LabVIEW™ Upgrade Notes

These upgrade notes describe the process of upgrading LabVIEW for Windows, Mac OS X, and Linux to LabVIEW 2011. Before you upgrade, read this document for information about the following topics:

- The order in which you should complete the tasks associated with upgrading LabVIEW.
- Potential compatibility issues you should know about prior to loading any VIs you saved in a previous version of LabVIEW.
- New features and behavior changes in LabVIEW 2011.

National Instruments recommends that you also review the upgrade notes for each LabVIEW version between the version from which you are upgrading and LabVIEW 2011. To access the documents, refer to the National Instruments Web site at ni.com/info and enter the Info Code for the appropriate LabVIEW version from the following list:

- *LabVIEW 8.2 Upgrade Notes*—upnote82
- *LabVIEW 8.5 Upgrade Notes*—upnote85
- *LabVIEW 8.6 Upgrade Notes*—upnote86
- *LabVIEW 2009 Upgrade Notes*—upnote9
- *LabVIEW 2010 Upgrade Notes*—upnote10

Refer to the *LabVIEW Help* for more information about LabVIEW 2011 features, as well as for information about LabVIEW programming concepts, step-by-step instructions for using LabVIEW, and reference information about LabVIEW block diagram nodes, tools, dialog boxes, and so on. Access the *LabVIEW Help* by selecting **Help»LabVIEW Help**.

Contents

NATIONAL INSTRUMENTS™

Upgrading to LabVIEW 2011

The following procedure suggests the order in which you should complete the tasks associated with upgrading to a new version of LabVIEW and which documents you should read as you complete these tasks. National Instruments recommends that you read both the *LabVIEW Release Notes* and this document before you upgrade to a new version of LabVIEW.

1. To verify that you are aware of all compatibility issues before you install LabVIEW, refer to the following sections of this document prior to installing the new version of LabVIEW:
   - *Upgrading to LabVIEW 2011*—This section includes instructions for upgrading toolkits and modules, copying environment settings and `user.lib` files from a previous version of LabVIEW, and converting VIs to LabVIEW 2011.
   - *Upgrade and Compatibility Issues*—This section includes compatibility issues that might affect VIs you upgrade from a previous version of LabVIEW to the new version of LabVIEW.

     Note You also can refer to the National Instruments Web site at `ni.com/info` and enter the Info Code `lvupgradetests` to download tests that can evaluate VIs for some compatibility issues.

   - *LabVIEW 2011 Features and Changes*—This section includes brief descriptions of the new features in this version of LabVIEW. Refer to the *LabVIEW Help* for more complete information about using these features. Access the *LabVIEW Help* by selecting **Help»LabVIEW Help**.
2. (Optional) Uninstall any previous version(s) of LabVIEW.
3. Install and activate the new version of LabVIEW. To verify that you complete all tasks associated with installing LabVIEW, refer to the following sections of the *LabVIEW Release Notes*:
   - *Installing LabVIEW 2011* and the appropriate subsection for the platform on which you are installing
   - *Installing LabVIEW Add-Ons* if you are installing LabVIEW toolkits or modules from media other than the LabVIEW Platform DVDs
   - **(Windows)** *Activating the LabVIEW License* and all subsections
   - (Optional) *Installing and Configuring Hardware* and the appropriate subsection for the platform on which you are installing
   - *Where to Go from Here*
4. Refer to the *LabVIEW Readme* for issues fixed in the new version of LabVIEW, information about known issues in the new version of LabVIEW, and documentation additions that are not reflected in the *LabVIEW Help*. To access the *LabVIEW Readme*, navigate to the `labview\readme` directory and open the `readme.html` file.
5. Copy environment settings from a previous version of LabVIEW. Refer to the *Copying Environment Settings from a Previous Version of LabVIEW* section of this document for more information about copying environment settings.

6. Copy `user.lib` files from a previous version of LabVIEW. Refer to the *Copying user.lib Files from a Previous Version of LabVIEW* section of this document for more information about copying `user.lib` files.
7. Convert VIs to LabVIEW 2011. Refer to the *Converting VIs* section of this document for more information converting VIs saved in a previous version of LabVIEW.

Upgrading from Previous Versions of LabVIEW

You can install LabVIEW 2011 without uninstalling previous versions of LabVIEW. While versions of LabVIEW might share components, upgrading to a new version of LabVIEW does not affect the performance of previous versions of LabVIEW on the computer because the new version installs in a different directory. LabVIEW 5.*x* and earlier install in the `labview` directory. LabVIEW 6.0 and later install in the `labview x` directory, where *x* is the version number.

Replacing an Existing Version of LabVIEW

To replace your existing version of LabVIEW, uninstall the existing version, run the LabVIEW 2011 installer, and set the installation directory to the `National Instruments` directory where you installed the previous version of LabVIEW.

(Windows XP) You also can replace the existing version of LabVIEW with LabVIEW 2011 by using the Add/Remove Programs applet in the Control Panel to uninstall the existing version of LabVIEW. **(Windows 7/Vista)** Use the Programs and Features applet in the Control Panel to uninstall the existing version of LabVIEW. The uninstaller does not remove any files you created in the top-level `labview` directory.

Note When you uninstall or reinstall LabVIEW, LabVIEW uninstalls the `.llb` files in the `vi.lib` directory, including any VIs and controls you saved in the `.llb` files. Save your VIs and controls in the `user.lib` directory to add them to the Controls and Functions palettes.

Copying Environment Settings from a Previous Version of LabVIEW

To use LabVIEW environment settings from a previous version of LabVIEW, copy the LabVIEW preferences file from the `labview` directory in which the previous version is installed.

Caution If you replace the LabVIEW 2011 preferences file with a preferences file from a previous version, you might override preference settings added to LabVIEW since the previous version.

After you install LabVIEW 2011, copy the LabVIEW preferences file to the LabVIEW 2011 directory.

(Windows) LabVIEW stores preferences in the `labview.ini` file in the `labview` directory.

(Mac OS X) LabVIEW stores preferences in the LabVIEW preferences text file at `~/Library/Preferences/LabVIEW 11.0 Preferences`.

(Linux) LabVIEW stores preferences at `/home/<username>/natinst/.config/LabVIEW-2011/labview.conf`, where <username> is the username of the user running the current instance of LabVIEW.

Note **(Linux)** The preferences format changed from *myapp*`.preferences_name: value` to `preference_name = value` in LabVIEW 2009. After you copy the LabVIEW preferences file to the LabVIEW 2011 directory, you must manually change the preferences to match the new format if necessary.

Copying user.lib Files from a Previous Version of LabVIEW

To use files from the `user.lib` directory of a previous version of LabVIEW, copy the files from the `labview` directory in which the previous version is installed. After you install LabVIEW 2011, copy the files to the `user.lib` directory in the LabVIEW 2011 directory.

Converting VIs

When you open a VI last saved in a previous version of LabVIEW, you must save the VI in LabVIEW 2011, or the conversion process, which uses extra memory resources, occurs every time you access the VI. Also, you might experience a large run-time degradation of performance for any VI that has unsaved changes, including a recompile.

For VIs last saved in LabVIEW 6.0 or later, LabVIEW 2011 automatically converts and compiles the VIs when you open them. However, for VIs last saved in LabVIEW 5.*x* and earlier, you must take extra steps to open and save the VI. Refer to the National Instruments Web site at `ni.com/info` and enter the Info Code `ext8h9` for more information about converting VIs to a different version of LabVIEW.

Note VIs you save in LabVIEW 2011 do not load in earlier versions of LabVIEW. To maintain compatibility with an earlier version, before you save VIs in LabVIEW 2011, keep a backup copy of VIs you plan to use in the previous version. You also can select **File»Save for Previous Version** to save VIs so they can run in a previous version.

If your computer does not have enough memory to convert all the VIs at once, convert the VIs in stages. Examine the hierarchy of VIs you want to convert and begin by loading and saving subVIs in the lower levels of the hierarchy. Then progress gradually to the higher levels of the hierarchy. Open and convert the top-level VI last. You also can select **Tools»Advanced»Mass Compile** to convert a directory of VIs. However, mass compiling converts VIs in a directory or LLB in a set order. Refer to the **Fundamentals»Creating VIs and SubVIs»How-To»Saving VIs»Mass Compiling VIs** topic on the **Contents** tab of the *LabVIEW Help* for a description of the order in which LabVIEW processes files when you mass compile. If the conversion process encounters a high-level VI first, mass compiling requires approximately the same amount of memory as if you opened the high-level VI first.

You can monitor memory usage by selecting **Help»About LabVIEW** to display a summary of the amount of memory you currently are using.

Upgrading Modules, Toolkits, and Instrument Drivers

If you are upgrading from a previous version of LabVIEW, you must install current, compatible versions of any modules, toolkits, or instrument drivers that you installed for the previous version of LabVIEW. The LabVIEW Platform DVDs include most modules and toolkits that are compatible with LabVIEW 2011. For those modules and toolkits that are not on the LabVIEW Platform DVDs, refer to the National Instruments Web site at `ni.com/info` and enter the Info Code `compat` for more information about which LabVIEW modules and toolkits are compatible with the current version of LabVIEW.

NI Modules and Toolkits

The following table lists whether to use the LabVIEW Platform DVDs or the module or toolkit installation media depending on your operating system and LabVIEW add-ons.

