LabVIEW™ Upgrade Notes

Version 8.2

These upgrade notes describe the process of upgrading LabVIEW for Windows, Mac OS, and Linux to version 8.2, issues you might encounter when you upgrade, and new features.

If you are upgrading from LabVIEW 7.1 or earlier to LabVIEW 8.2, refer to the *LabVIEW 8.0 Upgrade Notes* for information about the enhancements, changes, and added features in LabVIEW 8.0. National Instruments recommends that all users upgrading from LabVIEW 7.1 or earlier read the *LabVIEW 8.0 Features and Changes* section of the *LabVIEW 8.0 Upgrade Notes* in addition to these upgrade notes. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `upnote8` to access the *LabVIEW 8.0 Upgrade Notes*.

Refer to the *LabVIEW Help* for more information about LabVIEW 8.2 features, as well as for information about LabVIEW programming concepts, step-by-step instructions for using LabVIEW, and reference information about LabVIEW VIs, functions, palettes, menus, tools, properties, methods, events, dialog boxes, and so on. The *LabVIEW Help* also lists the LabVIEW documentation resources available from National Instruments. Access the *LabVIEW Help* by selecting **Help»Search the LabVIEW Help**.

Contents

NATIONAL INSTRUMENTS™

Upgrading to LabVIEW 8.2

If you are upgrading from a previous version of LabVIEW, read this section, *Upgrading to LabVIEW 8.2,* and the *Upgrading from LabVIEW x.x* sections in the *Upgrade and Compatibility Issues* section of this document first, where *x.x* is the version of LabVIEW from which you are upgrading.

Converting VIs

When you open a VI last saved in LabVIEW 4.0 or later, LabVIEW 8.2 automatically converts and compiles the VI. You must save the VI in LabVIEW 8.2, 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. Refer to the *LabVIEW Help* for more information about this performance and memory issue.

Note VIs you save in LabVIEW 8.2 do not load in earlier versions of LabVIEW. Select **File»Save for Previous Version** to save VIs so they can run in LabVIEW 8.0. Before saving VIs in LabVIEW 8.2, keep a backup copy of VIs you plan to use in LabVIEW 8.0 or earlier.

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 alphabetical order. 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 Toolkits, Instrument Drivers, and Add-Ons

After you install LabVIEW 8.2, make sure you have a compatible version of any toolkits and add-ons, then reinstall the toolkits and add-ons in the LabVIEW 8.2 directory. You first might need to uninstall the toolkit from previous versions of LabVIEW. Refer to the documentation for the LabVIEW toolkit or add-on for more information about installation.

Note LabVIEW module versions must match the LabVIEW version. For example, you must use the LabVIEW Real-Time Module 8.2 with LabVIEW 8.2.

You also must mass compile existing toolkit, instrument driver, and add-on VIs for use in LabVIEW 8.2. Refer to the *Converting VIs* section of this document for more information about mass compiling VIs.

The following toolkits, instrument drivers, and add-ons require upgrades or downloads for use in LabVIEW 8.2:

- If you have the LabVIEW Application Builder, you must upgrade to LabVIEW Application Builder 8.2. The LabVIEW 8.2 Professional Development System includes Application Builder 8.2. Refer to the *LabVIEW Application Builder Readme* located in the **(Windows)** `labview\readme` directory or **(Mac OS and Linux)** `labview` directory for more information about installing the LabVIEW Application Builder.
- You must use VI Analyzer 1.1 in LabVIEW 8.*x*. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `exd8yy` to access the Upgrade Advisor and purchase VI Analyzer 1.1.
- You must download additional VIs to use the Internet Toolkit 6.0 with LabVIEW 8.*x*. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `itkver6` to download the necessary VIs.

- The instrument driver for the HP/Agilent 34401A Digital Multimeter (DMM) now more closely resembles the National Instruments DMM template driver. This driver is not compatible with the HP34401A driver that LabVIEW 7.*x* and earlier use. If you need compatibility with the LabVIEW 7.*x* HP34401A driver, download that driver from the National Instruments Instrument Driver Network at `ni.com/idnet`.

