RELEASE NOTES

NI TestStand™

Version 2010

These release notes contain NI TestStand 2010 system requirements, installation instructions, information about new features, and other changes since TestStand 4.2.

Contents

NATIONAL INSTRUMENTS™

Getting Started

The best way to familiarize yourself with TestStand is to explore the *Guide to TestStand Documentation* topic in the *NI TestStand Help*, which contains links to all the TestStand documentation in electronic format. Select **Start»All Programs»National Instruments»TestStand» Documentation»NI TestStand Help** or select **Help»NI TestStand Help** in the TestStand Sequence Editor to access the *NI TestStand Help*.

Note The *NI TestStand Reference Manual* is available only as a PDF file, located at `<TestStand>\Doc\Manuals\TestStandReferenceManual.pdf`.

Recommended System Requirements

National Instruments recommends the following system requirements to run TestStand 2010. Minimum system requirements follow in parentheses.

- Pentium 4/M or equivalent processor (Pentium III, Celeron 866 MHz, or equivalent minimum)
- 1 GB of memory (256 MB minimum)
- 1 GB minimum of free hard disk space
  - 1 GB of free hard disk space for TestStand
  - Additional 260 MB (x86) or 610 MB (x64) of free hard disk space for Microsoft .NET Framework Version 2.0 if not already installed.
  - Additional free hard disk space for device drivers from the National Instruments Device Drivers DVD. The amount of free hard disk space required varies according to the drivers you choose to install. Install only the drivers for the hardware you will use.
- Super video graphics array (SVGA) resolution or higher video adapter (1024 × 768 minimum video resolution)
- Microsoft-compatible mouse

Use the following Microsoft operating systems with TestStand 2010:

- Windows 7 (32-bit and 64-bit), including Starter Edition
- Windows Vista (32-bit and 64-bit) Business, Enterprise, or Ultimate Service Pack 2
- Windows XP Service Pack 3
- Windows Server 2008 R2 (64-bit) and Windows Server 2003 R2 (32-bit); TestStand 2010 supports only R2 editions of Windows Server

Refer to the *TestStand and Windows 7/Vista* section of this document for more information about using TestStand on Windows 7/Vista.

TestStand does not support Guest user accounts on Windows. TestStand supports Administrator and Limited accounts on Windows 7/Vista and Administrator, Power Users, and Users accounts on Windows XP.

Note This version of TestStand supports, and was tested with, the latest operating system service packs that were available at the time this version of TestStand was released. National Instruments recommends using TestStand with the latest operating system service pack.

Use the following Microsoft software with TestStand 2010:

- Microsoft Internet Explorer version 7.0 or later
- Microsoft Visual Studio 2010, 2008, or 2005
- Microsoft .NET Framework 2.0, 2.1, 3.5, which uses Microsoft .NET Common Language Runtime (CLR) 2.0
- Microsoft .NET Framework 4.0, which uses Microsoft .NET CLR 4.0. Refer to the *Visual Studio 2010 Support* section of this document for more information about calling .NET assemblies written with the .NET Framework 4.0.

TestStand supports integration with Visual Studio Standard but does not support integration with Visual Studio Express. The .NET and Microsoft Foundation Class (MFC) examples use projects and solutions created in Visual Studio 2005. If you are using an earlier version of Visual Studio, create new projects and solutions from the source files TestStand provides.

Use the following National Instruments application development environments (ADEs) with TestStand 2010:

- LabVIEW 2010 f2 or later, 2009 SP1, 8.6.1, or 8.5.1

  You can use LabVIEW 8.0.1 and 8.2.1 VIs with TestStand 2010, but National Instruments does not support these versions of LabVIEW with TestStand 2010. TestStand 2010 does not support calling LabVIEW 7.1.1 or earlier VIs.

  LabVIEW examples and user interfaces use files created with the minimum supported version of LabVIEW 8.5.1. TestStand API support files for LabVIEW use files saved with LabVIEW 8.0.
- LabWindows™/CVI™ 2010, 2009 SP1, 9.0.1, or 8.5.1

  You can use earlier versions of LabWindows/CVI with TestStand 2010, but National Instruments does not support these versions of LabWindows/CVI with TestStand 2010. LabWindows/CVI examples, user interfaces, and API use files created with LabWindows/CVI 8.5.1.

TestStand can execute code modules developed with versions of ADEs other than the listed supported versions, but National Instruments performs only limited testing with earlier versions of ADEs. TestStand might be able to execute code modules developed with versions of ADEs later than the listed supported versions, but National Instruments cannot ensure support for versions released after TestStand. Refer to the National Instruments Web site at `ni.com/info` and enter the Info Code `tslcp` for more information about the TestStand life cycle policy and supported versions of LabVIEW and LabWindows/CVI.

Installation Instructions

Unless you specify another location during installation, the installation program copies core files to `<Program Files>\National Instruments\TestStand`.

Note You cannot install TestStand to a network path or a mapped network path. You must install TestStand on a local computer.

National Instruments recommends exiting all programs before running the installer. Applications that run in the background, such as virus scanning utilities, might cause the installer to take longer than average to complete.

Complete the following steps to install TestStand.

1. Log on as an administrator or as a user with administrator privileges.
2. Insert the TestStand 2010 installation media and follow the instructions that appear on the screen.

National Instruments recommends installing the complete TestStand program. If you perform a custom installation and do not install all the TestStand features, you can run the installation program again later to install additional features.

If you install LabVIEW, LabWindows/CVI, or Visual Studio after you install TestStand, launch the sequence editor or a user interface once so TestStand configures these applications to locate TestStand API files. When you install a newer version of LabVIEW after you install TestStand, run the TestStand Version Selector and select the most recent version of TestStand to ensure the LabVIEW User Interface VIs function properly. Select **Start»All Programs»National Instruments»TestStand» TestStand Version Selector** to launch the TestStand Version Selector application, `TSVerSelect.exe`, located in `C:\Program Files\National Instruments\Shared\TestStand Version Selector`.

Refer to the National Instruments Web site at `ni.com/info` and enter the Info Code `rddrau` to access the latest software drivers and updates.

Activating TestStand Licenses

After you install TestStand, you must use the NI Activation Wizard to activate the software or initiate the evaluation period for the software. When you activate TestStand, you need the serial number and the name of the software kit. You can find both of these items on the Certificate of Ownership card included in your software kit. Refer to the *Activating Your Software* topic in the *NI TestStand Help* for more information about how to activate TestStand.

National Instruments offers a variety of licenses for the different ways you can use TestStand in development and deployment applications. You can select from the following types of licenses: the TestStand Development System License, the TestStand Custom Sequence Editor License, the TestStand Debug Deployment Environment License, and the TestStand Base Deployment Engine License.

In most cases, when you first install TestStand, you activate a TestStand Development System License. Use the other licenses to activate TestStand on computers to which you deploy TestStand custom sequence editors or user interfaces you develop. Refer to the *Licensing Options for TestStand Systems* section of Chapter 1, *Introduction to TestStand*, of the *Using TestStand* manual for information about how to use the available licenses when you build a TestStand-based test solution.

Use the following descriptions only as a reference for the licensing options. Refer to `ni.com/activate` for more information about activating TestStand licenses. Refer to `ni.com/teststand` to purchase a TestStand license. Contact a local National Instruments representative for more information or for questions about specific licensing needs.

Note This document does not replace the National Instruments Software License Agreement installed in the `<National Instruments>\Shared\MDF\EULAs\NIReleased` directory.

TestStand Evaluation Package

When you run TestStand in Evaluation Mode, the software expires after 30 days. The software runs as a fully functional Development System for the first 7 days of the evaluation period. After 7 days, TestStand runs with the following restrictions:

- Sequence execution time limit of 10 minutes
- Continuous application usage time limit of 1 hour

You can activate a license at any point during or after the 30-day evaluation period.

TestStand Development System License (777777-09)

Activate the TestStand Development System License to develop and edit test sequences and to save sequence files within the TestStand Sequence Editor, within a TestStand custom sequence editor, or programmatically using the TestStand API. This license also grants the ability to develop custom sequence editors and operator interfaces.

TestStand Custom Sequence Editor License (777775-01)

Activate this license to develop and edit test sequences and sequence files within a TestStand custom sequence editor or programmatically using the TestStand API.

TestStand Debug Deployment Environment License (779851-09)

The TestStand Debug Deployment Environment License offers the most flexibility for deploying TestStand and LabVIEW-based, LabWindows/CVI-based, and Measurement Studio-based systems.

Activate this license to install the development versions of TestStand, LabVIEW, LabWindows/CVI, Measurement Studio, and any corresponding add-on toolkits on a single test station so you can debug deployed test applications on the test station. This license grants the ability to make minor edits to fix bugs in deployed test applications but does not grant the ability to perform any development tasks using TestStand, LabVIEW, LabWindows/CVI, or Measurement Studio on the test station.

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.

TestStand Base Deployment Engine License (777774-03)

The TestStand Base Deployment Engine License is the minimum license required for all deployed TestStand-based applications. Activate this license to deploy the TestStand Engine, a TestStand Operator Interface, and sequence files to the single test station to which the license applies. This license does not grant the ability to perform any development tasks using the TestStand Sequence Editor, a TestStand custom sequence editor, or the TestStand API.

