right-click the **Main** step group and select **Paste**.
   - Select **MainSequence** on the Sequences pane.
Drag a Sequence Call step from the Insertion Palette to just before the Label Test Callback step.

- Name the step CD Under Camera Prompt Callback.
- Select the Module tab from the Step Settings pane.
- Enable the Use Current File checkbox.
- Select CD Under Camera Prompt from the Sequence pull-down menu.
- Select the Properties tab on the Step Settings pane.
- Select the Run Options category.
- Select Disabled in the Result Recording Option drop-down menu.

3. Move the station global variable creation to the process model.

- Select the TS2 CD Test.seq sequence file.
- Right-click SequenceFileLoad in the Sequences pane and select Cut.
- Select the CD Test Model.seq sequence file.
- Right-click ProcessModelPostResults in the Sequences pane and select Paste.
- Select File»Save All.

4. Remove extra entry points from CD Test Model.seq.

- Right-click Test UUTs on the Sequences pane and select Delete.
- Right-click Single Pass on the Sequences pane and select Delete.
**Note** The Test UUTs and Single Pass entry points no longer work with your sequence file because you removed the MainSequence sequence from the client sequence file. To repair the Single Pass sequence rather than deleting it, repeat steps 4, 6, 9 and 10 of **Part A: Replace MainSequence Callback with Contents Test and Label Test Callbacks** for the Single Pass sequence.

5. Save the **CD Test Model.seq** sequence file.

**Testing**

1. Test the station global variable creation.
   - Close both sequence files.
   - Select **View»Station Globals** to open the Station Globals pane.
   - Delete the Test Settings variable.
   - Close and reopen TestStand.
   - Open `<Exercises\TestStand 2\CD Test\TS2 CD Test.seq`.
   - Verify that the Variables Created dialog box launches.
     The process model loads when you load a sequence file that references it. Therefore, the variables are created whenever you load the client sequence file or the process model.
   - Save the sequence file.

2. Test the new execution entry point.
   - Select **Execute»CD Test**.
   - Verify that there are no differences between the original functionality and the new functionality.

**Note** Because the Contents Test or Label Test callbacks fail when steps within them fail, you may see a Failure Chain in your report when the sequence fails. If you want to remove the failure chain, disable the **Step Failure Causes Sequence Failure** options for the callback steps in the **MainSequence** sequence and then add code to manually determine the result of the **MainSequence**. Use the status of the two steps to determine the result of the **MainSequence** sequence. Refer to the code after the **MainSequence** callback in the entry point and in the cleanup section of the entry point for an example.

End of Exercise 4-2
Exercise 4-3  Customizing Data Collection and Report Generation (Optional)

Goal
Add additional results to the ResultList and the report.

Scenario
The need to repeatedly ignore errors may be a sign of problems in your test system. You created an error handling step in Exercise 3-1 to ignore errors, but an ignored error is not visible in the report or database results from a UUT. Modify the process model so that users can add ignored errors to the report by enabling a configuration setting.

Design
In a previous exercise, you included a configuration setting to include ignored errors in the results. In this exercise, you apply the value of that configuration setting to include the ErrorCleared property from the ErrorHandling step in the report. To do this, perform the following separate steps.

• Include the ErrorCleared property in the ResultList local variable so you can add it to a report or log it to a database. Add the ErrorCleared property to the ResultList local variable by calling the AddExtraResult method from within the process model. Create a callback for adding additional results to give the process model or client sequences a location to add other results if necessary.
  You could add the ErrorCleared property directly to the result container, but it would always be part of the results, ignoring the configuration setting. You could call API methods from a Post-Substep or Post-Expression of the ErrorHandler step type to add the property to the results, but this would create a dependency between the ErrorHandler step type and the configuration settings of the CD Test system, and reduce its reusability.

• Set the IncludeInReport flag of the ErrorCleared property to let the report generation routines know that they should include it in the report.
  You could set this flag programmatically using the TestStand API. However, there is no particular benefit to doing so in this case, because you can set a static flag in the custom step type. The flag is ignored unless the property is added to the results, so setting the flag does not cause any problems.