Upgrading Additional National Instruments Software

You must use NI TestStand 3.5 or later in LabVIEW 8.*x*. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `exd8yy` to access the Upgrade Advisor and purchase NI TestStand 3.5 or later.

Upgrading from Previous Versions of LabVIEW

Upgrading to new versions of LabVIEW does not affect previous versions of LabVIEW on the computer because the new versions install in a different directory. LabVIEW 5.*x* and earlier install in the `labview` directory. LabVIEW 6.0 and later install in the `labview x.x` directory, where *x.x* is the version number. You can install LabVIEW 8.2 without uninstalling previous versions of LabVIEW.

Replacing an Existing Version of LabVIEW

To replace your existing version of LabVIEW, uninstall the existing version of LabVIEW, run the LabVIEW 8.2 installer, and set the installation directory to the same `labview` directory where you installed the previous version of LabVIEW.

(Windows) You also can replace the existing version of LabVIEW with LabVIEW 8.2 by using the Add/Remove Programs applet in the Control Panel to uninstall the existing version of LabVIEW. The uninstaller does not remove any files you created in the `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 f you replace the LabVIEW 8.2 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 8.2, copy the LabVIEW preferences file to the LabVIEW 8.2 directory.

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

(Mac OS) LabVIEW stores preferences in the `LabVIEW Preferences` file in the `Library:Preferences` folder in your home directory.

(Linux) LabVIEW stores preferences in the `.labviewrc` file in your home directory.

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 8.2, copy the files to the `user.lib` directory in the LabVIEW 8.2 directory.

Upgrade and Compatibility Issues

Refer to the following sections for upgrade and compatibility issues specific to different versions of LabVIEW.

Refer to the `readme.html` file in the `labview` directory for known issues, additional compatibility issues, and information about late addition features in LabVIEW 8.2.

Upgrading from LabVIEW 8.0

You might encounter the following compatibility issues when you upgrade to LabVIEW 8.2 from LabVIEW 8.0.

Platforms Supported

LabVIEW 8.2 includes the following changes in platforms supported:

- LabVIEW 8.2 does not support Windows XP x64.
- LabVIEW 8.2 does not support Mac OS X 10.3.8 or earlier.
- LabVIEW 8.2 provides some support for Macintosh computers with Intel processors. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `macintel` for more information about Macintosh support.

System Requirements

(Mac OS) LabVIEW 8.2 requires at least 500 MB of disk space for the minimum LabVIEW installation or 700 MB disk space for the complete LabVIEW installation.

(Linux) LabVIEW 8.2 requires at least 430 MB of disk space for the minimum LabVIEW installation or 620 MB disk space for the complete LabVIEW installation.

Printed Documentation

The following documents did not change for LabVIEW 8.2. Therefore, the content might not reflect changes made in LabVIEW 8.2.

- *LabVIEW Quick Reference Card*
- *LabVIEW Fundamentals Manual*—Because the *LabVIEW Fundamentals Manual* is a subset of the **Fundamentals** book in the *LabVIEW Help*, refer to the **Fundamentals** book on the **Contents** tab of the *LabVIEW Help* for updated content.

VI and Function Behavior Changes

The behavior of the following VIs and functions changed in LabVIEW 8.2.

Communicating between Application Instances

In LabVIEW 8.2, you cannot use the Obtain Queue, Obtain Notifier, Create User Event, Create Semaphore, and Create Rendezvous functions to communicate between LabVIEW application instances. If you obtain or create a queue, notifier, user event, semaphore, or rendezvous reference in one application instance, you cannot use that reference in another application instance.

Back Transform Eigenvectors VI

The **job**, **index low**, **index high**, and **Scale** inputs of the Back Transform Eigenvectors VI are required inputs.

DataSocket Write Function

In LabVIEW 8.0.1, the default behavior for DataSocket Write function changed to asynchronous. If you have LabVIEW 8.0 and LabVIEW 8.2 installed on your computer, the DataSocket API Client VI example in the `labview\examples\Shared Variable` directory returns an error when you stop the VI. You must update LabVIEW 8.0 to LabVIEW 8.0.1 to use this example in LabVIEW 8.2.