Operating System	Important Notes
Windows	Use the LabVIEW Platform DVDs to install LabVIEW 2011 and versions of modules and toolkits that are compatible with LabVIEW 2011. Additionally, you can choose to evaluate modules or toolkits you have not purchased. The LabVIEW Platform DVDs allow you to install new versions of a toolkit with LabVIEW 2011 without uninstalling or modifying previous versions. Refer to the *LabVIEW Release Notes* for information about installing LabVIEW, modules, and toolkits.
Mac/Linux; Windows, if the LabVIEW Platform DVDs do not include the module or toolkit	Use the installation media you received when you purchased the module or toolkit. Before using the installation media, make sure you have a compatible version of the module or toolkit you want to install. Refer to the National Instruments Web site at `ni.com/info` and enter the Info Code `compat` for more information about which LabVIEW modules and toolkits are compatible with the current version of LabVIEW. Then install the compatible modules and toolkits in the LabVIEW 2011 directory. Mass compile any VIs that you saved in previous versions of LabVIEW. Refer to the *Mass Compiling LabVIEW* section of this document for more information.

Note Some versions of toolkits do not work with LabVIEW 2011. Installing an incompatible toolkit might cause some features in the toolkit or LabVIEW to behave incorrectly. National Instruments recommends that you verify compatibility before attempting to install toolkits. Refer to the National Instruments Web site at `ni.com/info` and enter the Info Code `compat` for more information about which LabVIEW modules and toolkits are compatible with the current version of LabVIEW. **(Windows XP)** If you install an incompatible version and corrupt your LabVIEW 2011 installation, first uninstall the toolkit and then repair the LabVIEW installation using the Add/Remove Programs applet in the Control Panel. **(Windows 7/Vista)** Use the Programs and Features applet in the Control Panel to repair the installation.

Instrument Drivers

You must install current instrument drivers to control and communicate with any instrument hardware you plan to use. If you installed an instrument driver with a previous version of LabVIEW, you must reinstall that instrument driver with LabVIEW 2011 support using one of the following methods:

- **NI Modular Instrument drivers**—Use the NI Device Drivers DVD or CD to install NI Modular Instrument drivers.
- **Plug and Play instrument drivers**—**(Windows and Linux)** Use the NI Instrument Driver Finder to search for and install LabVIEW Plug and Play instrument drivers without leaving the LabVIEW development environment. Select **Help»Find Instrument Drivers** to launch the Instrument Driver Finder.
- **IVI driver or non-certified instrument drivers**—Use the Instrument Driver Network to search for and install an IVI driver or a non-certified driver. Refer to the National Instruments Web site at `ni.com/info` and enter the Info Code `ex3mbp` to access the Instrument Driver Network.

Note If you reinstall instrument drivers using the NI Instrument Driver Finder, National Instruments recommends that you mass compile the `labview\instr.lib` directory.

Third Party Add-Ons

Contact the vendor of third-party LabVIEW add-ons to determine whether the add-on is compatible with LabVIEW 2011 for your operating system. Make sure you mass compile any VIs that are related to the add-on.

Refer to the *Mass Compiling LabVIEW* section of this document for more information.

Mass Compiling LabVIEW

When you open a VI last saved in a previous version of LabVIEW, LabVIEW automatically converts and compiles the VI. You must save the VI in the current version of LabVIEW, or the conversion process, which uses extra memory resources, occurs every time you access the VI. If you install LabVIEW modules and toolkits that are not on the LabVIEW Platform DVDs or install any third-party add-ons, National Instruments recommends that you mass compile any VIs installed by the module, toolkit, or third-party add-on.

Refer to the **Fundamentals»Creating VIs and SubVIs»How-To»Saving VIs»Mass Compiling VIs** topic on the **Contents** tab of the *LabVIEW Help* for more information about mass compiling VIs.

Upgrading Additional National Instruments Software

There are known compatibility issues with LabVIEW 2011 and TestStand 4.2.1 and earlier verisons. Refer to the National Instruments Web site at `ni.com/info` and enter the Info Code `exvaku` to access the KnowledgeBase article that details these issues.

Refer to the `Readme.html` file for the version of NI TestStand you use, located on the NI TestStand CD and in the `<TestStand>\Doc` directory, for additional information about LabVIEW and NI TestStand issues.

You must use NI Spy 2.3 or later *or* NI I/O Trace 3.0 in LabVIEW 2011. NI Spy was renamed NI I/O Trace after NI Spy 2.7.2. NI I/O Trace is available on the NI Device Drivers media.

LabVIEW 2011 supports Measurement Studio 8.0 and later. Refer to the National Instruments Web site at `ni.com/info` and enter the Info Code `exd8yy` to access the Upgrade Advisor and purchase Measurement Studio 8.0 or later.

Upgrade and Compatibility Issues

Refer to the following sections for upgrade and compatibility issues specific to different versions of LabVIEW. Refer to the National Instruments Web site at `ni.com/info` and enter the Info Code `lvupgrade` for recommendations about how to upgrade to the latest version of LabVIEW.

Refer to the `readme.html` file in the `labview` directory for information about known issues in the new version of LabVIEW, additional compatibility issues, and information about late-addition features in LabVIEW 2011.

Upgrading from LabVIEW 2010

You might encounter the following compatibility issues when you upgrade to LabVIEW 2011 from LabVIEW 2010.

Platforms Supported

LabVIEW 2011 does not support Windows XP Service Pack 1 or earlier.

Disk Space Requirements

(Windows) LabVIEW 2011 requires at least 340 MB for the LabVIEW Run-Time Engine installation or 3.3 GB of disk space for the complete LabVIEW installation.

(Mac OS X) LabVIEW 2011 requires at least 563 MB of disk space for the LabVIEW Run-Time Engine installation or 1.2 GB of disk space for the complete LabVIEW installation.

(Linux) LabVIEW 2011 requires at least 125 MB of disk space for the LabVIEW Run-Time Engine installation or 1.2 GB of disk space for the complete LabVIEW installation.

Refer to the *LabVIEW Release Notes* for additional system requirements for LabVIEW 2011.

VI and Function Behavior Changes

In LabVIEW 2011, the **multicast addr** input of the UDP Multicast Open VI is a required input. Also, the **port** output is renamed **port out**.

Deprecated VIs, Functions, and Nodes

The Zero Phase Filter VI is rewritten in LabVIEW 2011 to remove the **init/cont** input from each polymorphic instance. To use the new functionality, replace instances of the Zero Phase Filter VI from previous versions of LabVIEW with VIs of the same name from the Functions palette.

Property, Method, and Event Behavior Changes

In LabVIEW 2010, the Clear Compiled Object Cache method clears the object cache associated with a specific target. In LabVIEW 2011, the Clear Compiled Object Cache method clears the entire user cache for the running version of LabVIEW. Therefore, although VIs created in LabVIEW 2010 that contain the Clear Compiled Object Cache method do not break in LabVIEW 2011, they delete more VI object files than they did previously, which causes the associated VIs to recompile when loaded.

Deprecated Properties, Methods, and Events

LabVIEW 2011 does not support the Subsystem From Selection method of the SimDiagram class.

Migrating Build Specifications for Targets That Do Not Support SSE2 Instructions

To migrate a build specification for a target that does not support SSE2 instructions to LabVIEW 2011, you must disable the SSE2 optimizations for the build specification. If you do not disable the optimizations, LabVIEW still allows you to build the associated application, but the application cannot run on the intended target.

Refer to the **Fundamentals»Building and Distributing Applications»Configuring Build Specifications»Verifying That Target Hardware Supports SSE2 Instructions** topic on the **Contents** tab of the *LabVIEW Help* for information about which hardware types support SSE2 instructions.

Polymorphic VI Terminals that Support 64-bit and Double-Precision Numeric Data Types

LabVIEW coerces extended-precision numeric data to double-precision numeric data if you wire the data to a terminal of a polymorphic VI that supports both the double-precision numeric and 64-bit integer data types. This behavior matches the behavior in LabVIEW 8.5 and 8.6. However, in LabVIEW 8.2, 2009, and 2010, LabVIEW selects the instance with the 64-bit signed integer.

Improved Error Reporting for Certain LabVIEW Shared Libraries

When you call a LabVIEW shared library with the Call Library Function Node in previous versions of LabVIEW, the shared library fails to execute on target computers that do not have required resources installed. However, in those situations, the shared libraries do not automatically return an error or otherwise indicate that execution failed. In LabVIEW 2011, when the Call Library Function Node calls these shared libraries, LabVIEW returns an error to indicate the failure. Therefore, affected LabVIEW shared libraries that misleadingly do not return an error in LabVIEW 2010 *do* return an error in LabVIEW 2011.

This error-reporting enhancement affects, but is not limited to, VIs that call LabVIEW shared libraries with any of the following characteristics:

- A VI inside the shared library uses licensed features that are not installed on the target computer.
- A VI inside the shared library uses a Call Library Function Node whose associated shared library is not installed on the target computer.
- The VIs inside the shared library were compiled using the SSE2 optimizations but the target computer does not support SSE2 instructions.

Changes to the Locations LabVIEW Searches for NI Example Finder Data Files

LabVIEW 2011 searches for NI Example Finder data files (`.bin3`) in fewer locations than previous versions of LabVIEW. To ensure LabVIEW finds example VIs you create for the NI Example Finder, you must place the `.bin3` files in one of the following directories:

- `labview\examples\exbins`—Previous versions of LabVIEW allowed you to place the `.bin3` files anywhere within the `examples` directory.
- `labview\instr.lib`
- `labview\user.lib`

Upgrading from LabVIEW 2009

You might encounter the following compatibility issues when you upgrade to LabVIEW 2011 from LabVIEW 2009. Refer to the *Upgrading from LabVIEW 2010* section of this document for information about other upgrade issues you might encounter.

Platforms Supported

- LabVIEW 2010 and later supports Windows 7.
- LabVIEW 2010 and later does not support Windows 2000.

Refer to the *LabVIEW Release Notes* for additional system requirements for LabVIEW 2011.