Recommended Database Client Software

Use the following recommended database client software with the database components included with TestStand:

- **Microsoft Access**—Use the Microsoft Jet 4.0 Object Linking and Embedding Database (OLE DB) Provider.
- **Microsoft SQL Server**—Use the Microsoft OLE DB Provider for SQL Server or the SQL Native Client provider.
- **Oracle**—Use the latest Oracle Provider for OLE DB and Oracle Client software. You can download the Oracle Provider from the Oracle Web site at `www.oracle.com`.

Note National Instruments does not recommend using the Microsoft OLE DB Provider for Oracle because it does not support all the OLE DB features TestStand requires.

Note Oracle recommends using Oracle Provider 11.1.0.6.0 or later if you want to read 64-bit integer values stored in NUMBER columns.

- **MySQL**—Use the MySQL Open Database Connectivity (ODBC) Driver 3.51 or later.
- **Sybase SQL Anywhere**—Use the Adaptive Server Anywhere ODBC Driver 9.0 or later.

Installing Additional Software Components

TestStand includes the device driver software on a DVD. If you require device driver software on CDs, refer to the National Instruments Web site at `ni.com/info` and enter the Info Code `drivercd`.

TestStand 2010 installs the following additional software components:

- .NET Framework 2.0 Service Pack 1
- LabVIEW 2009 SP1 f1, 8.6.1, and 8.5.1 Run-Time Engine (RTE)
- LabWindows/CVI 2009 SP1 RTE
- LabWindows/CVI SQL Toolkit DLL version 2.2, `cvidb32.dll`
- National Instruments Session Manager
- NI Variable Engine
- NI Update Service

TestStand no longer requires or installs the Remote Execution Support for NI TestStand feature for any version of LabVIEW.

Note TestStand can use newer versions of the LabVIEW RTE when you install LabVIEW on a development system. You can include newer versions of the LabVIEW RTE in

deployments using the Drivers and Components dialog box of the TestStand Deployment Utility. Refer to the *NI TestStand Help* for more information about the Drivers and Components dialog box.

The National Instruments Device Driver DVD contains the following suggested components:

- National Instruments Measurement & Automation Explorer (MAX)
- Interchangeable Virtual Instrument (IVI) Compliance Package (ICP)

To use IVI, download and install IVI Compliant-specific drivers from the Instrument Driver Network, located at `ni.com/devzone/idnet`.

Installing Multiple Versions of TestStand on the Same Computer

You can install TestStand 2010 on a computer that contains a previous TestStand version, but you cannot install TestStand 2010 over a previous TestStand version. You cannot install TestStand 2010 in the default installation directory for a previous version of TestStand. For example TestStand 2010 does not install in `Program Files\National Instruments\TestStand 4.2.1`. If you installed a previous version of TestStand in a non-default directory, you can uninstall the previous version of TestStand and install TestStand 2010 to that directory.

Uninstalling TestStand 2.0.1 or Earlier

Because the uninstallers for TestStand 2.0.1 or earlier remove the `<TestStand>\OperatorInterfaces\User` directory, you must complete the following steps to safely uninstall the previous TestStand version and preserve all configuration files and files located in the `User` subdirectories.

1. Move the `<TestStand>\OperatorInterfaces\User` directory to a location outside the `<TestStand>` directory.
2. Navigate to the standard Windows Control Panel facility for adding and removing programs and run the TestStand uninstaller. If the uninstaller launches a dialog box that requests confirmation to remove all TestStand configuration files and pre-installed user components, click **No**.
3. When the uninstaller completes, move the `OperatorInterfaces\User` directory back into the original `<TestStand>` directory.

   You can now install TestStand 2010 into this directory.

TestStand and Windows 7/Vista

TestStand 2010 runs on Windows 7/Vista.

Windows Vista and Later Behavior Change

Windows Vista introduced a change in behavior that can cause applications to receive WM_PAINT messages unexpectedly while performing ActiveX/COM calls from .NET components or .NET applications. National Instruments has tested TestStand 2010 and fixed instances where this behavior change caused the sequence editor and TestStand User Interface (UI) Controls to behave incorrectly. Refer to the National Instruments KnowledgeBase at `ni.com/info` and enter the Info Code `rdwttc` for more information about this Windows Vista behavior change and how it might affect applications that you write or modify.

User Account Control Elevation Prompts

The User Account Control (UAC) security component in Windows 7/Vista requires administrator privileges for some tasks, such as installing software, running certain applications, and changing system settings. If you are logged in as a user with standard privileges, Windows 7/Vista launches a UAC elevation prompt for prohibited tasks. You cannot resolve the UAC elevation prompts programmatically. Refer to Microsoft documentation for more information about UAC prompts.

TestStand 2010 uses a Windows service to automatically handle most TestStand-related UAC prompts and notifications. The National Instruments TestStand Service runs with administrator privileges in the background and is responsible for tasks that require administrator privileges. The TestStand Service does not automatically handle applying settings for remote execution. Performing these actions while logged in to Windows 7/Vista as a user with standard privileges results in a UAC elevation prompt.

Visual Studio Integration

When you launch Visual Studio from TestStand, Visual Studio runs with the same privileges you used to run TestStand. If you launch TestStand while logged in as a user with standard privileges and you launch Visual Studio from TestStand, you cannot execute tasks in Visual Studio that require administrator privileges.

Using the TestStand Version Selector

Although you can install TestStand 2010 on a computer that contains a previous TestStand version, only one version of TestStand can be active at a time. If you must install TestStand 2010 on the same computer as a previous version, use the TestStand Version Selector to specify the active version. Select **Start»All Programs»National Instruments»TestStand»TestStand Version Selector** to launch the TestStand Version Selector application, `TSVerSelect.exe`, located in the `C:\Program Files\National Instruments\Shared\TestStand Version Selector` directory.

If you activate TestStand 2010 and run a TestStand User Interface from the previous TestStand version, the user interface uses the TestStand 2010 engine, step types, and components. If you activate the previous TestStand version and run a TestStand 2010 User Interface or the sequence editor, those applications do not function correctly.

Directory Relocation

To comply with Windows 7/Vista restrictions on writing to the `Program Files` directory and to improve usability for Windows XP users who do not have permission to write to the `Program Files` directory, TestStand 4.1 or later installs some files in different locations from previous versions of TestStand. Refer to the *TestStand and Windows 7/Vista* section for more information about using TestStand on Windows 7/Vista.

Table 1 lists the directory locations in TestStand 4.0 and the current locations. The TestStand documentation and Table 1 refer to these directories in the following ways:

- **<TestStand>**—Located by default at `C:\Program Files\National Instruments\TestStand` on Windows 32-bit systems and at `C:\Program Files (x86)\National Instruments\TestStand` on Windows 64-bit systems.
- **<TestStand Public>**—Located by default at `C:\Users\Public\Documents\National Instruments\TestStand` on Windows 7/Vista and at `C:\Documents and Settings\All Users\Documents\National Instruments\TestStand` on Windows XP.
- **<TestStand Application Data>**—Hidden by default and located at `C:\ProgramData\National Instruments\TestStand` on Windows 7/Vista and at `C:\Documents and Settings\All Users\Application Data\National Instruments\TestStand` on Windows XP.

- **<TestStand Local Application Data>**—Hidden by default and located at `<User Directory>\AppData\Local\National Instruments\TestStand` on Windows 7/Vista and at `<User Directory>\Local Settings\Application Data\National Instruments\TestStand` on Windows XP.

Table 1. Directories Relocated in TestStand 4.1

Location in TestStand 4.0	Current Location
`<TestStand>\AdapterSupport\CVI`	`<TestStand Public>\AdapterSupport\CVI`
`<TestStand>\AdapterSupport\LabVIEW`	`<TestStand Public>\AdapterSupport\LabVIEW`
`<TestStand>\Cfg`	`<TestStand Application Data>\Cfg`
`<TestStand>\CodeTemplates\NI`	`<TestStand>\CodeTemplates`
`<TestStand>\CodeTemplates\User`	`<TestStand Public>\CodeTemplates`
`<TestStand>\Components\NI`	`<TestStand>\Components`
`<TestStand>\Components\NI\RuntimeServers`	`<TestStand Public>\Components\RuntimeServers`
`<TestStand>\Components\User`	`<TestStand Public>\Components`
`<TestStand>\Examples`	`<TestStand Public>\Examples`
`<TestStand>\Setup`	`<TestStand Public>\Setup`
`<TestStand>\Tutorial`	`<TestStand Public>\Tutorial`
`<TestStand>\UserInterfaces\NI`	`<TestStand>\UserInterfaces`
`<TestStand>\UserInterfaces\User`	`<TestStand Public>\UserInterfaces`

Note The `AdapterSupport\CVI` and `AdapterSupport\LabVIEW` directories were moved to the `<TestStand Public>` directory because TestStand requires write access to these directories for standard users.

Use the `<TestStand Public>\CodeTemplates`, `<TestStand Public>\Components`, and `<TestStand Public>\UserInterfaces` directories in place of the corresponding `User` directories in TestStand 4.0 or earlier. Use the `<TestStand>\CodeTemplates`, `<TestStand>\Components`, and `<TestStand>\UserInterfaces` directories in place of the corresponding `NI` directories in TestStand 4.0 or earlier. You can use the `Engine.GetTestStandPath` method to find the directories programmatically.