Implementation
1. Create a new sequence in the process model to add extra results.

   • Open `<Exercises>\TestStand 2\CD Test\CD Test Model.seq` if it is not already open.
   • Right-click PreUUT on the Sequences pane and select **Insert Sequence**.
   • Name the new sequence **Add Optional Results**.
   • Right-click **Add Optional Results** in the Sequences pane and select **Sequence Properties**.

Sample
☐ Select the Model tab.

☐ Select Callback from the Type pull-down menu and click OK.

2. Call the AddExtraResult method to add the ErrorCleared property to the ResultList local variable.

☐ Drag a Statement step from the Insertion Palette to the Main step group.

☐ Name the new step Add ErrorCleared.

☐ Select the Expression tab from the Step Settings pane.

☐ Enter the expression RunState.Execution.AddExtraResult("Step.ErrorCleared", "ErrorCleared")

This expression calls the AddExtraResult method for the current Execution object. The AddExtraResult method instructs TestStand to add the ErrorCleared property to the ResultList local variable during sequence execution for any step that has an ErrorCleared property.

Figure 4-10. Add ErrorCleared Expression

☐ Select the Properties tab on the Step Settings pane.

☐ Select the Preconditions category of the Properties tab.
Enter the expression `StationGlobals.TestSettings.LogIgnoredErrors`.
This expression applies the LogIgnoredErrors configuration setting. This expression instructs the step to run only if the setting is `True`.

**Figure 4-11. Add ErrorCleared Precondition**

3. Call the Add Optional Results callback from the execution entry point.
   - Select **CD Test** from the Sequences pane.
   - Insert a Sequence Call step just below the PreUUT Callback step.
   - Name the step **Call Add Optional Results**.
   - Select the **Module** tab from the Step Settings pane.
   - Enable the **Use Current File** checkbox.
   - Select **Add Optional Results** from the Sequence pull-down menu.
4. Set the IncludeInReport flag of the ErrorCleared property.

   - Open <Exercises>\TestStand 2\CD Test\TS2 CD Test.seq if it is not already open.

   - **Tip** It is easier to update the flags on steps you have already used if you load them into memory before making changes.

   - Click the **Types** button to open the Types palette.
   - Select **MyTypes.ini** from the View Types For list.
   - Expand the ErrorHandler step type.
   - Right-click the ErrorCleared property and select **Properties**.
   - Click the **Advanced** button and select **Flags**.
   - Enable the **PropFlags.IncludeInReport** checkbox.
   - Enable the **Apply Changes to Flags and Type Flags to All Loaded Instances of the Type** checkbox.

**Figure 4-12.** Add ErrorCleared Expression
Click OK to close the Edit Flags dialog box.

Click OK to close the ErrorCleared Properties dialog box.

Repeat this step for the ErrorCleared property of the ErrorHandlingStringValueTest step type.

5. Select File » Save All.

Select Increment Type Versions from the Warning dialog box.

Increment the version causes any conflicts to use the newer version of the file.

Select Minor from the Version Number to Increment pull-down menu.

Click OK.

Testing
1. Test the configuration setting.

Select the T52 CD Test.seq sequence file.

Select Configure » CD Test Settings.

Verify that the Include Ignored Errors in Results checkbox is disabled.

Click OK to close the dialog box.

Select Execute » CD Test.

Test one UUT, then stop the test.

Verify that the ErrorCleared property is not included in the results of the Volume Label Test step.
2. Test the additional item in the report.
   - Select **Configure»CD Test Settings**.
   - Enable the **Include Ignored Errors in Results** checkbox.
   - Click **OK** to close the dialog box.
   - Select **Execute»CD Test**.
   - Test one UUT, then stop the test.
   - Verify that the ErrorCleared property is included in the results of the Volume Label Test step. The property should have a value of 0.