VI and Function Behavior Changes

The following VIs use a higher attenuation than the value of the **stopband attenuation** input to design an elliptic filter when the filter **order** is high:

- Elliptic Coefficients
- Elliptic Filter
- Elliptic Filter PtByPt

VISA Find Resource Function

In LabVIEW 2010 and later, the VISA Find Resource function returns error code –1073807343 if the system does not locate any devices.

Deprecated VIs, Functions, and Nodes

LabVIEW 2010 and later does not support the following VIs, functions, and nodes:

- **Code Interface Node**—Use the Call Library Function Node instead.
- **Convert TDM to TDMS**—Use the Convert to TDM or TDMS VI instead. This VI converts a file to the `.tdm` or `.tdms` file format.
- **Convert TDMS to TDM**—Use the Convert to TDM or TDMS VI instead.
- **Get Property Type**—Use the Get Property Info VI instead. This VI returns information about the properties of a data file, channel group, or channel.
- **FFT Power Spectrum**—Use the FFT Power Spectrum and PSD VI instead.
- **FFT Power Spectral Density**—Use the FFT Power Spectrum and PSD VI instead.
- **List Properties**—Use the Get Property Info VI instead.
- **Merge Errors** VI—Use the Merge Errors function instead.
- **Merge Queries**—Use the Merge Storage Refnums VI instead.

Floating-Point Math Operations

Due to changes to the LabVIEW compiler, the results of several mathematical operations performed using floating-point numbers might differ from results returned in previous versions of LabVIEW. The accuracy of algorithms written in LabVIEW using floating-point numbers is the same and in many cases improved in LabVIEW 2010 and later. However, in a few operations the results might be less accurate

than in previous versions because LabVIEW 2010 and later implements functions internally with the same numeric precision as the input data types rather than a higher numeric precision than the input data types as in previous versions. The acceptable error for the results of these operations is still appropriate for the data types of the inputs.

Note Refer to the National Instruments Web site at ni.com/info and enter the Info Code exdj8b for more information about mathematical operations using floating-point numbers.

Creating LabVIEW Classes

In LabVIEW 2009 or earlier, you can create a class with a strictly typed VI refnum that includes itself or a child class in the connector pane of the VI. In LabVIEW 2010 and later, the class breaks unless you use a VI refnum that is not strictly typed or remove the VI refnum from the private data control.

Building an Installer (Windows)

In LabVIEW 2010 and later, if you load a project with an installer that requires Windows 2000 or later, LabVIEW updates the system requirements to Windows XP or later. After you install LabVIEW 2010 and later, you cannot use a previous version of LabVIEW on the computer to build installers that run on Windows 2000.

Using the Correct Calling Convention in a Call Library Function Node

In LabVIEW 8.2, LabVIEW 8.6, and LabVIEW 2009, when you specify the incorrect calling convention for a Call Library Function Node, LabVIEW recovers from the error and uses the correct calling convention. LabVIEW 2010 and later do not perform this check, which requires you to select the correct calling convention yourself. Therefore, if you convert VIs that contain Call Library Function Nodes from LabVIEW 8.2, LabVIEW 8.6, or LabVIEW 2009 to LabVIEW 2010 or later, the VIs will crash if they have the incorrect calling convention selected.

Complete the following steps to prepare a VI that contains Call Library Function Nodes to be converted to LabVIEW 2010 or later:

1. Open the VI in the LabVIEW version in which it was last saved.
2. Right-click each Call Library Function Node and select **Configure** from the shortcut menu to display the **Call Library Function** dialog box.
3. Click the **Error Checking** tab.
4. Place a checkmark in the **Maximum** checkbox to enable maximum error checking. This selection causes LabVIEW to notify you at run time if you select the incorrect calling convention.
5. Click the **OK** button.
6. After you select maximum error checking for each Call Library Function Node, run the VI.
7. Select the correct calling convention for each Call Library Function Node that returns an error.

After you resolve all calling convention errors, you can convert the VI to LabVIEW 2010 or later.

Upgrading from LabVIEW 8.6

You might encounter the following compatibility issues when you upgrade to LabVIEW 2011 from LabVIEW 8.6. Refer to the *Upgrading from LabVIEW 2009* and *Upgrading from LabVIEW 2010* sections of this document for information about other upgrade issues you might encounter.

Refer to the *LabVIEW Release Notes* for system requirements for LabVIEW 2011.

VI and Function Behavior Changes

The behavior of the following VIs and functions changed in LabVIEW 2009 and later.

Bluetooth VIs and Functions

You must have Windows XP Service Pack 2 or later installed to use the VIs and functions on the Bluetooth palette.

Signal Generation VIs

The following VIs on the Signal Generation palette are rewritten in LabVIEW 2009 and later. To use the new functionality, replace these VIs with VIs of the same name from the Functions palette:

- Bernoulli Noise
- Binary MLS
- Binomial Noise
- Gamma Noise
- Gaussian White Noise
- Poisson Noise
- Uniform White Noise

Miscellaneous VI and Function Behavior Changes

LabVIEW 2009 and later includes the following miscellaneous VI and function behavior changes:

- If you wire a value that has a unit with an odd exponent to the square root function, the wire breaks because LabVIEW does not support units with fractional exponents.
- The Bessel Coefficients VI is rewritten to implement cutoff frequencies more accurately. As a result, the Bessel Coefficients VI and any calling VIs might run more slowly than in previous versions of LabVIEW.
- LabVIEW deploys Web services to version-specific directories. For example, a typical root location for deployed Web services in LabVIEW 2009 is `C:\Documents and Settings\All Users\Application Data\National Instruments\Web Services 2009 32-bit`. You must redeploy any Web services created in a previous version of LabVIEW to use the Web services in LabVIEW 2009 or later. To delete a Web service deployed by a previous version of LabVIEW, you must manually remove it from the deployed location.
- The Integral x(t) VI is rewritten. To use the new functionality, replace this VI with the Integral x(t) VI from the Functions palette.
- The **section** and **refnum** inputs of the Get Key Names VI are required inputs.
- The **refnum** input of the Get Section Names VI is a required input.
- The **refnum** input of the Not A Config Data Refnum VI is a required input.
- You no longer can use the Check if File or Folder Exists VI to check for paths in stand-alone applications or shared libraries.

Deprecated VIs and Functions

LabVIEW 2009 and later does not support the following VIs and functions:

- **LToCStr**—Use the LToCStrN function instead. The LToCStrN function differs from the LToCStr function because it takes a parameter specifying the size of the C string buffer to which LabVIEW copies the string. These functions are Code Interface Node (CIN) functions. The Code Interface Node is deprecated in LabVIEW 2010 and later. Use the Call Library Function Node instead.
- **Open Config Data (compatibility)**—Use the Open Config Data VI instead. The Open Config Data VI differs from the Open Config Data (compatibility) VI because the Open Config Data VI includes the **file created?** output.

- **Sound VIs (Mac OS X)**—Use the Sound VIs instead. LabVIEW 2009 and later supports the same API for Windows, Mac OS X, and Linux.
- **Unconstrained Exponential Fit**—Use the Exponential Fit VI instead. The Exponential Fit VI differs from the Unconstrained Exponential Fit VI because the Exponential Fit VI does not include the **refine?** input but does include the **parameter bounds** input and **offset** output.
- **Unconstrained Gaussian Peak Fit**—Use the Gaussian Peak Fit VI instead. The Gaussian Peak Fit VI differs from the Unconstrained Gaussian Peak Fit VI because the Gaussian Peak Fit VI includes the **parameter bounds** input and **offset** output.

Deprecated Properties, Methods, and Events

LabVIEW 2009 and later does not support the following properties, methods, and events:

- Bus Name property of the DigitalGraph class. Use the Plot Name property instead.
- Callees' Names property of the VI class. Use the Get VI Dependencies (Names and Paths) method instead. The Get VI Dependencies (Names and Paths) method provides the same functionality as the Callees' Names property when you use the default values for all input parameters.
- Callees property of the VI (ActiveX) class.

Renamed Properties, Methods, and Events

- In LabVIEW 2009 and later, the XML Parser classes do not include XML in their names. For example, XML_Attributes becomes Attributes.
- The following properties, methods, and events are renamed in LabVIEW 2009 and later.

Class	LabVIEW 8.6 Name	LabVIEW 2009 and Later Name	Type
Document	Do Namespaces	Process Namespaces	Property
Document	Do Schema	Process Schema	Property
Variable	Alarming:BadStatus:AckType	Alarming:BadStatus:Ack Type	Property
Variable	Alarming:BadStatus:AllowLog	Alarming:BadStatus:Allow Log	Property
Variable	Alarming:Boolean:AckType	Alarming:Boolean:Ack Type	Property
Variable	Alarming:Boolean:AlarmOn	Alarming:Boolean:Alarm On	Property
Variable	Alarming:Boolean:AllowLog	Alarming:Boolean:Allow Log	Property
Variable	Alarming:Hi:AckType	Alarming:Hi:Ack Type	Property
Variable	Alarming:Hi:AllowLog	Alarming:Hi:Allow Log	Property
Variable	Alarming:HiHi:AckType	Alarming:HiHi:Ack Type	Property
Variable	Alarming:HiHi:AllowLog	Alarming:HiHi:Allow Log	Property
Variable	Alarming:Lo:AckType	Alarming:Lo:Ack Type	Property
Variable	Alarming:Lo:AllowLog	Alarming:Lo:Allow Log	Property
Variable	Alarming:LoLo:AckType	Alarming:LoLo:Ack Type	Property
Variable	Alarming:LoLo:AllowLog	Alarming:LoLo:Allow Log	Property
Variable	Alarming:RateOfChange:AckType	Alarming:RateOfChange:Ack Type	Property
Variable	Alarming:RateOfChange:AllowLog	Alarming:RateOfChange:Allow Log	Property