To modify the installed code templates or components or to create new code templates or components, copy the files from the <TestStand> directory to the <TestStand Public> directory and make changes to the copies. To modify the installed user interfaces or to create new user interfaces, modify the files TestStand installs in the <TestStand Public>\UserInterfaces directory. When you modify installed files, 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 new installations of the same version of TestStand do not overwrite the customizations and ensures that uninstalling TestStand does not remove the files you customize.

Migrating to TestStand 2010

If you are migrating from a version of TestStand earlier than TestStand 4.1, complete the following tasks:

- If you saved files in a relocated directory, copy the files to the new location in TestStand 4.1 or later.
- Update projects and files that reference files and directories that have moved in TestStand 4.1 or later. You can use the Engine.GetTestStandPath method to retrieve file and directory paths programmatically.

  If you use relative paths to reference files in the <TestStand> directory, the paths might break.

  If LabWindows/CVI returns a warning that some files were not found when you open a project, remove the files from the project and re-add them from the <National Instruments>\Shared\CVI\instr\TestStand\API directory in LabWindows/CVI 8.5 or later and from the <CVI>\instr\TestStand\API directory in LabWindows/CVI 8.1.1 or earlier.

  If Visual Studio returns a warning that some TestStand API or adapter support files were not found when you open a project, add the $(TestStand), $(TestStandAppData), and $(TestStandPublic) environment variables as needed to the Additional Include Directories control for the project.
- Use the TestStand Deployment Utility to update deployable images or installers for deployable images to redirect file destinations to the TestStand 4.1 or later locations. Select **View Destination** in the Distributed Files control and select the new location in the Installation Destination control in the Installer Properties section on the Distributed Files tab.

Migrating User Components

You can copy the following directories and files from the previous TestStand installation to the appropriate TestStand 2010 directory to migrate configuration settings and user components. If you are migrating from a version earlier than TestStand 4.1, refer to the *Directory Relocation* section for more information about directory location changes.

- `<TestStand Application Data>\Cfg\StationGlobals.ini`
- `<TestStand Application Data>\Cfg\TestExec.ini`
- `<TestStand Application Data>\Cfg\TestStandModelReportOptions.ini`
- `<TestStand Application Data>\Cfg\Users.ini`
- `<TestStand Public>\CodeTemplates`
- `<TestStand Public>\Components`
- `<TestStand Public>\UserInterfaces`

If you use custom Tools menu items in the previous TestStand version, complete the following steps to export the items from that installation and import them into TestStand 2010.

1. In the previous version, select **Tools»Customize** in the sequence editor to launch the Customize Tools Menu dialog box.
2. Click the **Export Items to File** button to launch the Export Tools Menu dialog box.
3. Select the menu items to export to a Tools menu file and click **OK**.
4. Create a `<TestStand Public>\Setup\ToolMenusToInstall` directory.
5. Place the Tools menu file you created in step 3 in the new directory.
6. Launch the TestStand 2010 Sequence Editor. TestStand adds the new menu items to the Tools menu and deletes the Tools menu file.

Migrating Changes to TestStand Components

TestStand includes several components you can customize, such as process models, user interfaces, and certain step types. If you made changes to one of these components, place the custom component in the appropriate `<TestStand Public>` directory after you install TestStand 2010.

If you made substantial or complex changes to the component, use a file-comparison tool to determine the changes between the TestStand 2010 version of the component and the original version of the component you modified and to apply the TestStand 2010 improvements to the custom version of the component.

If you made minor changes to the component, use a file-comparison tool to determine the changes you made to the component and to reapply the improvements to a copy of the TestStand 2010 version of the component.

You can use the following types of file-comparison tools:

- To compare sequence files, select **Start»All Programs»National Instruments»TestStand»Tools»File Diff/Merge Tool** to launch the Select Files dialog box of the TestStand File Diff/Merge application.
- To compare text files, use a source code comparison tool, such as Microsoft WinDiff. You can also use the Diff command in the Edit menu of the LabWindows/CVI Source window.
- To compare VI files, select **Tools»Compare** in LabVIEW to use the Compare VIs tool. This tool is available only in the LabVIEW Professional Development System.

Note Subsets of different versions of the same component are not necessarily interoperable without modifications. For example, you cannot replace a single sequence in the TestStand 2010 process models with the corresponding sequence from older process models without making further modifications. If you customized the process models, you must ensure that TestStand can find all the subordinate components the process models use and that any of those components that are ActiveX servers are registered. The default process model sequence for TestStand uses separate sequences, DLLs, and ActiveX servers to support database logging and report generation features.

Behavior Changes

TestStand includes the following behavior changes between version 4.2.1 and version 2010. Refer to the *NI TestStand Help* for more information about behavior changes between earlier versions of TestStand. Refer to the *TestStand and Windows 7/Vista* section for more information about behavior changes in Windows 7/Vista that might affect applications that you write or modify.

- TestStand 2010 does not support Windows 2000 or LabVIEW 7.1.1. TestStand 4.2.1 is the last version that supports Windows 2000 and LabVIEW 7.1.1.
- TestStand 2010 is the last version to support writing out files using the TestStand 2.0 and 2.0.1 file formats.
- The sequence editor File menu includes the following changes:
  - The File menu no longer includes the top-level **New Sequence File**, **Open Sequence File**, **Close**, **New Workspace File**, **Open Workspace File**, **Close Workspace File**, **Add File to Workspace**, **Save**, and **Save As** menu items.

  - Select **File»New** to create new sequence files, workspace files, and TestStand Sequence Analyzer projects.
  - Select **File»Open File** to open existing TestStand files.
  - Select **File»Close <*filename*>** to close the active window on the Workspace pane.
  - Select **File»Add <*filename*> to Workspace** to add the file associated with the active window to the workspace.
  - Select **File»Save <*filename*>** to write the contents of the active window or pane to disk.
  - Select **File»Save <*filename*> As** to write the contents of the active Sequence File window or current workspace on the Workspace pane to disk using a new name you enter.
- The **Open Sequence File** button on the sequence editor toolbar changed to the **Open File** button.
- The .NET Adapter includes the following changes:
  - In previous versions of TestStand, you could call code modules with the .NET Adapter even when the prototype in the assembly did not match the prototype you specified for the step. In TestStand 2010, the .NET Adapter and the TestStand Sequence Analyzer return an error when the prototype does not match exactly. However, for backward compatibility, the .NET Adapter uses a similar algorithm as in previous versions of TestStand to try to run the code modules despite the error. National Instruments does not recommend relying on this behavior. National Instruments recommends that you use the sequence analyzer and update the prototypes of .NET steps as needed when the prototypes in code modules change.
  - The .NET Adapter no longer supports storing or retrieving Int64 or UInt64 parameters as a floating-point number because double-precision, floating-point numbers cannot store all possible 64-bit integer values. Use the new 64-bit integer data type representations of TestStand Number variables instead.
  - In previous versions of TestStand, the .NET Adapter did not support the UIntPtr data type as a native type. The adapter now supports this data type, which might lead to incompatibilities if you were using this type in a previous version of TestStand.
  - Previous versions of the .NET Adapter did not honor the Dispose option on the .NET Module tab of the Step Settings pane or on the Module tab of the Edit .NET Call dialog box for unspecified output-only parameters and return values. Now, when you enable the Dispose option for such a parameter, TestStand correctly calls Dispose on the object after the call.

  - Previous versions of the .NET Adapter changed single-dimensional array parameters with non-zero lower bounds to have a zero lower bound for a subset of numeric data types. Other data types generated a run-time error if the lower bounds of the single-dimensional array was not zero because .NET does not support non-zero lower bounds for single-dimensional arrays. To preserve backward compatibility, the TestStand 2010 .NET Adapter converts all single-dimensional arrays to zero-based indexes as well but applies the change to all data types to be more consistent.
  - In TestStand 2010, the data structure in which TestStand stores the settings for .NET steps changed considerably. Specify a step and use the `StationOptions.ShowHiddenProperties` property to view the new data structure.
  - The `DotNetParameter.Direction` property now returns flags that correctly match parameter direction settings on the .NET Module tab on the Step Settings pane and the Module tab of the Edit .NET Call dialog box.
  - The API for the .NET Adapter changed considerably in TestStand 2010. Refer to the *API* section of the *.NET Adapter Enhancements* section of this document for more information about these changes.
  - Out parameters, in/out parameters, and return value parameters now return an error on the .NET Module tab on the Step Settings pane and on the Module tab of the Edit .NET Call dialog box if you pass a container, such as a `PropertyObject` or `SequenceContext`, as the argument even if the data type of the parameter is an interface that the container supports. The step executes without error to maintain run-time backward compatibility, and TestStand assigns the output value back to the object as in previous versions of TestStand. However, National Instruments recommends you use object reference variables to store `PropertyObjects` that you pass as output parameters or pass such objects as purely input parameters in the cases in which you are not trying to return a new object as an output.
- The LabVIEW VI Options dialog box includes the following changes:
  - The Lock VI Diagrams option is now the Apply New Password option.
  - The Check for Broken VIs During Analysis option and the Check for Broken VIs After Build option merged into the new Check for Broken VIs option.
  - New options include the Remove Block Diagrams option, the Consolidate Files Shared by Projects option, and the Output VIs to Packed Project Library option. Click the **Options** button next to

the Output VIs to Packed Project Library option to launch the Packed Project Library Options dialog box, which you use to configure options for how the deployment utility builds packed project libraries.