End of Exercise 4-3
Exercise 4-4  Creating a Custom Process Model Plug-in (Optional)

Goal
Create and use a custom process model plug-in for result processing.

Scenario
Your sequence developers want operators to quickly see how many steps in the MainSequence are logged to the report for each UUT tested. This allows the test operators to quickly verify that all tests have been executed and logged. They would like to see a message popup that displays this number along with a text message. The developers also request that the following options be configurable:

- The text displayed in the dialog
- Whether to log the displayed message text to a file

Design
Create a simple result processing plug-in to notify the user how many step results are being logged to the report. By using a plug-in, you can implement this functionality without changes to the process model files.

In the plug-in, use the Model Plugins - Configure Standard Options entry point sequence to create a settings dialog which allows the user to configure two properties:

- The message to be displayed (String)
- Whether to log the message text to a file (Boolean)

Add the properties to the <Plug-in>Options type, so that they are accessible by all plug-in sequences. Configure the default values of the Options description expression, the icon name, and the properties you created above using the ModelPluginComponentDescription File Global variable.

The message popup and file logging functionality should be implemented in the Model Plugins - UUT Done entry point sequence. The message popup should display the following information:

- Message text: <Message text> is the value of the message parameter
- Number of Results: <n> is the number of elements in the ResultList array for the MainSequence
Note: You can use the provided VIs to implement the configuration and file logging functionality:

- `ConfigurePluginOptions.vi`—UI for configuration dialog
- `LogToDisk.vi`—logs a string to the specified file

Implementation:
1. Create a new process model plug-in from the result processing dialog.
   - Open `<Exercises>\TestStand 2\CD Test\TS2 CD Test.seq` if it is not already open.
   - Select `Configure»Result Processing` to launch the Result Processing dialog box.
   - Enable the `Show More Options` checkbox.

   ![Figure 4-13. Result Processing Additional Options](image)

   - Click the `Advanced` button to launch the Advanced Result Processing Settings dialog box.

   ![Advanced Result Processing Settings Dialog Box](image)
Click the **Create New Process Model Plug-in** button.

Name the plug-in `CDTest_DisplayNumberOfResults` and click **Save**.

**Note**  Typically, you use your company name as a prefix for new plug-in.

Examine the Process Model Plug-in Created dialog box.

Close all the dialog boxes.

A new window opens for the `CDTest_DisplayNumberOfResults` plug-in sequence file you created. Select the Sequences pane.

Right-click and delete all the sequences except:

- Model Plugin - Configure Standard Options
- Model Plugin - UUT Done
When developing plug-ins, you can delete any plug-in sequences you are not using. If you need to add a plug-in sequence back at a later time, create a new plug-in, and copy the desired sequence from the new plug-in.

2. Add new properties to store configuration information for the plug-in.

- Click the**Types** button and launch the Types pane.
- In the **View Types For** pane, select **CDTest_DisplayNumberOfResults.seq**.
- In the Types pane, expand the **CDTest_DisplayNumberOfResultsOptions** type under Custom Data Types.
- Right-click the <Right click to insert Field> placeholder select **Insert Field»Boolean**.
- Name the new property **LogToDisk**.
- Under **CDTest_DisplayNumberOfResultsOptions**, right-click <Right click to insert Field> placeholder and select **Insert Field»String**.
- Name the new property **MessageText**.
- Save the **CDTest_DisplayNumberOfResults.seq**. If prompted to increment the Type version, select **Major** and click **OK**.

Note The properties in this type are accessible to all plug-in sequences as parameters. These parameters are populated by the process model files when the plug-in sequences are called.

3. Configure and set default values for the plug-in in the ResultProcessing dialog.

- Select **CDTest_DisplayNumberOfResults.seq** to view the plug-in sequence file.
- In the Variables pane, set the value of
  FileGlobals.ModelPluginComponentDescription.Default.Base.OptionsDescriptionExpression to "Display number of results".
- Set the value of FileGlobals.ModelPluginComponentDescription.Default.Base.IconName to "Generic.ico".