Class	LabVIEW 8.6 Name	LabVIEW 2009 and Later Name	Type
Variable	Alarming:U32BitField:AckType	Alarming:U32BitField:Ack Type	Property
Variable	Alarming:U32BitField:AlarmOn	Alarming:U32BitField:Alarm On	Property
Variable	Alarming:U32BitField:AllowLog	Alarming:U32BitField:Allow Log	Property
Variable	Alarming:U32BitField:SelectMask	Alarming:U32BitField:Select Mask	Property
Variable	Logging:LogData	Logging:Log Data	Property
Variable	Logging:LogEvents	Logging:Log Events	Property
Variable	Logging:TimeResolution	Logging:Time Resolution	Property
Variable	Logging:ValueResolution	Logging:Value Resolution	Property
Variable	Network:AccessType	Network:Access Type	Property
Variable	Network:BuffSize	Network:Buffer Size	Property
Variable	Network:ElemSize	Network:Element Size	Property
Variable	Network:PointsPerWaveform	Network:Points Per Waveform	Property
Variable	Network:ProjectBinding	Network:Project Binding	Property
Variable	Network:ProjectPath	Network:Project Path	Property
Variable	Network:UseBinding	Network:Use Binding	Property
Variable	Network:UseBuffering	Network:Use Buffering	Property
Variable	Real-Time:ArrayLength	Real-Time:Array Length	Property
Variable	Real-Time:BufferLength	Real-Time:Buffer Length	Property
Variable	Real-Time:DatapointsInWaveform	Real-Time:Datapoints In Waveform	Property
Variable	Real-Time:UseBuffering	Real-Time:Use Buffering	Property
Variable	Scaling:EngineeringMax	Scaling:Engineering Max	Property
Variable	Scaling:EngineeringMin	Scaling:Engineering Min	Property
Variable	Scaling:InvertMask	Scaling:Invert Mask	Property
Variable	Scaling:RawMax	Scaling:Raw Max	Property
Variable	Scaling:RawMin	Scaling:Raw Min	Property
Variable	Scaling:SelectMask	Scaling:Select Mask	Property

Application Builder Changes

LabVIEW 2009 and later includes the following Application Builder changes.

File Layout Changes

In LabVIEW 8.6, the Application Builder saves VIs and library files in a flat list within the application and saves VIs with conflicting filenames outside the application in separate directories. In LabVIEW 2009 and later, the Application Builder stores source files within the application using a layout similar to the

directory structure of the source files on disk. This internal file layout preserves source file hierarchy inside the application.

If you call VIs dynamically, use relative paths to ensure the application loads the VIs correctly at run time.

Custom Configuration File Changes

In LabVIEW 8.6 and earlier, when you build a stand-alone application that includes a custom configuration file, LabVIEW appends LabVIEW environment settings to the existing contents of the file when the following conditions are true:

- The custom configuration file has the same name as the application.
- The custom configuration file is in the same directory as the application.
- In the build specification for the application, the **Use custom configuration file** checkbox on the **Advanced** page of the **Application Properties** dialog box does not contain a checkmark.

When these conditions are true in LabVIEW 2009 and later, LabVIEW overwrites the contents of custom configuration files with LabVIEW environment settings.

Case Structure Output Tunnel Changes

LabVIEW 2009 and later determines the data type from a Case structure output tunnel by using a data type that can handle all cases in the structure, including cases that never execute. For example, consider a Case structure with two cases, TRUE and FALSE. In the TRUE case, an 8-bit unsigned integer is wired to an output tunnel. In the FALSE case, a 32-bit unsigned integer is wired to the output tunnel. In LabVIEW 8.5*.x* and 8.6*.x*, if you wire a constant to select the TRUE case, the data type from the output tunnel is 8-bit unsigned integer because the constant prevents the FALSE case from executing. In LabVIEW 2009 and later, if you wire a constant to select the TRUE case, the data type from the output tunnel is 32-bit unsigned integer.

This change in behavior might cause VIs created in LabVIEW 8.5*.x* and 8.6*.x* to break in LabVIEW 2009 and later if the output data type is a fixed-point number or fixed-sized array.

Custom Icon Editor VI Changes

In LabVIEW 8.6 or earlier, when LabVIEW calls a VI that is a custom icon editor, LabVIEW automatically opens the front panel of the VI. In LabVIEW 2009 and later, you must configure a VI that is a custom icon editor to open its own front panel on call. For simple VIs that do not need to rearrange their front panels before they open, use the Execution:Show Front Panel on Call property. For more complex VIs that need to rearrange their front panels before they open, use the Front Panel:Open method.

Custom Probes Changes (Linux)

Custom probes you save in LabVIEW 8.6 or earlier do not open in LabVIEW 2009 and later. You must manually copy the custom probes from the `LabVIEW Data` directory of the previous version of LabVIEW into the `LabVIEW Data` directory of LabVIEW 2009 and later. You can find the `LabVIEW Data` directory for LabVIEW 2009 and later at `/home/<username>/LabVIEW Data`.

.NET Changes

Creating and communicating with .NET objects requires the .NET Framework 2.0 or later.

LabVIEW MathScript Changes

LabVIEW MathScript is no longer a part of the Full and Professional Development Systems. In LabVIEW 2009 and later, LabVIEW MathScript becomes the LabVIEW MathScript RT Module. You cannot run VIs from previous versions of LabVIEW that contain MathScript Nodes until you install and activate the MathScript RT Module or remove the MathScript Nodes from the VIs. If you have already

purchased the MathScript RT Module, select **Help»Activate LabVIEW Components** to activate the product.

Upgrading from LabVIEW 8.5

You might encounter the following compatibility issues when you upgrade to LabVIEW 2011 from LabVIEW 8.5. Refer to the *Upgrading from LabVIEW 8.6*, *Upgrading from LabVIEW 2009*, and *Upgrading from LabVIEW 2010* sections of this document for information about other upgrade issues you might encounter.

Platforms Supported

LabVIEW 8.6 and later does not support Macintosh computers with PowerPC processors.

Refer to the *LabVIEW Release Notes* for additional system requirements for LabVIEW 2011.

VI and Function Behavior Changes

The behavior of the following VIs and functions changed in LabVIEW 8.6 and later.

Report Generation VIs

The VIs on the Report Generation palette were rewritten using LabVIEW classes. The **report in** control and **report out** indicator changed from reference number data types to LabVIEW class data types. If you did not create constants, controls, or indicators by right-clicking the type definition refnum, the VIs might not work correctly because LabVIEW cannot update those objects for you. Additionally, any Call By Reference node that calls the previous refnum data type of the **report in** and **report out** parameters will not work as expected.

If you create HTML reports using the VIs on the Report Generation palette to run on a target, make sure you reference the target when you create a report. If you create an HTML report on a host computer and then deploy to a target without referencing the target, VIs appear broken and do not run.

The **orientation** input of the Set Report Orientation VI changed from a word unsigned integer number (U16) to a long integer number (I32).

The default value for the **include Express VI configuration information** input of the Append VI List of SubVIs to Report VI changed from TRUE to FALSE.

The default value for both instances of the **show grid lines** input of the Append Table to Report VI changed from FALSE to TRUE.

External Code (DLLs and CINS)

The memory manager functions include only one zone of memory, DS (data space). If you work with C or C++ CINs or DLLs that manage LabVIEW memory, replace all references to AZ (application zone) memory functions with the DS equivalent function. The Code Interface Node is deprecated in LabVIEW 2010 and later. Use the Call Library Function Node instead.

Miscellaneous VI and Function Behavior Changes

LabVIEW 8.6 and later includes the following miscellaneous VI and function behavior changes:

- The STFT Spectrograms VI was rewritten with two new inputs in LabVIEW 8.6 and later. Replace versions of this VI from previous versions of LabVIEW with an STFT Spectrograms VI from the Functions palette to use the new functionality.
- Many of the Mathematics and Signal Processing VIs changed from non-reentrant VIs to reentrant VIs. Because of these changes, you should not call many of these VIs from a reentrant VI set to share clones between instances. Refer to the National Instruments Web site at `ni.com/info` and enter the Info Code `exrehi` for more information about which VIs you cannot call from a VI set to share clones between instances.

- LabVIEW 8.6 and later forces single-process shared variables to be target-relative. You cannot configure single-process shared variables to be absolute.
- If you wire an empty path to the **path in** input of the Call Library Function Node, LabVIEW no longer returns an error.
- The **output element** output of the Get Report Type VI changed to **report type**. LabVIEW automatically renames and reconnects code you take from previous LabVIEW versions and insert into LabVIEW 8.6 and later. However, the VI breaks if you use the Call By Reference node to call the Get Report Type VI.
- The **report type** input of the New Report VI is a required input. You must wire data to this input.

Deprecated VIs and Functions

LabVIEW 8.6 and later does not support the following VIs and functions:

- **Nonlinear System Single Solution**—Use the nD Nonlinear System Single Solution VI instead. The nD Nonlinear System Single Solution VI differs from the Nonlinear System Single Solution VI because it is reentrant.
- **Nonlinear System Solver**—Use the nD Nonlinear System Solver VI instead. The nD Nonlinear System Solver VI differs from the Nonlinear System Solver VI because it is reentrant.
- **Create Semaphore**—Use the Obtain Semaphore Reference VI instead. The Obtain Semaphore Reference VI differs from the Create Semaphore VI because if you use the Create Semaphore VI multiple times to create more than one semaphore with the same name, LabVIEW creates multiple copies of a single reference to that semaphore. However, if you use the Obtain Semaphore Reference VI to obtain multiple references to the same semaphore, each reference number is unique. Because LabVIEW does not automatically convert existing VIs to use the Obtain Semaphore Reference VI, you must manually update VIs saved in a previous version of LabVIEW.
- **Destroy Semaphore**—Use the Release Semaphore Reference VI instead. The Release Semaphore Reference VI differs from the Destroy Semaphore VI because if you use the Destroy Semaphore VI to destroy a semaphore, LabVIEW also destroys any other copies of the reference to that semaphore. However, if you use the Release Semaphore Reference VI to release a reference to a semaphore, other references to that semaphore remain valid, and LabVIEW destroys the semaphore only if no other references to the semaphore exist. Because LabVIEW does not automatically convert existing VIs to use the Release Semaphore Reference VI, you must manually update VIs saved in a previous version of LabVIEW.