- The System Source tab of the TestStand Deployment Utility now includes a Deploy Files section, which includes the following options:
  - From TestStand Workspace File
  - Load Workspace in Sequence Editor
  - From Directory and Include Subdirectories
  - From TestStand Public Directories

  The System Source tab no longer includes the Workspace File Path control.
- The Distributed Files tab of the TestStand Deployment Utility now includes an Include Without Processing Item or Dependencies option and an Include All Files in LabVIEW Project option.
- Installing TestStand 2010 on a system prevents previous versions of the TestStand Deployment Utility from creating an installer for Windows 2000. Refer to the National Instruments Web site at `ni.com/info` and enter the Info Code `tsdinstallw2k` for more information about this issue.
- The Auto Deploy Shared Variables for All LabVIEW Projects option in the LabVIEW Adapter Configuration dialog box changed to Auto Deploy Shared Variables on First Load of any VI in a Project. The Auto Undeploy Shared Variables For All LabVIEW Projects option changed to Auto Undeploy Shared Variables on Last Unload of all VIs in a Project.

Note Enabling this option does not deploy shared variables defined in LabVIEW packed project libraries specified in LabVIEW projects you use in TestStand. Use the Deploy Library step type to deploy or undeploy to the local computer the shared variables defined in a LabVIEW packed project library file.

- By default, when TestStand 2010 calls VIs using the LabVIEW 2009 or later RTE, VIs for which the preferred execution system is **same as caller** execute using multiple threads. You can use the new options in the LabVIEW Adapter Configuration dialog box to modify this default behavior. National Instruments recommends that you review the *General Strategies for Optimizing LabVIEW Code Module Performance on SMP Systems* topic in the *NI TestStand Help* and that you run performance benchmarks to decide whether to fine-tune test system performance using the new options in the LabVIEW Adapter Configuration dialog box.

- In the sequence editor and user interfaces in Editor Mode, you can no longer add both attributes and a subproperty named `Attributes` to a container.
- The value of the `LVParamType_OtherRef` constant is now 0x43 instead of 0x41.
- When you select Binary String in the Type column of the VI Parameter Table on the LabVIEW Module tab to store a LabVIEW string that contains binary data in a TestStand property, TestStand no longer escapes the string before storing it and no longer substitutes hexadecimal codes for the unprintable characters, such as the `NUL` character, in the string. Instead, TestStand compresses the binary data and then encodes the compressed data before storing it using only printable ASCII characters from ASCII 48 (0x30) to ASCII 111 (0x6F). To pass a compressed string to a VI, select **Binary String** in the Type column. TestStand unencodes and decompresses the binary data that the string stores before passing it to the VI.
- In certain situations, the TestStand `Seconds(True)` function returns incorrect elapsed time values when the operating system timer is running slower than actual time. Additionally, Wait steps take longer to execute than the time you specify in the step when the operating system timer is running slower than actual time. Refer to the National Instruments Web site at `ni.com/info` and enter the Info Code `tstime` to access the National Instruments KnowledgeBase article 5703Q01O, *Using the TestStand Seconds Function or Wait Step may Result in Incorrect Elapsed Time*, for more information about this issue.
- The LabVIEW 8.5 version of the Report Generation Toolkit VIs changed to object-oriented VIs. With this change, the Report Generation Toolkit references changed from LabVIEW references, interpreted as numerics in TestStand, to LabVIEW Class references, interpreted as object references in TestStand 2010. This change in behavior introduces an incompatibility when upgrading systems that use previous versions of TestStand and LabVIEW 8.2.1 or earlier and the Report Generation Toolkit earlier than version 8.5.

  If you upgrade such a system to TestStand 2010 and LabVIEW 8.5 or later, attempting to execute a sequence that contains LabVIEW code modules that pass Report Generation Toolkit references to and from TestStand results in a module loading error to indicate that the prototypes of the VIs changed and need updating. If you open the sequence and select one of these code module steps in the TestStand Sequence Editor, the LabVIEW Module tab returns an error prompting you to reload the prototype.

Complete the following steps to upgrade systems that use previous versions of TestStand and LabVIEW 8.2.1 or earlier and the Report Generation Toolkit earlier than version 8.5 to use TestStand 2010 and LabVIEW 8.5 or later.

1. Mass-compile all LabVIEW code modules in LabVIEW 8.5 or later.
2. Open the sequence files you need to update in TestStand 2010 and change the type of any variables used to store Report Generation Toolkit references from number to object reference.
3. Reload the prototype of all LabVIEW steps configured to call code modules that pass Report Generation Toolkit references to and from TestStand. You can reload the prototype of each step manually or you can select **Tools»Update VI Calls** to launch the Update VI Calls dialog box, in which you can update the Express VIs a LabVIEW step instance calls and check or update a Standard VI call prototype.
4. When you use the Update VI Calls tool, ensure that you enable the **Force Prototype Reload** option for Express VIs and Standard VIs in the Type of Calls to Update section.
5. Because the parameter type changed from number to object reference when you reload the prototype, TestStand cannot automatically remap the parameter argument. Manually re-specify the Report Generation Toolkit reference parameters for each LabVIEW step configured to call a code module that passes Report Generation Toolkit references to and from TestStand.

- The `Engine.DisplayHelpFile` and `Engine.DisplayHelpTopic` methods create a help window with different default settings than in previous versions of TestStand. In previous versions of TestStand, the help window included the **Hide**, **Back**, **Forward**, **Stop**, **Refresh**, **Home**, **TestStand**, and **Options** toolbar buttons. In TestStand 2010, the help window includes only the **Hide**, **Locate**, **Back**, **Forward**, and **Options** toolbar buttons. Additionally, the Search tab includes the Advanced Search controls.
- TestStand 2010 does not include the TestStand 3.5 Visual Studio 2003 version of the .NET user interfaces. TestStand 4.2.1 is the last version that includes those files. Refer to the National Instruments Web site at `ni.com/info` and enter the Info Code `ex3bcm` to obtain the source code for these user interfaces.
- Some keyboard shortcuts have changed. Refer to the *Keyboard Navigation* topic in the *NI TestStand Help* for more information about current keyboard shortcuts.

What's New in TestStand 2010

This section describes the new features in TestStand 2010 and other changes since TestStand 4.2. This section only summarizes each new feature. Refer to the TestStand documentation listed for a particular feature for more information about that feature.

TestStand Sequence Analyzer

Use the TestStand Sequence Analyzer in the TestStand Sequence Editor or the stand-alone sequence analyzer application to find errors, enforce custom development guidelines you establish, and gather statistics about workspace files, sequence files, directories, type palette files, station globals files, template files, and users files during development or before deployment. The sequence analyzer uses a built-in set of rules and analysis modules to analyze the files you specify and generate messages that correspond to each issue found during analysis. The built-in rules are designed to detect, at edit time, the most common situations that can cause run-time failures.

In the sequence editor, click the **Current Sequence Analyzer Project** button, shown at left, on the Sequence Analyzer toolbar or select **Debug»Sequence Analyzer»Current Sequence Analyzer Project** to open the Current Sequence Analyzer Project window. Select **Start» All Programs»National Instruments»TestStand»Tools»Sequence Analyzer** to launch the stand-alone TestStand Sequence Analyzer application.

Use the Current Sequence Analyzer Project window in the sequence editor or the sequence analyzer application window to edit and save an analyzer project file (`.tsaproj`), which specifies the files to analyze and the rules and related settings to use during the analysis. The sequence editor creates a default project (`MyAnalyzerProject.tsaproj`) the first time you launch the sequence analyzer and then launches the most recently opened analyzer project file each time you open the Current Sequence Analyzer Project window. Closing the Current Sequence Analyzer Project window does not unload the current analyzer project file from memory. You can create, modify, and save project files to customize the set of rules, rule configuration settings, and set of files to use during analysis.

Each built-in TestStand rule defines a single task to perform during analysis, such as ensuring that the prototype of each code module is up to date or ensuring that each expression specified evaluates to the correct type and does not include any syntax errors.

You can set the severity level for individual rules, and you can disable individual rules in the current project. You can modify the settings of some built-in rules.

Use the Analysis Results pane in the sequence editor or the Messages tab in the sequence analyzer application to view and resolve the messages for the most recent analysis of the current analyzer project, sequence file, or workspace file. You can process messages while the sequence analyzer is running. The sequence analyzer overwrites the contents of the Analysis Results pane and the Messages tab each time you start an analysis session.

For each message in the Active Messages view of the Analysis Results pane or on the Messages tab, you can perform the following actions:

- Go to the location of the object in the TestStand file that caused the message and resolve the issue.
- Ignore the issue for subsequent analysis and remove it from the Active Messages view. Select this option only when you are certain the issue will not cause a run-time error.
- Mark the message as fixed and remove it from the Active Messages view.
- Disable or configure the corresponding rule on the Rules pane of the sequence analyzer.

When you mark a message as ignored after analyzing a sequence file or workspace file in the sequence editor, the sequence analyzer adds a copy of the ignored message to the list of ignored messages in the current analyzer project. The sequence analyzer ignores the message in subsequent analysis sessions for the current analyzer project, sequence file, or workspace file. When you mark an issue as fixed, the sequence analyzer reports the message again in subsequent analysis sessions if you do not fix the issue itself.