Save `CDTest_DisplayNumberOfResults.seq`. If prompted to increment the Type version, select `Minor` and click `OK`.

4. Create a step in the Model Plugins - Configure Standard Options sequence to launch the configuration dialog.

   - In the Sequences pane, select the **Model Plugins - Configure Standard Options** sequence.
   - Delete the steps in the Main step group.
   - In the Steps pane, insert a LabVIEW action step to the Main Step group.
   - Name the step `Launch Configuration Dialog`.
   - Configure the Step Settings pane as shown in Fig. 4-15.

**Figure 4-15. ConfigurePluginOptions.vi Settings**

- **Note** If prompted by the File Not Found dialog box, select the *Add the directory containing the file you selected to the list of search directories* option.

- **Note** The Input data fields allow the dialog to display the current settings when launched. The new settings are returned to the Output data fields.
5. Create a step that stores the desired message text to a local variable.
   - In the Sequences pane, select the **Model Plugin - UUT Done** sequence.
   - In the Variables pane, right-click Locals and select **Insert Local»String**.
   - Name the new local variable **MessageString**.
   - In the Steps pane, add a new **Statement** step to the Main step group.
   - Name the step **Generate Text**.
   - Select the **Expression** tab from the Step Settings pane.
   - Enter the expression:
     ```
     ```
     This expression concatenates the Message text and the number of results in the MainSequence of the client sequence file. The properties referred to by the expression for the number of results string does not exist until run time. Therefore, this expression will appear to have an error when you enter it, and you will not be able to use the expression browser to locate the properties. If you correctly enter the expression, TestStand will locate the values at runtime.

6. Create a step that displays the desired text.
   - In the Sequences pane, select the **Model plugin - UUT Done** sequence.
   - Insert a Message Popup step in the Main step group after the Generate Text step.
   - In the Step Settings pane, select the **Text and Buttons** tab and enter **Locals.MessageString** in the **Message Expression** field.

7. Create a step that conditionally logs text to a file.
   - Ensure the **Model plugin - UUT Done** sequence is selected in the Sequences pane.
   - Drag a new LabVIEW Action step to the Main step Group and place it below the Message Popup step.
   - Name the new step **Log Text to Disk**.
   - In the Step Settings pane, select **Preconditions** on the **Properties** tab.
Enter Parameters.ModelPlugin.PluginSpecific.Options.LogToDisk into the precondition expression field.
This precondition ensures that the step executes only if the plug-in is configured to log to disk.

Configure the Modules tab as shown in Fig. 4-16.

Fig. 4-16. WriteTextToFile.vi Settings

**Testing**

1. Add a new instance of the plug-in you created to the result processing configuration.
   - Open the result processing dialog by selecting Configure»Result Processing.
   - Click the Insert New button, then select the CDTest_DisplayNumberofResults plug-in.

2. In the Result Processing dialog box, ensure the plug-in settings that you configured are working properly.
   - Verify that the information icon is used for the plug-in.
   - Verify that the Options text is Display number of results.

3. Configure the plug-in settings.
   - Click the Options button for the new CDTest_DisplayNumberOfResults plug-in instance.
4. Test the plug-in during execution.
   □ Open `<Exercises>`\TestStand 2\CD Test\TS2 CD Test.seq, if it is not already open.
   □ Select Execute»CD Test.

   <Note> You must run the execution only once.
   □ Verify that a dialog box appears after the sequence executes with the "Test complete" message text and the number of results indicated as 2.
   □ Close the dialog box and stop the test execution.
   □ Verify that the log file containing the message text is created at `C:\test.log`.

5. Disable the Log To File option.
   □ Click Configure»Result Processing.
   □ Click the Options button for the CDTest_DisplayNumberOfResults plugin.
   □ Configure the plug-in settings so that the Log To File option is disabled.
   □ Select Execute»CD Test.

   <Note> You must run the execution only once.
   □ Verify that the log file was not updated after this execution.

End of Exercise 4-4