File I/O VIs

The Write To Spreadsheet File VI and Read From Spreadsheet File VI are polymorphic VIs. The Write to Spreadsheet File VI adapts to the value you wire to the **format** input. The Read From Spreadsheet File VI includes the following instances: DBL, I64, and string.

GPIB Status Function

In LabVIEW 8.0, the GPIB Status function did not execute if the **error in** input received an error. In LabVIEW 8.2, the GPIB Status function always executes, even if the **error in** input receives an error.

Histogram VI

The default for the **intervals** input of the Histogram VI changed to `10`.

Open VI Reference Function

The default behavior for the **options** input of the Open VI Reference function is to prompt users to find missing subVIs of the referenced VI. A new value, `0x20`, specifies not to display the **Find** dialog box or prompt users to find missing subVIs of the referenced VI.

Polynomial Roots VI

If **P(x)** equals a nonzero constant, the Polynomial Roots VI does not return an error. However, if **P(x)** equals `0`, the Polynomial Roots VI returns error `-20111`. The input polynomial coefficients for this VI cannot all be zeros.

Ramp Pattern VI

In the Ramp Pattern VI, if **samples** is `1` and **exclude end?** is TRUE, the VI returns an array with one element of **start**, with no error. In LabVIEW 8.0, the VI returned an error with these conditions.

Read Registry Value Simple VI

LabVIEW 8.0 incorrectly handled `REG_MULTI_SZ` string formatting, which the VI used for a flattened array of strings. This issue required you to write a parser to handle this type of data for the Read Registry Value Simple VI. In LabVIEW 8.2, the Read Registry Value Simple VI returns this type of data in the same format used in the Write Registry Value Simple VI. You no longer need to add your own parser. Using your own parser with these VIs in LabVIEW 8.2 causes the Read Registry Value Simple VI to return bad data.

Resample Waveforms (single-shot) VI

The default value of the **open interval?** input of the Resample Waveforms (single shot) VI changed from TRUE to FALSE, which selects a closed interval. If you do not update existing code accordingly, the VI might not return the expected result.

Sound VIs

In the Sound Input Read and the Sound File Read Simple VIs, the **t0** component of the **data** output returns the time stamp for the first sample read. LabVIEW approximates the initial time that it reads the first sample.

Calling The Sound Output Stop VI no longer is necessary to stop the sound on a continuous sound task.

The Sound Output Wait VI works in **Continuous Samples** mode and in **Finite Samples** mode.

Waveform VIs

LabVIEW 8.2 includes changes to the following Waveform VIs:

- Basic Level Trigger Detection VI—In both instances of this VI, the **slope** input changed to **trigger slope**.
- Get Waveform Subset VI—Includes the following instances: WDT Get Waveform Subset DBL, WDT Get Waveform Subset CDB, WDT Get Waveform Subset EXT, WDT Get Waveform Subset I16, WDT Get Waveform Subset I32, WDT Get Waveform Subset I8, and WDT Get Waveform Subset SGL. The **start/duration format** input no longer includes an **Absolute Time** option. The **start** input changed to **start samples/time**, and the **actual start** output changed to **actual start samples/time**.
- Get Waveform Time Array VI—The **X array** output changed from a double-precision, floating-point numeric data type to a time stamp data type.
- Get Y Value VI—This VI and the corresponding polymorphic instances were renamed to Get XY Value. The Get XY Value VI now includes an **X value** output, and the **data value** output changed to **Y value**.
- Number of Waveform Samples VI—This VI is a polymorphic VI with the following instances: WDT Number of Waveform Samples DBL, WDT Number of Waveform Samples CDB, WDT Number of Waveform Samples EXT, WDT Number of Waveform Samples I16, WDT Number of Waveform Samples I32, WDT Number of Waveform Samples I8, and WDT Number of Waveform Samples SGL.