You can save the messages in an XML report file to view in an external application. The XML file uses the `AnalyzerReportViewer.xsl` style sheet located in the `<TestStand>\Components\Stylesheets\Analyzer` directory. When you generate a report file, the sequence analyzer provides a snapshot of the most recent analysis, including a list of the analyzed files, a list of the rules and settings used during the analysis, and a list of all the generated messages.

By default, the sequence analyzer enables the Analyze File Before Executing option in the Sequence Analyzer Options dialog box. In the sequence editor, click the **Toggle Analyze File Before Executing** button, shown at left, on the Sequence Analyzer toolbar or select **Debug»Sequence Analyzer»Toggle Analyze File Before Executing** to enable or disable this option. When you enable this option, the sequence analyzer uses the rules

and settings in the current sequence analyzer project to analyze the active sequence file when you initiate an execution. Enabling this option can help you find errors early during development.

You can create custom rules and analysis modules for the sequence analyzer to use. For example, you might want to create a custom rule and analysis module to validate that all sequence files use a specific process model file. Refer to the files located in the `<TestStand Public>\Examples\AnalyzerCustomRules` directory for examples of custom rules and analysis modules. You can also distribute custom rules and related supporting files to use in analysis projects or to analyze files on other computers on which TestStand is installed.

The set of rules available on a computer is the same for all analyzer projects you open on that computer. When you add a new rule to the computer and then open an older project, the sequence analyzer automatically adds the new rule to the project. Similarly, the sequence analyzer removes from projects any rules that no longer exist on the computer.

Refer to the *NI TestStand Help* for more information about the sequence analyzer.

LabVIEW Support

TestStand 2010 includes support for LabVIEW projects (LabVIEW 2009 SP1 f1 or later), LabVIEW packed project libraries (LabVIEW 2010 or later), and LabVIEW classes (LabVIEW 2009 SP1 or later). TestStand 2010 also supports automatically using the specific version of the RTE that corresponds to a VI and supports symmetric multiprocessing in VIs executed from TestStand.

Refer to the National Instruments Web site at `ni.com/info` and enter the Info Code `lv2009SP1f1` for more information about LabVIEW 2009 SP1 f1.

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.

Project Integration

In TestStand 2010, you can run VIs under LabVIEW projects through the LabVIEW Development System and the LabVIEW RTE.

Note You must have LabVIEW 2009 SP1 f1 or later to use LabVIEW projects in TestStand.

The LabVIEW Module tab of the Step Settings pane in the sequence editor and the Edit LabVIEW VI Call dialog box in a TestStand User Interface now contain an optional Project Path control, in which you can specify a path to the project that contains the VI to call. When you specify a project path, use the VI Path control to specify the path within the project to locate the VI. Click the **Browse for VI** button to select a VI in the project.

When TestStand calls VIs under a project, the VIs run using project-specific settings, such as conditional compilation and NI-DAQmx tasks, channels, and scales. TestStand can automatically deploy and undeploy LabVIEW libraries that contain shared variables the LabVIEW project defines. Use the LabVIEW Project Settings section of the LabVIEW Adapter Configuration dialog box to specify whether TestStand automatically deploys and undeploys shared variables for the LabVIEW projects that the adapter loads to call VIs.

You can also use the Deploy Library step type to deploy or undeploy to the local computer the shared variables defined in a LabVIEW project library file or packed project library file, or to deploy or undeploy shared variables defined in LabVIEW project libraries within a selected LabVIEW project.

Note TestStand can deploy only shared variables defined in a LabVIEW packed project library file using the LabVIEW Deploy Library step type if the packed project library file defines only shared variables and does not contain any VI files. TestStand cannot auto-deploy shared variables defined in a LabVIEW packed project library specified in a LabVIEW project or in a LabVIEW project library saved in an LLB specified in a LabVIEW project.

Click the **Create LabVIEW Project** or **Edit LabVIEW Project** buttons to open a new or existing project that a step uses. TestStand opens the project in the active version of the LabVIEW Development System.

The TestStand Deployment Utility includes, in built deployments, the projects and VIs that sequence files call.

Packed Project Libraries

LabVIEW packed project libraries are LabVIEW project libraries that package multiple LabVIEW-related files into a single file with a `.lvlibp` extension. In TestStand, you can directly open and call VIs exported from a LabVIEW packed project library, or you can access them through a LabVIEW project. Refer to the *LabVIEW Packed Project Libraries* section of Chapter 7, *Effectively Using LabVIEW with TestStand*, of the *Using LabVIEW and LabWindows/CVI with TestStand* manual for more information about editing, updating, patching, and deploying VIs in LabVIEW packed project libraries. Refer to the *NI LabVIEW Help* for more information about LabVIEW packed project libraries.

Note You must have LabVIEW 2010 or later to use LabVIEW packed project libraries in TestStand.

Classes

TestStand 2010 or later supports passing an instance of a LabVIEW Class data type, also called a LabVIEW class object, wired to the connector pane of a VI to and from LabVIEW 2009 SP1. In earlier versions of TestStand, passing LabVIEW Class objects required using the Flatten To String and Unflatten From String functions.

You can store LabVIEW class objects in TestStand object reference variables. TestStand does not have any information of the LabVIEW class object. You cannot access the properties or invoke the methods of the LabVIEW class object through the object reference directly from a TestStand step.

The lifetime of the LabVIEW class object is the lifetime of the TestStand object reference variable that holds the LabVIEW class object. When the object reference is released either by going out-of-scope or by obtaining a new value, the LabVIEW class object is also released. This behavior is different than other LabVIEW reference data types, such as an I/O reference.

Refer to the *Using LabVIEW Classes with TestStand* section of Chapter 7, *Effectively Using LabVIEW with TestStand*, of the *Using LabVIEW and LabWindows/CVI with TestStand* manual for more information about using LabVIEW class objects.

Note You must have LabVIEW 2009 SP1 or later to use the LabVIEW Class data type in TestStand

Automatically Detecting Version of LabVIEW RTE to Use

Select **Auto detect using VI version** for the LabVIEW Run-Time Engine Version option in the LabVIEW Adapter Configuration dialog box to run VIs using the LabVIEW RTE version in which the VIs were saved. In previous versions of TestStand, you could select only one version of the LabVIEW RTE to run the VI, and the LabVIEW Adapter used that version of the LabVIEW RTE to run all VIs. TestStand returns errors when you try to run VIs that do not match the version of the LabVIEW RTE you select. You can now use the Auto detect using VI version option to avoid these errors when you use VIs saved in different versions of LabVIEW.

Symmetric Multiprocessing in VIs Executed from TestStand

Previous versions of TestStand executed VIs whose execution system was set to **same as caller** using only one LabVIEW execution thread in the LabVIEW RTE. TestStand used multiple LabVIEW execution threads to execute VIs whose execution system was set to something other than **same as caller**, such as **other 1** or **other 2**. Users could not specify the number of threads used to execute VIs in the run-time engine or specify a thread affinity.

In TestStand 2010, you can use the LabVIEW Adapter Configuration dialog box to specify whether VIs execute using multiple threads, the number of such threads, and whether the additional LabVIEW execution threads adopt the thread affinity of the TestStand execution thread. These settings apply only to VIs with the preferred execution system set to **same as caller** and only when the VIs execute using the LabVIEW RTE. Refer to the *NI TestStand Help* for more information about using symmetric multiprocessing in VIs executed from TestStand.

Other LabVIEW Enhancements

TestStand 2010 includes the following additional enhancements for LabVIEW support:

- The TestStand - Start Modal Dialog VI and the TestStand - End Modal Dialog VI are now polymorphic VIs that accept the default sequence context or an engine context as an input.

.NET Adapter Enhancements

The .NET Adapter features a redesigned internal architecture that resolves many outstanding issues from previous versions of TestStand.

The adapter now supports *dot notation* for chaining calls, supports generic types, supports calling .NET string class members on TestStand strings, and supports storing any .NET object reference in a TestStand object reference variable. You can now store and retrieve .NET objects in TestStand object reference variables programmatically using the `PropertyObject.SetValInterface` and `PropertyObject.GetValInterface` methods of the TestStand API, and you can access .NET objects stored in TestStand object reference variables with the .NET Adapter or in .NET code modules. In addition, the Variables pane in the sequence editor now shows additional information about .NET objects stored in TestStand object reference variables at run time.

Dot Notation Calls

In previous versions of TestStand, calling a member or accessing a property or field in a .NET code module in which the member exists several layers deep within an object hierarchy required multiple TestStand steps to proceed through the layers. You needed to store the class instance from each subclass property in a variable.

In TestStand 2010, you can specify a chain of calls and property or field accessors using a single step, much like you can use a single statement in other languages such as C#. For example, you can call a method, access a property on its return value, and access a subfield of the property, all in a single step without having to store any intermediate values. In most cases, using a single step to complete multiple calls yields a performance improvement over making the same calls using multiple steps.

Note When you specify a chain of calls using dot notation, you can optionally store intermediate return values in TestStand variables. For example, you can specify subsequent .NET steps starting from one of the intermediate values instead of from the original root object to avoid the need to repeatedly call the same methods or repeatedly access the same properties or fields.