Note To avoid unexpected results, do not pass references you use with the Create Semaphore VI or the Destroy Semaphore VI to the Obtain Semaphore Reference VI or the Release Semaphore Reference VI, and vice versa.

- **Xmath script node**—Use the MathScript Node instead. Because the MathScript syntax is different from the Xmath syntax, you might need to modify existing scripts to work in the MathScript Node. In LabVIEW 2009 and later, LabVIEW MathScript becomes the LabVIEW MathScript RT Module. You cannot run VIs from previous versions of LabVIEW that contain MathScript Nodes until you install and activate the MathScript RT Module or remove the MathScript Nodes from the VIs.

Property, Method, and Event Behavior Changes

The behavior of the following properties, methods, and events changed in LabVIEW 8.6 and later:

- The Camera Controller:Type property of the SceneGraphDisplay class includes an **Oriented** value.
- The Camera Controller:Type property of the SceneWindow class includes an **Oriented** value.
- The Scaling:Enabled property of the Variable class allows you to use scaling only for network-published shared variables, I/O variables, or I/O aliases.

Deprecated Properties, Methods, and Events

LabVIEW 8.6 and later does not support the following properties, methods, and events:

- Control Value:Set [Flattened] method of the VI class. Use the Control Value:Set method instead.
- Control Value:Get All [Flattened] method of the VI class. Use the Control Value:Get All method instead.
- Control Value:Get [Flattened] method of the VI class. Use the Control Value:Get method instead.
- VIModificationBitSet property of the VI (ActiveX) class.
- Modifications:VI Modifications Bitset property of the VI class. Use the new Modifications:VI Modifications Bitset property instead. In LabVIEW 8.5 and earlier, the Modifications:VI Modifications Bitset property returns a 32-bit value. In LabVIEW 8.6 and later, the new Modifications:VI Modifications Bitset property returns a 64-bit value.

Renamed Properties, Methods, and Events

The following properties, methods, and events are renamed in LabVIEW 8.6 and later.

Class	LabVIEW 8.5 Name	LabVIEW 8.6 and Later Name	Type
GObject	Bounds:Height	Bounds:Area Height	Property
GObject	Bounds:Width	Bounds:Area Width	Property
ProjectItem	Disconnect from Disk	Stop Auto-populating	Method
TreeControl	Expand/Contract Symbol:Show at Indent Level 0	Expand/Contract Symbol:Show Symbol at Root	Property
VI	Control Value:Set [Variant]	Control Value:Set	Method
VI	Control Value:Get [Variant]	Control Value:Get	Method
VI	Control Value:Get All [Variant]	Control Value:Get All	Method

Shared Variable Changes

When a VI that includes a shared variable on the block diagram is running or reserved to run, you cannot edit the following properties of the shared variable until the VI stops running and is no longer reserved to run:

- All properties on the **Variable** page of the **Shared Variable Properties** dialog box.
- **Use Buffering** properties on the **Network** page of the **Shared Variable Properties** dialog box.
- **(RT Module)** All properties on the **Real-Time FIFO** page of the **Shared Variable Properties** dialog box.

You also cannot remove or rename the shared variable or items related to the variable in the **Project Explorer** window until the VI is no longer reserved to run.

Shared Components of the Application Builder

LabVIEW installs a component for building shared libraries that is shared with all versions of LabVIEW on the local computer. If you install an older version of LabVIEW after installing LabVIEW 8.6 or later, the shared component is replaced by an older version. If you then try to build a shared library in the most current version of LabVIEW, you receive an error because the shared component is missing functionality that LabVIEW 8.6 or later relies on. To correct this issue, reinstall LabVIEW 8.6 or later.

Saving Password-Protected VIs for Previous Versions

In LabVIEW 8.6 and later, if you save a password-protected VI for a previous LabVIEW version, you must enter the password. You also can enter the password programmatically as an input to the Open VI Reference function.

Upgrading from LabVIEW 8.2

You might encounter the following compatibility issues when you upgrade to LabVIEW 2010 from LabVIEW 8.2. Refer to the *Upgrading from LabVIEW 8.5*, *Upgrading from LabVIEW 8.6*, *Upgrading from LabVIEW 2009*, and *Upgrading from LabVIEW 2010* sections of this document for information about other upgrade issues you might encounter.

Platforms Supported

LabVIEW 8.5 and later includes the following changes in platforms supported:

- LabVIEW 8.5 and later supports Windows Vista and Windows Vista 64-bit.
- LabVIEW 8.5.*x* supports Macintosh computers with both Intel and PowerPC processors. LabVIEW 8.6 and later does not support Macintosh computers with PowerPC processors.

Refer to the *LabVIEW Release Notes* for additional system requirements for LabVIEW 2011.

Windows Vista Compatibility Issues

LabVIEW 8.5 and later supports the Windows Vista OS on 32- and 64-bit systems with the following functionality changes.

The In Port and Out Port VIs do not appear on the Functions palette because they allow read/write access to any I/O port on the system, which is discouraged for security reasons on the Vista OS.

- **(Windows Vista)** VI components install properly but show up as unsigned in the Windows Defender log. The VIs do run properly.
- **(Windows Vista 64-bit)** These VIs return error –4850.

VI, Function, and Node Behavior Changes

The behavior of the following VIs, functions, and nodes changed in LabVIEW 8.5 and later.

Enhancements to Analysis VIs and Functions

In each version of LabVIEW, National Instruments enhances many of the algorithms behind LabVIEW and C functions. National Instruments also upgrades LabVIEW to use the latest compilers. These enhancements, along with changes in computer hardware and software, might cause differences in the numerical results between LabVIEW 8.2 or earlier and LabVIEW 8.5 and later. When you compare double-precision, floating-point numbers, you might notice small differences on the order of 1E–16. Refer to the National Instruments Web site at `ni.com/info` and enter the Info Code `exiigr` for more information about comparing floating-point numbers.

Mathematics VIs

LabVIEW 8.5 and later includes changes to the following Mathematics VIs:

- **Find All Zeroes of f(x)**—This VI was renamed the Find All Zeros of f(x) VI.
- **Zeroes and Extrema of f(x)**—This VI was renamed the Zeros and Extrema of f(x) VI.

Numeric Functions

LabVIEW 8.5 and later includes changes to the following Numeric functions:

- **Round To +Infinity**—This function was renamed the Round Toward +Infinity function.

- **Round To -Infinity**—This function was renamed the Round Toward -Infinity function.

Signal Processing VIs

In the Transition Measurements VI, the **preshoot** output changed to **pre-transition**. This output also changed from a 64-bit double-precision, floating-point numeric data type to a cluster data type. The **overshoot** output changed to **post-transition**. This output also changed from a 64-bit double-precision, floating-point numeric data type to a cluster data type.

Hyperbolic Functions

LabVIEW 8.5 and later includes changes to the following hyperbolic functions:

- The Inverse Hyperbolic Cosine function returns `NaN` when the input value is a real number that is out of range for the function.
- The Inverse Hyperbolic Secant function returns `NaN` when the input value is a real number that is out of range for the function.

Libraries & Executables VIs and Functions

In the Call Library Function Node, when configuring a Pascal string pointer, you must wire a value to the string input on the block diagram. When configuring a C string pointer, you must wire a value to the string input or specify the string size in the **Minimum size** pull-down menu on the **Parameters** page of the **Call Library Function** dialog box. You cannot run the VI until you specify values for the strings.

Polymorphic VI Terminals that Support 64-bit and Double-Precision Numeric Data Types

LabVIEW coerces extended-precision numeric data to double-precision numeric data if you wire it to a terminal of a polymorphic VI that supports both the double-precision numeric and 64-bit integer types. This coercion preserves a portion of the fractional component of the original data.

Miscellaneous VI, Function, and Node Behavior Changes

LabVIEW 8.5 and later includes the following miscellaneous VI, function, and node behavior changes:

- The Instr Get Attribute VI and Instr Set Attribute VI no longer ship with LabVIEW. If you use either of these VIs in an application, replace them with the Property Node on the VISA Advanced palette for equivalent functionality.
- The **All Folders** parameter of the Recursive File List VI can contain folder shortcuts, but the VI does not recurse them.
- In LabVIEW 8.2, script nodes, such as the Formula Node and MathScript Node, automatically wrap lines of script that are too long to display completely. In LabVIEW 8.5 and later, script nodes do not wrap lines. Thus, when you open a VI saved in LabVIEW 8.2 or earlier that contains a script node, the script node border might hide parts of long lines of script.

Property, Method, and Event Behavior Changes

The behavior of the following properties, methods, and events changed in LabVIEW 8.5 and later:

- The Data Binding:Path property of the Control class is read/write and settable when the VI is running. To write this property, you must bind the control to an NI Publish-Subscribe-Protocol URL before you begin writing.
- The Target:CPU property of the Application class includes the value `AMD/Intel x64`.
- The Target:Operation System property of the Application class includes the values `Windows x64` and `Linux x64`.
- The Point to Row Column method of the TreeControl class returns the tag `TREE_COLUMN_HEADERS` when you wire a point within the column headers of the tree.
- The LabVIEW Class:Create method includes a name input. If you do not wire the **name** input, LabVIEW prompts the user to name the class at run time.