.NET Step Settings

The .NET Module tab on the Step Settings pane in the sequence editor and the Module tab of the Edit .NET Call dialog box in a TestStand User Interface include changes to improve usability and support the redesigned adapter internal architecture.

In previous versions of TestStand, you specified constructors in a separate window or dialog box, existing root class object variables in a separate expression control, and the kind of member you want to call.

In TestStand 2010, the .NET Module tab on the Step Settings pane and the Module tab of the Edit .NET Call dialog box now contain a .NET Invocation control, in which you can specify the chain of calls, from constructor to the final subproperty, by typing a member name and using the period (`.`) character to separate each method call or property or field access.

The .NET Invocation control includes pop-up lists that show the calls that are valid as you type or click within the control. You can use the pop-up lists to quickly select each member you want to call. Click the `<Click here to add call>` text or right-click, double-click, or press <Ctrl-Space> in the .NET Invocation control for an already-specified call to display the list of members you can call. Click the member in the list or type the member name to specify it. Press <Enter> or <Tab> to insert the currently selected call in the member list into the .NET Invocation control. Calls that you incorrectly or partially specify display in red.

After you specify the calls, the .NET Invocation control shows the signatures of each call in the chain of calls. Click anywhere on a call signature to select the call and view or edit its parameters in the Parameters Table. Click a particular parameter in the signature to highlight the parameter in the Parameters Table. Parameters that you incorrectly or partially specify display in red.

For the first call in the chain you specify in the .NET Invocation control, the member list shows only constructors, static members, and the following special case call types:

- **Use Existing Object**—Use this call type to specify an existing object to use instead of calling a constructor. The Parameters Table for this call contains an **Existing Object** parameter for you to specify. You must specify a valid TestStand object reference variable or basic type variable (in the case of the `System.String` type, `System.Int32` type, or other basic types) that refers to the object on which to perform the next call. If you specify a struct instead of a class for the root class of the step, you can also specify an instance of a TestStand custom data type that corresponds to the struct instead of a TestStand object reference variable because a struct is a .NET value type, and a class is a .NET reference type.

  Click the **Create Type from Struct** button in the Parameters Table to launch the Create Custom Data Type from Struct dialog box and create a named data type. If you specify an instance of a named data type, the .NET Adapter creates a temporary .NET struct and copies the corresponding fields between the struct and the TestStand variable.
- **Create Remote Object**—Use this call type to retrieve a .NET object using .NET remoting. The Parameters Table for this call contains a **Remote URI** parameter for you to specify the universal remote identifier (URI) of the server and an optional return value that specifies a TestStand object reference variable to store the object.

For subsequent calls in the chain you specify in the .NET Invocation control, the member list shows the instance members of the class that corresponds to the return value of the previous call.

The .NET Module tab on the Step Settings pane and the Module tab of the Edit .NET Call dialog box now also contain **Edit Code** and **Create Code** buttons, which you can click to launch Microsoft Visual Studio to edit or create the code for a call currently selected in the .NET Invocation control.

Note You must have a supported version of Visual Studio installed and a call with a valid class name currently selected in the .NET Invocation control to use the Create Code button. The currently selected call in the .NET Invocation control must be fully specified and valid for you to use the Edit Code button.

Previous versions of TestStand included a Use Step Load/Unload Options to Specify Object Creation and Lifetime option in the Configure Class Constructor window in the sequence editor and in the Configure Class Constructor dialog box in a user interface. In TestStand 2010, the Configure Class Constructor window and dialog box no longer exist. Instead, use the Advanced Settings window in the sequence editor or in a user interface to specify this option. Click the **Advanced Settings** button on the .NET Module tab of the Step Settings pane or on the Module tab of the Edit .NET Call dialog box to launch the Advanced Settings window.

Generics Support

The .NET Adapter supports all fully-specified generic types that the public members of an assembly use. Thus, the adapter supports any generic type used as a parameter, return value, property, or field type in a public member. For example, you can call an assembly with a method that takes the `List<Int32>` type as a parameter and you can create instances of the type and call members on the type.

API

The .NET Adapter supports new TestStand API classes, properties, and methods for programmatically specifying and reloading .NET module prototypes and accessing the settings of a .NET step. Refer to the *TestStand 2010 API and UI Controls Additions and Changes* topic in the *NI TestStand Help* for more information about new API classes, methods, and properties the .NET Adapter supports. Additionally, some API methods and properties are now obsolete.

Although you can still use the `ClassReference`, `ConstructorParameters`, `ConstructorPrototype`, `CreateObject`, `DisposeObject`, `IsConstructorPrototypeIncompatible`, `MemberFlags`, `MemberHelpString`, `MemberName`, `MemberType`, `Parameters`, `RemoteHost`, `SpecifyHostByExpression`, and `UnmappedConstructorArgumentValues` properties and the `ClearUnmappedConstructorArgumentValues`, `GetConstructorMetadataToken`, `GetMetadataToken`, `LoadConstructorPrototypeFromMetadataToken`, and `LoadPrototypeFromMetadataToken` methods of the `DotNetModule` class, they might not return exactly the same results as in previous versions of TestStand because of the significant changes made to the underlying `DotNetModule` data structure and architecture of the .NET Adapter in TestStand 2010. Most of the common usages of these properties and methods work as expected. However, if you use these properties and methods to modify or specify a `DotNetModule` object programmatically, you might need to modify the various settings on the `DotNetModule`

object in a different order than in previous versions of TestStand to obtain the expected result.

National Instruments recommends specifying the module settings in the following order:

- `DotNetModule.SetAssembly`
- `DotNetModule.ClassName`
- `DotNetModule.CreateObject`
- `DotNetModule.SpecifyHostByExpression`
- `DotNetModule.RemoteHost`
- `DotNetModule.ClassReference`
- `DotNetModule.DisposeObject`
- `DotNetModule.MemberName`
- `DotNetModule.MemberType`
- `DotNetModule.MemberFlags`
- `DotNetModule.LoadConstructorPrototypeFromMetadataToken`
- `DotNetModule.LoadPrototypeFromMetadataToken`
- `DotNetModule.ConstructorParameters`
- `DotNetModule.Parameters`

Consider using the newer API for specifying .NET code modules by adding `DotNetCalls` to the `DotNetModule.Calls` collection and using the `DotNetCall` interface to specify the members you want to call.

Deployment Enhancements

The TestStand Deployment Utility includes the following enhancements:

- You can now deploy files using a directory structure instead of a workspace file. Enable the **From Directory** option in the Deploy Files section on the System Source tab of the deployment utility to deploy files organized in a directory you specify. Enable the **Include SubDirectories** option to also include files in subdirectories of the directory you specify.
- You can now include files in a deployment without processing the files. Enable the **Include without Processing Item or Dependencies** option on the Distributed Files tab of the deployment utility to copy a file to the deployment image directory without any additional processing during the build. The deployment utility does not attempt to find any dependencies of the file during analysis. You might use this option to deploy files that have already been processed, such as third-party step types that use LabVIEW VIs or the LabVIEW User Interface.

You must ensure the LabVIEW RTE can run the VIs for which you enable this option. The deployment utility does not attempt to determine if VIs with this setting are broken but does check for broken VIs in the image directory after copying the VIs to warn you if VIs you do not process break processed code modules.

The deployment utility returns a warning if it includes a file in a deployment image directory as a dependency and you enabled the Include without Processing Item or Dependencies option for the file.

- The deployment utility supports deploying VIs called in the context of a LabVIEW project. By default, the deployment utility includes only the VIs steps call in a sequence. Enable the **Include All Files in LabVIEW Project** option on the Distributed Files tab of the deployment utility to include all the files in a LabVIEW project. When you select View Source from the View ring control on the Distributed Files tab of the deployment utility, the files appear under the LabVIEW project to which they belong. When you select View Destination from the View ring control, you can select the destination for the LabVIEW project, which determines the destination for all the files in the project.
- Enable the **Consolidate Files Shared By Projects** option in the LabVIEW VI Options dialog box for the deployment utility to consolidate all files in LabVIEW projects into one project. Enabling this option can improve the build speed and decrease the image size for deployments that call VIs in multiple LabVIEW projects. However, consolidating source distributions might not be possible for all sets of projects. Attempting to consolidate source distributions when not possible might result in slower build times. When you disable this option, the deployment utility builds one LabVIEW source distribution for each LabVIEW project. If multiple LabVIEW projects share files, the deployment utility includes multiple copies of these files.
- Enable the **Output VIs to a Packed Project Library** option to build a LabVIEW packed project library, and click the **Options** button next to the Output VIs to a Packed Project Library option to launch the Packed Project Library Options dialog box, which you use to configure options for how the deployment utility builds packed project libraries, such as the base library name, versioning information, descriptive information about the build, and source file options.
- You can now remove the block diagrams from VIs you deploy. Enable the **Remove Block Diagrams** option in the LabVIEW VI Options dialog box of the deployment utility to remove the block diagrams of VIs in a deployment to prevent users from viewing and editing the block diagrams. When you create a deployment with LabVIEW 8.6 or earlier, the deployment utility does not remove block diagrams for VIs that are part of XControls.

- If you do not have a LabVIEW development system installed on the computer to which you are deploying a system, the default value of the LabVIEW Run-Time Engine option in the LabVIEW Adapter Configuration dialog box is Auto detect using VI version, and you do not need to deploy the `TestExec.ini` file that specifies the LabVIEW RTE version to use.
- The deployment utility now reports a warning on the Build Status tab if the installer includes VIs but does not include the LabVIEW RTE required to run those VIs.
- You can now save a more detailed version of the status log, and you can also save an installer build log. Click the **Save Logs As** button on the Build Status tab of the deployment utility to save the status log, a detailed status log, and an installer build log in a ZIP file at a location you specify.

  The status log (`TSDU Status Log.txt`) contains information about the deployment process, such as when the process begins and when the deployment utility is building VIs and creating a project. This file also includes information about warnings and errors that occurred while deploying.

  The detailed status log (`TSDU Detailed Status Log.txt`) contains the same information about warning and errors as the standard status log. In addition, the detailed status log includes information about the operating system version, the TestStand version, and the LabVIEW version used to compile VIs.

  The installer build log (`TSDU Installer Build Log.txt`) contains details about the installer creation process that National Instruments can use to diagnose installer-related issues.

  The deployment utility stores the detailed status log and installer build log files in the Windows temporary directory.
- The Custom Commands dialog box now includes a Skip Custom Commands When Installing Silently option. When you enable this option, the installer does not execute custom commands when you run `setup.exe` with the `/q` command-line argument to run the installer in silent mode. Use this option to skip custom commands that require user interaction.
- The TestStand Deployment Utility includes several new warning and error messages.

Refer to Chapter 14, *Deploying TestStand Systems*, of the *NI TestStand Reference Manual* for more information about the TestStand Deployment Utility.

TestStand File Diff/Merge Application

TestStand 2010 includes a new stand-alone TestStand File Diff/Merge application that replaces the TestStand Differ application, which, in previous versions of TestStand, you could use to compare only non-type differences between two sequence files. The new file diff/merge application supports comparing and merging non-type differences among two or three sequence files, and comparing type differences among two sequence files or type palette files.

When you initiate a file comparison in the sequence editor, TestStand launches the file diff/merge application instead of performing the comparison directly in the sequence editor. You can navigate directly from the selected objects in the application to view the object in the sequence editor.

Note The file diff/merge application supports only comparing type palette files and does not support manually merging the type differences. The application can automatically resolve type differences when loading files to compare.

TestStand 2010 also includes a shared launching application you can use with third-party source code control (SCC) providers that support launching an external application to perform two- and three-way file diff and merge operations. The launching application determines the active version of TestStand and launches the corresponding TestStand diffing application that version of TestStand installs. Using the launching application alleviates the requirement to reconfigure the SCC provider each time you activate a different version of TestStand.

Select **Start»All Programs»National Instruments»TestStand»Tools» File Diff/Merge Tool** to launch the Select Files dialog box of the file diff/merge application. Use the Diff tab of the Select Files dialog box to specify two sequence files or type palette files you want to compare. Use the Merge tab to specify three sequence files you want to compare—a base file from which modified versions originate and two modified versions—and a fourth sequence file to become the resulting merged file to create.

Click the **Options** button to launch the File Diff/Merge Options dialog box, in which you can configure settings for the file diff/merge application, including options for loading and automatically merging types, managing type conflict resolution, and so on.

When you specify files to compare or merge in the Select Files dialog box and click OK, the TestStand File Diff/Merge window shows the following information:

- **File List**—The specified files you are comparing and a count of the changes, insertions, and deletions for each file.

- **Differences Table**—Displays the differences grouped by sequences, file global variables, file properties, and types for each specified file. You can expand and collapse each grouping, and you can use the navigation buttons to browse the differences.

When you use the Diff tab of the Select Files dialog box to compare two files, the TestStand File Diff/Merge window shows the changes, insertions, and deletions between the two files. Differences appear in the appropriate File column in bold text. For an item that has a difference, click the **Editing Options** button in the appropriate File column and select an option from the context menu for accepting the difference from the other file. The Diff tab provides the same functionality for comparing files as the TestStand Differ application and the Type File Differ application in previous versions of TestStand.

When you use the Merge tab of the Select Files dialog box to merge the differences among a base file and two modified versions of the file, the TestStand File Diff/Merge window shows the same information as when you compare two files but also includes a Merged column in the Differences table that you can use to specify which file version to accept in the final merged file.

When a change in either modified version of the file does not result in a conflict, such as when the change exists only in one or the other of the two modified versions of the file, the file diff/merge application automatically proposes a merged version that combines the changes from each modified version of the file.

When a change results in a conflict, such as when both modified versions of the file have modified the same property from the base file, the file diff/merge application cannot automatically determine which modified version to accept in the final merged file, and the conflict appears in bold italic text in the Differences table.

Click the **Merging Options** button in the Merged column and select an option from the context menu to manually select the version you want to accept in the final merged file. The background color of the Merged column changes depending on which version you accept. You can click the **Change File Background Color** button in the Files list to change the background color associated with each specified file. Use the Find Conflict navigation buttons in the Differences table to browse through all the conflicts.

The TestStand File Diff/Merge window also includes a Filters tab for hiding items that have no differences.

You can generate an XML report that documents the differences found or merges performed in the specified files. Select **File»Generate Report** or click the **Generate Report** button on the TestStand File Diff/Merge toolbar

to launch the Report Options dialog box, in which you can specify a file path for the generated XML report and enter comments for the report itself or for each file you are comparing, including the final merged file. You can also specify to view the report after it is generated.

Refer to the *NI TestStand Help* for more information about the file diff/merge application.

API Additions

Refer to the *TestStand 2010 API and UI Controls Additions and Changes* topic in the *NI TestStand Help* for more information about new API classes, methods, and properties. Additionally, some API methods and properties are now obsolete. The *NI TestStand Help* also documents API changes and additions for some earlier versions of TestStand.

Note Although some components, such as the sequence editor, expose .NET assemblies, TestStand does not support the use of undocumented assembly API.

Other TestStand Enhancements

TestStand 2010 includes support for 64-bit integers, support for C/C++ pointers, additional results enhancements, PropertyObject attributes enhancements, Visual Studio 2010 support, and other minor enhancements.

Support for 64-Bit Integers

TestStand 2010 supports 64-bit integer data types. You can select the representation of TestStand number and array of numbers data types as double-precision 64-bit floating (default), signed 64-bit integer, or unsigned 64-bit integer. Use the double-precision 64-bit floating representation, the default integer numeric type for previous versions of TestStand, for all 32-bit or smaller integer numeric data types.

All the TestStand adapters except the HTBasic Adapter support passing 64-bit integer parameters directly, creating and editing code that includes 64-bit integers, and verifying prototypes that use these data types when you use an application development environment that supports 64-bit integers.

Note **(LabWindows/CVI)** LabWindows/CVI 8.6 or earlier does not support calling the `PropertyObject.SetValVariant` or `PropertyObject.GetValVariant` methods with a 64-bit integer or 64-bit unsigned integer. Use the `PropertyObject.SetValInteger64`, `PropertyObject.SetValUnsignedInteger64`, `PropertyObject.GetValInteger64`, or `PropertyObject.GetUnsignedValInteger64` methods instead.

Note **(LabVIEW)** LabVIEW does not support calling the `PropertyObject.SetValInteger64`, `PropertyObject.GetValInteger64`, `PropertyObject.SetValInteger64ByOffset`, `PropertyObject.GetValInteger64ByOffset`, `PropertyObject.SetValUnsignedInteger64`, `PropertyObject.GetValUnsignedInteger64`, `PropertyObject.SetValUnsignedInteger64ByOffset`, or `PropertyObject.GetValUnsignedInteger64ByOffset` methods. Use the `PropertyObject.GetValVariant` or `PropertyObject.SetValVariant` methods instead.

For database logging, you can now use Big Integer and Unsigned Big Integer as the column or parameter types to log 64-bit integers in numeric fields in the database without losing precision. The TestStand default database schemas log numeric properties in string fields and arrays of numeric properties in binary fields. Thus, TestStand logs 64-bit integers in string fields and arrays of 64-bit integers in binary fields.

All TestStand report generators, except the ATML report generator, support generating reports that contain 64-bit integer data types. In HTML, text, and XML report formats, signed 64-bit integers use the `%d` format string, and unsigned 64-bit integers use the `%u` format string.

Note Oracle recommends using Oracle Provider 11.1.0.6.0 or later if you want to read 64-bit integer values stored in NUMBER columns.

Note Currently, Numeric Limit step types do not fully support using 64-bit integer data types. Numeric limits and the data source value must use the double-precision, floating-point representation.

Support for Pointers

You can now use the pointer/handle data type instead of a numeric data type for a pointer and handle parameters when you use the LabWindows/CVI and C/C++ DLL Adapters. TestStand stores pointer/handle parameters in object reference variables as pointers. In addition, when you use the .NET Adapter, you can now store IntPtrs and UIntPtrs in object reference variables as pointers instead of in number variables. Using object references instead of numbers to store pointers and handles means you can avoid changes to module specifications in the future if you eventually have 64-bit versions of code modules and applications.