- The Control Value:Get [Variant], Control Value:Get [Flattened], Control Value:Set [Variant], and Control Value:Set [Flattened] methods no longer trim leading and trailing whitespace when searching for controls. In LabVIEW 8.6, the Control Value:Get [Flattened] and Control Value:Set [Flattened] methods are deprecated. Use the Control Value:Get and Control Value:Set methods instead, respectively.

Deprecated Properties, Methods, and Events

LabVIEW 8.5 and later does not support the following properties, methods, and events:

- Default Instance property of the LVClassLibrary class. Use the Get LV Class Default Value VI instead.
- Geometry property of the SceneObject class. Use the Drawable property instead.
- Grid Colors property of the GraphChart class. Use the Grid Colors property of the GraphScale class instead.
- Grid Colors:X Color property of the GraphChart class. Use the Grid Colors:Major Color and Grid Colors:Minor Color properties instead.
- Grid Colors:X Color property of the GraphChart class. Use the Grid Colors:Major Color and Grid Colors:Minor Color properties instead.
- Grid Colors:Y Color property of the GraphChart class. Use the Grid Colors:Major Color and Grid Colors:Minor Color properties instead.
- Legend:Plots Shown property of the WaveformChart class. Use the Legend:Number of Rows property instead.
- Legend:Plots Shown property of the WaveformGraph class. Use the Legend:Number of Rows property instead.
- Pixel Width property of the ListBox class. Use the Bounds:Area Width property instead.
- Scrollbars Visible property of the Picture class. Use the Horizontal Scrollbar Visible and Vertical Scrollbar Visible properties instead.
- Set Geometry method of the SceneObject class. Use the Set Drawable method instead.
- Scene:Geometry:New Mesh method of the Application class. Use the Scene:Drawable:Geometry:New Mesh method instead.
- Drag Starting event of the Control class. Use the Drag Starting event of the appropriate control class instead.
- Drag Starting? event of the Control class. Use the Drag Starting? event of the appropriate control class instead.

Renamed Properties, Methods, and Events

The following properties, methods, and events are renamed in LabVIEW 8.5 and later.

Class	LabVIEW 8.2 Name	LabVIEW 8.5 and Later Name	Type
AbsTime, Numeric	Data Range	Data Entry Limits	Property
AbsTime, Numeric	Data Range:Increment	Data Entry Limits:Increment	Property
AbsTime, Numeric	Data Range:Maximum	Data Entry Limits:Maximum	Property
AbsTime, Numeric	Data Range:Minimum	Data Entry Limits:Minimum	Property
AbsTime, Numeric	Out of Range Action	Response to Value Outside Limits	Property
AbsTime, Numeric	Out of Range Action:Increment	Response to Value Outside Limits:Increment	Property
AbsTime, Numeric	Out of Range Action:Maximum	Response to Value Outside Limits:Maximum	Property
AbsTime, Numeric	Out of Range Action:Minimum	Response to Value Outside Limits:Minimum	Property

Class	LabVIEW 8.2 Name	LabVIEW 8.5 and Later Name	Type
Application	Library:Get Project Library File Version	Library:Get File LabVIEW Version	Method
Application	Scene:Geometry:New Box	Scene:Drawable:Geometry:New Box	Method
Application	Scene:Geometry:New Cone	Scene:Drawable:Geometry:New Cone	Method
Application	Scene:Geometry:New Cylinder	Scene:Drawable:Geometry:New Cylinder	Method
Application	Scene:Geometry:New Height Field	Scene:Drawable:Geometry:New Height Field	Method
Application	Scene:Geometry:New Mesh	Scene:Drawable:Geometry:New Mesh	Method
Application	Scene:Geometry:New Sphere	Scene:Drawable:Geometry:New Sphere	Method
Application (ActiveX)	LibraryGetProjectLibFileVersion	LibraryGetFileLVVersion	Method
Digital, NumericText, and Scale	Format & Precision	Display Format	Property
Digital, NumericText, and Scale	Format & Precision:Format	Display Format:Format	Property
Digital, NumericText, and Scale	Format & Precision:Precision	Display Format:Precision	Property
DigitalTable	Column Headers Visible	Signal Number Visible	Property
DigitalTable	Row Headers Visible	Transitions Visible	Property
SceneGraphDisplay and SceneWindow	Clear Color	Background Color	Property
SceneObject	Set Geometry	Set Drawable	Method
VI	Connector Pane	Connector Pane:Set	Property

LabVIEW MathScript Behavior Changes (Windows, Not in Base Package)

LabVIEW 8.5 and later includes the following changes to LabVIEW MathScript:

- Changes you make to the search path list or the working directory using the following MathScript functions apply only to the current instance of the **LabVIEW MathScript Window** or the MathScript Node from which you call the function:
  - addpath
  - cd
  - path
  - rmpath

  LabVIEW resets the search path list and the working directory to the default when you close the **LabVIEW MathScript Window** or when the VI that contains the MathScript Node stops running.
- The syntax for the qz function changed from `[q, z, alpha, beta, evec] = qz(a, b)` to `[S, T, Q, Z, R, L] = qz(A, B, type)`.

LabVIEW Class Icons

If you created a LabVIEW class icon in LabVIEW 8.2 and you want the icon displayed when you add a class control or indicator to the block diagram, you must update the class icon to occupy a smaller

space so that the class mask does not obscure any part of the class icon. Use an image no larger than 32 pixels wide by 19 pixels high.

Opening LLBs in LabVIEW

The **Enable Windows Explorer for LLB files** option on the **Environment** page of the **Options** dialog box no longer exists. LabVIEW opens LLBs in the **LLB Manager** window. Refer to the National Instruments Web site at ni.com/info and enter the Info Code exvfc5 for more information about opening LLBs.

Timed Loop Priority Level Restriction

In LabVIEW 8.2.*x* and earlier, you can select up to 2 to the power of 32 for the priority level of a Timed Loop. LabVIEW 8.5 and later supports only priority levels less than 65,535.

Waveform Data Type

When indexing beyond the bounds of an array of waveforms, the resulting waveform is a proper default waveform with the dt value equal to 1, instead of an improper waveform with the dt value equal to 0. This also is true when executing a For Loop with a scalar output tunnel zero times.

Enum Coercion

LabVIEW 8.5 and later coerces out-of-range enums to the last value that fits into the range of the enum. Previous LabVIEW versions coerce out-of-range enums to 0.

Upgrading from LabVIEW 8.0 or Earlier

Refer to the National Instruments Web site at ni.com/info and enter the Info Code exc6mf to access upgrade and compatibility issues you might encounter when upgrading from LabVIEW 8.0 or earlier to the current version. Also, refer to the other *Upgrading from LabVIEW x* sections in this document for information about other upgrade issues you might encounter.

LabVIEW 2011 Features and Changes

The Idea Exchange icon denotes a new feature idea that originates from a product feedback suggestion on the LabVIEW Idea Exchange discussion forums. Refer to the National Instruments Web site at ni.com/info and enter the Info Code ex3gus to access the NI Idea Exchange discussion forums.

Refer to the readme.html file in the labview directory for known issues, a partial list of bugs fixed, additional compatibility issues, and information about late-addition features in LabVIEW 2011.

New Example VIs

Refer to the **New Examples for LabVIEW 2011** folder on the **Browse** tab of the NI Example Finder to view descriptions for and launch example VIs added to the current version of LabVIEW.

Block Diagram Enhancements

LabVIEW 2011 includes the following enhancements to the block diagram and related functionality.

Enhancements to Creating SubVIs

In previous versions of LabVIEW, when you create a subVI from a block diagram selection, you must clean up the connector pane and front panel of the new subVI manually. In LabVIEW 2011, LabVIEW automatically builds the connector pane and front panel of the subVI according to the following LabVIEW programming conventions:

- A 4×2×2×4 connector pane pattern, unless the subVI requires more terminals
- **error in** and **error out** terminals in the lower corners of the connector pane
- Refnum or class terminals in the upper corners of the connector pane

- Controls aligned on the left side of the front panel
- Indicators aligned on the right side of the front panel

For example, if you select the highlighted portion of the following block diagram, LabVIEW generates the following front panel and connector pane.