Additional Results Enhancements

Additional results include the following enhancements:

- You can now specify to log additional results for parameters while configuring the module for a call. Enable the **Log** option in the

Parameters Table on the Module tab of the Step Settings pane or on the Module tab of an edit step dialog box to log the parameter of the step as an additional result. Enabling this option is equivalent to marking the checkbox next to the additional result name on the Additional Results panel or in the Additional Results dialog box. For in/out parameters, enabling this option enables the [In] parameter and the [Out] parameter on the Additional Results panel and in the Additional Results dialog box. This option is indeterminate for in/out parameters if you specify to log only the [In] parameter value or only the [Out] parameter value. If this option is indeterminate, a tooltip specifies whether the Additional Results panel or Additional Results dialog box specifies to log the [In] parameter value or the [Out] parameter value.

- You can use the Parameters Table context menu to launch the Additional Result panel or the Additional Results dialog box. On the Module tab of the Step Settings pane, right-click a parameter in the Parameters Table and select **Advanced Logging** from the context menu to launch the Additional Results panel, in which you can configure more settings for parameter additional results. On the Module tab of an edit step dialog box, right-click a parameter in the Parameters Table and select **Advanced Logging** from the context menu to launch the Additional Result dialog box, in which you can configure more settings for parameter additional results. You cannot edit the custom additional results in this dialog box.
- You can now use a dialog box to configure default additional results or additional results hints. Right-click a step type in the Types Window, select **Properties** from the context menu, click the **Advanced** button on the General tab of the Step Type Properties dialog box, and select **Default Additional Results** from the context menu to launch the Configure Default Additional Results dialog box, in which you can configure default additional results for the step. The step type must have a default module to configure default parameter additional results. Select **Preconfigured Additional Results** from the context menu when you click the **Advanced** button to launch the Configure Preconfigured Additional Results dialog box, in which you can configure the additional results hints for the step type. The additional results hints define a list of preconfigured custom additional results you can choose to log when you edit the additional results of a step in a user interface.
- For custom additional results, you can click the **Change Result Type** button in the Additional Results dialog box, panel, or edit tab to change the expected type of the additional result.

  When you click the **Change Result Type** button and select an array type, you can now expand the Name column of the Additional Results list to view the subproperties of the array elements of the selected type and specify settings for each individual subproperty of the array

elements. The settings you specify apply to every element of the array. For example, you can remove the checkmark next to the additional result name to not log that subproperty for every array element. You cannot specify a Value to Log expression for each individual array element subproperty. You must specify a Value to Log expression for the array itself.

- You can use the new `Step.LogAdditionalResult` method to programmatically log custom additional results.
- The Additional Results dialog box, edit tab, and panel now support cut, copy, and paste.
- For additional results, you no longer need to specify a Name expression if the Value to Log expression evaluates to an object with a name. If you do not specify a Name expression, TestStand uses the name of the property object the Value to Log expression evaluates to as the name of the additional result.

PropertyObject Attributes Enhancements

TestStand 2010 includes expanded support for PropertyObject attributes, including new buttons and context menu items to launch the Edit Attributes dialog box. Other enhancements include support for type definitions, report generation, searching, the TestStand `PropertyObject.GetXML` and `PropertyObject.SetXML` methods, and lookup strings.

You can now edit sequence attributes on the Sequences pane. Select **Advanced»Edit Attributes** from the Sequences pane context menu to launch the Attributes dialog box, in which you can create and edit attributes for the sequence. When the sequence includes attributes, you can click the **Edit Attributes** button in the **Sequence** column of the Sequences pane to launch the Attributes dialog box.

You can now create attributes for type definitions. Right-click a type in the Types Window of the sequence editor, select **Properties** from the context menu, click the **Advanced** button on the General tab of the Properties dialog box, and select **Attributes** to launch the Attributes dialog box for a type definition. When editing the attributes of a type definition, you can edit both attributes and type attributes. For a type definition, attributes determine the default values of the attributes of type instances. Type attributes are attributes for the type definition and do not exist for type instances.

You can now include attributes in reports. Enable the **Include Attributes** option on the Contents tab of the Report Options dialog box or enable the `PropFlags_IncludeInReport` flag on both an attribute and on the PropertyObject object that owns the top-level attribute to include attribute information in reports. Typically, a report generator does not recursively

apply the `PropFlags_IncludeInReport` flag from a property to its attributes.

The default TestStand database schemas do not log attribute values for result properties. Refer to the *Logging Attribute Values* topic in the *NI TestStand Help* for examples that demonstrate how you can customize a database logging schema to log attributes.

You can now include attributes in searches. Enable the **Attributes** option on the Filter tab of the Find/Replace in Files dialog box and in the Find/Replace dialog box to include the attributes and type attributes of variables and properties in search results.

The `PropertyObject.GetXML` and the `PropertyObject.SetXML` methods now support attribute information in the generated XML. Set the **GenerationOptions** parameter of the `PropertyObject.GetXML` method to **XMLOption_ExcludeAttributes** to exclude attributes and type attributes from the generated XML.

You can now specify an attribute or type attribute using a lookup string. Use `Attributes` in a lookup string to specify the attributes of the property object and use `TypeAttributes` to specify the type attributes of the property object. For example, the following lookup string accesses an attribute of a local variable from a sequence:

```
"Locals.Foo.Attributes.MyAttributeNamespace.Attribute1"
```

Note If a property object has a subproperty named `Attributes` or `TypeAttributes`, the subproperty takes precedence over the attributes or type attributes in a lookup string. As a result, some TestStand features, such as accessing attributes using lookup strings, undo, and find and replace, do not support attributes if the parent property object also has a subproperty named `Attributes`. National Instruments does not recommend using attributes and a subproperty named `Attributes` on the same property object. You can convert the subproperty to attributes or rename the subproperty.

Visual Studio 2010 Support

TestStand 2010 supports Visual Studio 2010 for the following code operations for the C/C++ DLL Adapter and .NET Adapter:

- Create Code option on the C/C++ DLL Module tab and .NET Module tab of the Step Settings pane in the sequence editor and on the Source Code tab of the Edit C/C++ DLL Call dialog box or Module tab of the Edit .NET Call dialog box in a user interface.
- Edit Code option on the C/C++ DLL Module tab and .NET Module tab of the Step Settings pane in the sequence editor and on the Source Code tab of the Edit C/C++ DLL Call dialog box or Module tab of the Edit .NET Call dialog box in a user interface.

- Step Into option on the Debug toolbar in the sequence editor or in the Debug menu in the sequence editor or a user interface to step into the code module the current C/C++ DLL or .NET step calls.

If you use Visual Studio 2010 with a project or solution file created in a previous version of Visual Studio, Visual Studio launches a conversion wizard.

You can also use Visual Studio 2010 to call .NET assemblies written with the .NET Framework 4.0. You must use a configuration file in the same directory as the executable to call assemblies from TestStand that target the .NET Framework 4.0 because the .NET Framework 4.0 uses the .NET Common Language Runtime (CLR) version 4.0. Refer to the *Using the .NET Framework* section of Chapter 5, *Module Adapters*, of the *NI TestStand Reference Manual* for an example configuration file and for more information about .NET CLR and .NET Framework versions TestStand requires.

The TestStand Version Selector also installs the TestStand toolbox and C++ include paths for Visual Studio 2010.

TestStand 2010 does not install a version of the *NI TestStand Help* for use within Visual Studio 2010. You must manually launch the help from TestStand.

Displaying Measurement Data as Y Plots or XY Plots in Reports

TestStand supports displaying one- and two-dimensional measurement array data as a graph in HTML and XML reports using the `TSGraphControl.ocx` ActiveX control, located in the `<TestStand Public>\Components\Tools\GraphControl` directory. The graph control supports displaying data as a Y plot or as an XY plot. The graph control supports orientation of the array data by rows or columns.

You can add `PropertyObject` attributes to specify the layout and orientation of the array data. The graph control uses these `PropertyObject` attributes to interpret the data and display the appropriate plot or plots.

By default, the graph control adds plots using the row-based orientation of the array data. You can add a `PropertyObject` attribute with the `TestStand.DataOrientation` namespace to configure the graph control to interpret the array data by columns.

Refer to the *Displaying Measurement Data as Graphs* topic in the *NI TestStand Help* for more information about graphs in HTML and XML reports.

Additional Enhancements

TestStand 2010 includes the following additional enhancements:

- You can now sort the contents of the Variables pane, the Property Browser panel of the Properties tab of the Step Settings pane, the Station Globals window, the User Manager window, the Types window, the Attributes dialog box, and the Sequences pane. However, TestStand does not sort array elements or parameters on the Variables pane.

  Click any column header to sort the view by the values of that column in increasing order. Click the column header again to sort the values in decreasing order. Click the column header a third time to restore the values to the unsorted order.
- In previous versions of TestStand, some of the example programs located in the `<TestStand Public>\Examples` directory included separate extensive documentation in Microsoft Word or text format. In TestStand 2010, that documentation now resides in the *Example Programs* section of the *NI TestStand Supplemental Reference Help*. For these examples, you can launch the corresponding help topic directly from the introductory Message Popup step in the example sequence file. Not all examples in the `<TestStand Public>\Examples` directory include separate extensive documentation.
- The *NI TestStand Help* and manuals include updated content. The help files are located in the `<TestStand>\Doc\Help` directory. The manuals are located in the `<TestStand>\Doc\Manuals` directory.

Note If you open help files directly from the `<TestStand>\Doc\Help` directory, National Instruments recommends that you open `TSHelp.chm` first because this file is a collection of all the TestStand help files and provides a complete table of contents and index.