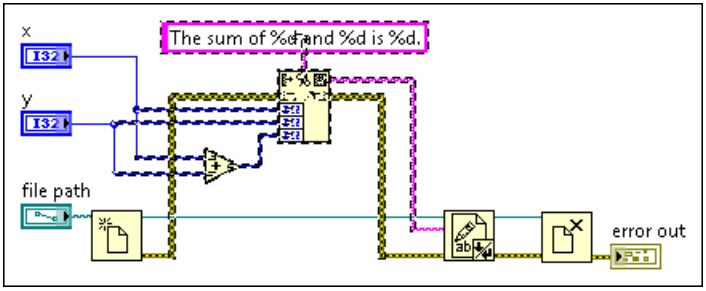

Figure 1. Block diagram code from which to create subVI

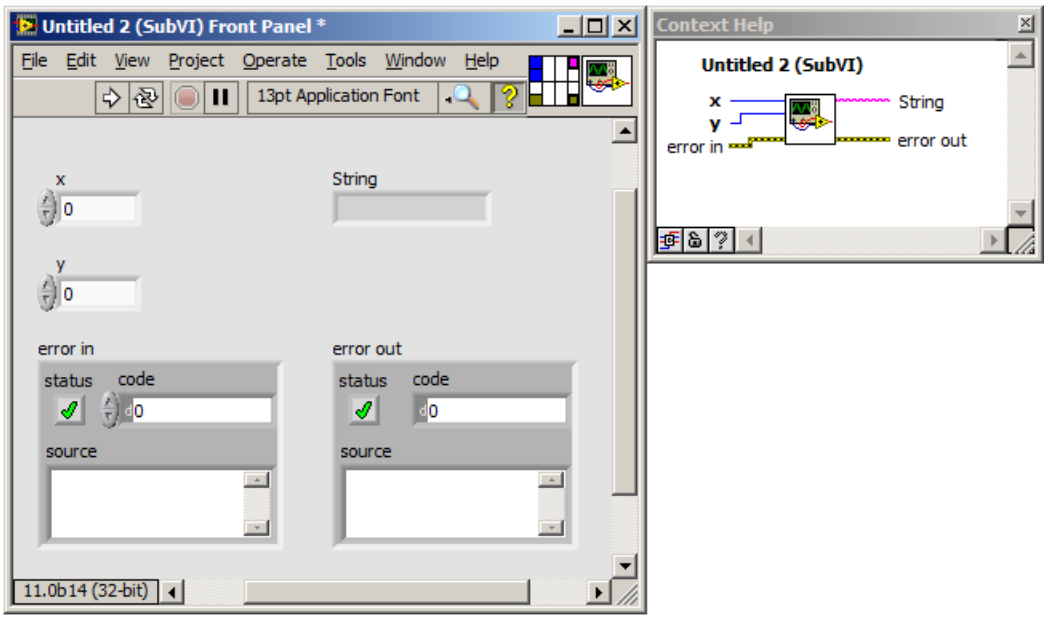

Figure 2. Front panel and connector pane of resulting subVI

[*Idea submitted by NI Discussion Forums member tst.*]

Type Definition Enhancements

LabVIEW 2011 includes the following enhancements to type definitions and strict type definitions:

- In LabVIEW 2010 and earlier, you can create a type definition only from the front panel. In LabVIEW 2011, you can create a type definition from the front panel *or* block diagram. Right-click

the constant, control, or indicator you want to make a type definition and select **Make Type Def.** from the shortcut menu. [*Idea submitted by NI Discussion Forums member crelf.*]

- A glyph, shown as follows, identifies instances of type definitions and strict type definitions on the block diagram:  . If you move the cursor over the glyph, a tip strip displays the filename of the type definition or strict type definition. [*Idea submitted by NI Discussion Forums member Broken Arrow.*]

Miscellaneous Block Diagram Enhancements

LabVIEW 2011 includes the following miscellaneous block diagram enhancements:

- You can use the **Align Objects** and **Distribute Objects** pull-down menus to align and distribute wires on the block diagram. [*Idea submitted by NI Discussion Forums member JackDunaway.*]
- When you define a new event case for a control or indicator, Value Change is the default event. [*Idea submitted by NI Discussion Forums member Bruce Ammons.*]
- You can wire an error cluster directly to the following Boolean functions to handle errors using logical operations: And, Or, Exclusive Or, Implies, Not, Not And, Not Exclusive Or, and Not Or. If an error occurs, the error cluster passes a TRUE value to the Boolean function. [*Idea submitted by NI Discussion Forums member Dany Allard.*]
- The parallel instances terminal of a For Loop with parallel iterations returns one of two types of output values. Right-click the parallel instances terminal  of a For Loop with parallel iterations and select **P Terminal Output** from the shortcut menu to select which of the following values the terminal returns:
  - **Number of Instances**—Returns the number of loop instances LabVIEW prepared for parallel execution.
  - **Current Instance ID**—Returns an identifier assigned to the loop instance currently executing.

  Refer to the **Fundamentals»Loops and Other Structures»Controlling Repetitive Operations Using Loops»For Loops: Repeating Operations a Set Number of Times** book on the **Contents** tab of the *LabVIEW Help* for more information about the parallel instances terminal.

Front Panel Enhancements

LabVIEW 2011 includes the following enhancements to the front panel and related functionality.

New Style of Front Panel Controls and Indicators

The Silver palette contains the new silver style of controls and indicators you can use to build the front panel. To access the silver style of controls and indicators, browse to the **Controls»Silver** palette. Use the silver controls and indicators to build a front panel with an updated appearance and feel. The Silver palette contains more Boolean controls and indicators than other collections of controls and indicators.

Choosing Which Plots to View on a Graph or Chart

To choose whether to display a plot on a graph or chart, right-click the glyph in the plot legend and select **Plot Visible** from the shortcut menu. If the graph or chart contains multiple plots, you can right-click the plot legend and select **Visible Items»Plot Visibility Checkbox** from the shortcut menu to display a checkbox next to each plot. Use the checkboxes to select which plots you want to display. You also can use the Plot Visibility Checkbox Visible property to display the checkboxes programmatically. [*Idea submitted by NI Discussion Forums member Sil3nc3r.*]

Miscellaneous Front Panel Enhancements

LabVIEW 2011 includes the following miscellaneous front panel enhancements:

- Every LabVIEW VI displays the connector pane next to the VI icon in the upper right corner of the front panel window. You can assign front panel controls and indicators as inputs and outputs

without switching back and forth between the VI icon and the connector pane. [*Idea submitted by NI Discussion Forums member blawson.*]

- The background color of the front panel is a lighter shade of gray than the color in previous versions of LabVIEW to complement the appearance of the controls and indicators on both the Modern and Silver palettes. Refer to the National Instruments Web site at `ni.com/info` and enter the Info Code `exd9s8` for more information about changing the background color of the front panel to match VIs saved in previous versions of LabVIEW.

Environment Enhancements

LabVIEW 2011 includes the following enhancements to the LabVIEW environment.

Dialog Box Enhancements

LabVIEW 2011 includes the following dialog box enhancements.

Clear Compiled Object Cache Dialog Box Enhancements

Due to optimizations to the VI object cache in LabVIEW 2011, you no longer can clear the VI object cache associated with a specific target. Instead, the **Clear Compiled Object Cache** dialog box allows you to clear only the following two caches:

- **User object cache**—Contains the compiled VI object files for all VIs that did not ship with LabVIEW. You might want to clear the **User** object cache for the following reasons:
  - You run out of disk space.
  - You delete a large number of VIs with separate compiled code and want to reclaim the disk space occupied by their compiled code.
- **Application Builder object cache**—Contains the compiled code for VIs in build specifications.

Miscellaneous Dialog Box Enhancements

LabVIEW 2011 includes the following miscellaneous dialog box enhancements:

- The new **Size** page of the **Properties** dialog box for arrays configures the number of dimensions in an array.
- The **Appearance** page of the **Properties** dialog box for strings includes the **Display Style Visible** checkbox, which allows you to display a glyph on a string control or indicator that indicates its display type.
- The **Plots** page of the **Properties** dialog box for 2D Error Bar graphs includes the **Error Mode** list, which sets whether the error bar displays on the y-axis or on the x- and y-axes.
- To customize the **Icon Editor** dialog box for LabVIEW 2011, or if you have a custom Icon Editor from a previous version of LabVIEW, you can refer to the National Instruments Web site at `ni.com/info` and enter the Info Code `LViconed11` to download the Icon Editor source files.
- Certain LabVIEW add-ons, such as the LabVIEW Robotics Module, include additional environment settings that allow you to focus on a particular subject area. If you install one or more of these LabVIEW add-ons, you can use the **Choose Environment Settings** dialog box to change from one environment to another without restarting LabVIEW.
- **(Windows)** The **Getting Started** window contains a new **Find LabVIEW Add-ons** item, which launches the VI Package Manager (VIPM) software so you can access LabVIEW add-ons and other code distributed on the LabVIEW Tools Network.

Miscellaneous Environment Enhancements

LabVIEW 2011 includes the following miscellaneous environment enhancements:

- You can perform an undo operation after you save a VI. [*Idea submitted by NI Discussion Forums member mishklyar.*]

- As you load a VI, you can select **Ignore All** to ignore loading all missing subVIs. [*Idea submitted by NI Discussion Forums member Ravens Fan.*]
- **(Windows)** The **Help** menu includes the **Check for Updates** item, which opens the **NI Update Service** window to check `ni.com` for any available updates.
- The VI toolbar buttons have an updated appearance to complement the background color of the front panel.
- You can configure LabVIEW to automatically separate compiled code from all new VIs rather than manually selecting this option for each VI that you create. To configure LabVIEW to separate compiled code from all new VIs, place a checkmark in the **Separate compiled code from source file for new VIs** checkbox on the **Environment** page of the **Options** dialog box. You can still override this global setting for individual VIs and projects.

  Refer to the **Fundamentals»Creating VIs and SubVIs»Concepts»Separating Compiled Code from VIs** topic on the **Contents** tab of the *LabVIEW Help* for more information about the benefits of separating compiled code from VIs.

Application Builder Enhancements

To delete the build specification files that LabVIEW creates during the most recent build, right-click the build specification in the **Project Explorer** window and select **Clean** from the shortcut menu. Selecting **Clean** does not delete files that LabVIEW creates before the most recent build.

New and Changed VIs, Functions, and Nodes

LabVIEW 2011 includes the following new and changed VIs, functions, and nodes. Refer to the **VI and Function Reference** book on the **Contents** tab of the *LabVIEW Help* for more information about VIs, functions, and nodes.

New VIs, Functions, and Nodes

LabVIEW 2011 includes the following new VIs, functions, and nodes.

Application Builder VIs

The Application Control palette includes the new Application Builder palette with the following new VIs:

- Build
- Clean
- Deploy

Application Control VIs and Functions

The Application Control palette includes the following new nodes:

- Start Asynchronous Call
- Wait On Asynchronous Call

Bessel Function VIs

The Bessel Functions palette includes the following new VIs:

- Hankel Function Hv
- Spherical Hankel Function hn

Error Function VIs

The Error Functions palette includes the following new VIs:

- Inverse Error Function
- Inverse Error Function Complement

- Scaled Error Function Complement

Geometry VIs

The Geometry palette includes the new Angle palette with the following new VIs:

- Absolute Angle Difference
- Angle Rotation
- Bisect Angle
- Check for Included Angle
- Complementary Angle
- Supplementary Angle
- Wrap Angle

The Geometry palette also includes the new Computational Geometry palette with the following new VIs:

- Contour Line
- Convex Hull
- Delaunay Triangulation
- Point in Polygon
- Polygon Area
- Voronoi Diagram

Linear Algebra VIs

The Linear Algebra palette includes the following new VIs:

- Cholesky Factorization Rank-1 Update
- Subspaces Angle

Scaling VIs

The Scaling palette includes the following new VIs:

- Degrees to Radians
- Radians per Second to RPM
- Radians to Degrees
- RPM to Radians per Second

Miscellaneous New VIs and Functions

LabVIEW 2011 includes the following miscellaneous new VIs and functions:

- The Dialog & User Interface palette includes the error cluster constant. [*Idea submitted by NI Discussion Forums member Broken Arrow.*]
- The Discrete Math palette includes the Prime Factor VI.
- The HTTP Client palette includes the ConfigSSL VI.
- The Interpolation & Extrapolation palette includes the Interpolate 2D Scattered VI.
- The Optimization palette includes the Global Optimization VI.
- The Orthogonal & Non-orthogonal Polynomials palette includes the Legendre Polynomial VI.
- The Packed Library palette includes the Get Exported File Path VI.
- The Probability & Statistics palette includes the Skewness and Kurtosis VI.
- The Signal Generation palette includes the Quasi Random VI.
- The Signal Operation palette includes the Digital Reversed Order VI.

Changed VIs and Functions

The following VIs and functions changed in LabVIEW 2011.

Geometry VIs

The following VIs on the Geometry palette include a **rotation order** input, which specifies the order of the axes around which you want to rotate the coordinates:

- 3D Cartesian Coordinate Rotation (Euler)
- Direction Cosines To Euler Angles
- Euler Angles To Direction Cosines

Miscellaneous VI and Function Changes

LabVIEW 2011 includes the following miscellaneous VI and function changes:

- **Averaged DC-RMS**—The **Ignore input time stamp** element of the **averaging control** cluster is renamed **Ignore input start time**.
- **Create Special Matrix**—The data type of the **matrix type** input changed from an enum to a 32-bit signed integer. This input also includes the following new values you can use to create a special matrix: `Hadamard`, `Hankel`, `Hilbert`, `Inverse Hilbert`, `Pascal`, `Rosser`, and `Wilkinson`.
- **General Linear Fit**—Includes the new **Weight Out** output, which returns the actual weight of general linear fitting or returns the value you enter for **Weight**, depending on the method you specify.
- **TDMS Read**—Includes the new **return channels in file order?** input, which specifies whether the function returns data channels in the same order as they exist in a `.tdms` file. If the value of this input is FALSE, this function returns data channels in the same order as you specify in the **channel name(s) in** input. The default is FALSE.
- **Variant To Data**—The error cluster terminals line up evenly with the majority of the error terminals of other functions and VIs. [*Idea submitted by NI Discussion Forums member David_L.*]
- **Zero Phase Filter**—Includes the following new instances: Zero Phase Filter (Cascade, DBL) and Zero Phase Filter (Cascade, CDB).
- The CIN functions are renamed the LabVIEW Manager functions.

New and Changed Classes, Properties, Methods, and Events

LabVIEW 2011 includes the following new and changed classes, properties, methods, and events.

Math Plots Properties and Methods

LabVIEW 2011 includes the new Plot Specific:Error Bar Mode property of the 2D Error Bar Properties class.

VI Server Properties and Methods

LabVIEW 2011 includes new VI Server classes, properties, methods, and events. Refer to the **LabVIEW 2011 Features and Changes»New VI Server Objects** topic on the **Contents** tab of the *LabVIEW Help* for a list of new classes, properties, methods, and events.

LabVIEW 2011 includes the following VI Server method changes:

- The following methods of the Application class support `.dae` files:
  - Scene:Read Scene File
  - Scene:Write Scene File
- For the Create from Data Type method of the VI class, the data type of the **Style** parameter changed from a 32-bit signed integer to an enum. The **Style** parameter also includes the following new values: `Probe`, `Diagram`, `Panel Control`, `3D Panel Control`, `Dialog Panel Control`,

`Panel Indicator`, `3D Panel Indicator`, `Dialog Panel Indicator`, `Power PC Control`, and `Power PC Indicator`.

Asynchronously Calling VIs by Reference

To allow data flow to continue in a VI without waiting for a target VI to finish executing, you must call the target VI asynchronously. Use the Start Asynchronous Call node to pass input parameters to and start execution of a target VI. Use the Wait On Asynchronous Call node to collect the outputs of the target VI in the calling VI at a later time.

Consider calling a target VI asynchronously for the following reasons:

- When the target VI might require a long time to execute and the calling VI does not immediately require the outputs of the target VI.
- To execute any number of reentrant instances of a VI in parallel.
- To implement a simpler interface and experience better performance than the Run VI method provides.

Refer to the **Fundamentals»Programmatically Controlling VIs»Concepts»Asynchronously Calling VIs** topic on the **Contents** tab of the *LabVIEW Help* for more information about calling VIs asynchronously.

[*Idea submitted by NI Discussion Forums member Jim Kring.*]

New Math and Signal Processing VIs

LabVIEW 2011 includes several new VIs for performing mathematics and signal processing operations. LabVIEW also contains several VIs with new or changed inputs and outputs. These new and changed VIs provide new built-in algorithms in several areas, including the following areas:

- Geometry
- Linear algebra
- Signal processing

Refer to the *New VIs, Functions, and Nodes* section of this document for a complete list of new VIs in LabVIEW 2011. Refer to the *Changed VIs and Functions* section for a complete list of changes to VIs.

Enhancements to .NET Support

LabVIEW 2011 includes the following enhancements to .NET support.

Debugging Unexpected Behaviors of .NET Assemblies in LabVIEW

.NET objects can exhibit unexpected behavior when LabVIEW has a different version of the assembly in memory than the version you want to use. To identify this discrepancy and debug the assembly calls, use the **.NET Assemblies in Memory** dialog box. If you determine that LabVIEW loaded the wrong version of an assembly, you can use this dialog box to adjust assembly locations on disk, create a configuration file, or reload updated assemblies to ensure that LabVIEW loads the correct assembly.

Refer to the **Fundamentals»Windows Connectivity»How-To».NET»Debugging Calls to .NET Assemblies** book on the **Contents** tab in the *LabVIEW Help* for more information about using the **.NET Assemblies in Memory** dialog box to debug calls to .NET assemblies.

Loading .NET 4.0 Assemblies in LabVIEW

The most recent programming environments for .NET assemblies target new assemblies to run in the Common Language Runtime (CLR) 4.0 by default. Although LabVIEW is most compatible with the CLR 2.0, you can force LabVIEW to use the CLR 4.0 to load .NET 4.0 assemblies.

Refer to the **Fundamentals»Windows Connectivity»How-To».NET»Loading .NET 4.0 Assemblies in LabVIEW** book on the **Contents** tab of the *LabVIEW Help* for more information about this feature, including the specific contents of the required configuration file.

Configuring I/O Variables Remotely

In previous versions of LabVIEW, you can configure an I/O variable only from the target that hosts the variable. In LabVIEW 2011, you can configure I/O variables from remote computers.

Refer to the **Fundamentals»Accessing Scanned I/O Data»Concepts»Using I/O Variables** topic on the **Contents** tab of the *LabVIEW Help* for more information about configuring I/O variables.

Verifying That Hardware Supports Compiler Optimizations

LabVIEW 2011 incorporates compiler optimizations that improve the run-time performance of VIs and applications on processors that support SSE2 instructions. When you compile VIs in the LabVIEW development environment, LabVIEW automatically detects whether your processor supports SSE2 instructions and enables or disables the compiler optimizations accordingly. However, when you build an application in LabVIEW, you must verify that all potential targets support SSE2 instructions. If any potential targets do not support SSE2 instructions, you must disable the compiler optimizations.

Refer to the **Fundamentals»Building and Distributing Applications»Configuring Build Specifications»Verifying that Target Hardware Supports SSE2 Instructions** topic on the **Contents** tab of the *LabVIEW Help* for more information about which processors and hardware types support SSE2 instructions.

Viewing and Controlling Front Panels Remotely without a License

In previous versions of LabVIEW, you need a license to view and control front panels from remote computers, and the license grants only a limited number of remote connections to front panels. In LabVIEW 2011, you can view and control front panels remotely without a license, and you can connect to remote front panels an unlimited number of times.

Refer to the **Fundamentals»Transferring Data over a Network»Viewing and Controlling Front Panels Remotely with the Web Server** book on the **Contents** tab of the *LabVIEW Help* for more information about viewing and controlling front panels remotely.

Improved Notification When the LabVIEW Run-Time Engine Is Missing

When you run a LabVIEW stand-alone application on a computer that does not have the LabVIEW Run-Time Engine installed, the application displays an error message that includes a link to the National Instruments Web site. This link allows users of the stand-alone application to download the LabVIEW Run-Time Engine without contacting the application vendor to troubleshoot the issue.

371780G-01 Jun11