- Read Waveform from File VI—Returns an error status of TRUE in the **error out** output when the error is end-of-file.
- Replace Subset VI—The **start** input changed to **start samples/time**, and the **actual start value** output changed to **actual start samples/time**.
- Search for Digital Pattern VI—The **start** input changed to **start index/time**.
- Search Waveform VI—The **time of best fit** and **time of fits** outputs changed from a double-precision, floating-point numeric data type to a time stamp data type.
- Waveform Min Max VI—The **min time** and **max time** outputs changed from a double-precision, floating-point numeric data type to a time stamp data type.
- Waveform to XY Pairs VI—The **x** element of the **XY pairs** output changed from a double-precision, floating-point numeric data type to a time stamp data type.

Property, Method, and Event Behavior Changes

The behavior of the following properties, methods, and events changed in LabVIEW 8.2:

- The default behavior for the **options** input of the ActiveX GetVIReference method is to prompt users to find missing subVIs of the referenced VI. A new value, `0x20`, specifies not to display the **Find** dialog box or prompt users to find missing subVIs of the referenced VI.
- The Add Item method of the ProjectItem class return an error when you try to add a shared variable to a library that is not opened in a project.
- If the **Auto Dispose Ref** input of the Run VI method is TRUE and the method returns an error, LabVIEW does not dispose of the reference.
- Valid values for the Application:Language property include `zh-cn` to indicate that Simplified Chinese is the language of the LabVIEW environment.
- In LabVIEW 8.0, .NET methods that pass array data types by reference pass all data as the refnum data type. .NET methods that pass array data types by reference, passes the data as the actual data type.
- The Edit Position property of the DigitalTable, MulticolumnListbox, Table, and TreeControl classes returns values of (–2, –2) to indicate that the user is not making edits to the text of the control. The Edit Row property of the ListBox class returns a value of –2 to indicate that the user is not making edits to the text of the control.

- In LabVIEW 8.0, the Defer Panel Updates property did not defer the update of front panels in a subpanel. In LabVIEW 8.2, the Defer Panel Updates property works with subpanels.
- The Application Instance Close and Application Instance Close? events replace the Application Exit and Application Exit? events. When you use the Application Instance Close event in a VI running outside a LabVIEW project, LabVIEW generates the event when you quit LabVIEW through the user interface or programmatically. LabVIEW generates the Application Instance Close? event when you quit LabVIEW through the user interface. When you register the Application Instance Close and Application Instance Close? events for a VI running within a LabVIEW project, LabVIEW generates the events when the application instance closes or when you quit LabVIEW.

Deprecated Properties, Methods, and Events

LabVIEW 8.2 does not support the following properties, methods, and events.

- LabVIEW 8.2 does not support the Connector Pane property.
- LabVIEW 8.*x* does not support the Data Type property in the Variable class. Use the Data Type (Variant) property in the Variable class instead.

Renamed Properties, Methods, and Events

The following properties, methods, and events are renamed in LabVIEW 8.2:

Class	LabVIEW 8.0 Name	LabVIEW 8.2 Name	Type
Application	Disconnect From Slave	LVRT:Disconnect From Slave	Method
Application	Application Exit	Application Instance Close	Event
Application	Application Exit?	Application Instance Close?	Event
IntensityGraph, MixedSignalGraph, and WaveformGraph	Cursor Palette Visible	Cursor Legend Visible	Property
Library	Delete Library Tag	Library Tag:Delete	Method
Library	Get Icon	Icon:Get	Method
Library	Get Library Tag	Library Tag:Get	Method

Class	LabVIEW 8.0 Name	LabVIEW 8.2 Name	Type
Library	Get Library Tag Names	Library Tag:Get Names	Method
Library	Get Lock State	Lock State:Get	Method
Library	Get Source Scope	Source Scope:Get	Method
Library	Save	Save:Library	Method
Library	Save a Copy	Save:Copy	Method
Library	Set Icon	Icon:Set	Method
Library	Set Library Tag	Library Tag:Set	Method
Library	Set Lock State	Lock State:Set	Method
Library	Set Source Scope	Source Scope:Set	Method
Listbox, MulticolumnListbox, and TreeControl	Drag/Drop: Allow Item Dragging	Drag/Drop:Allow Dragging	Property
Path and String	Allow Drop	Allow Dropping	Property
ProjectItems	Delete Tag	Tag:Delete	Property
ProjectItems	Get Tag	Tag:Get Tag	Property
ProjectItems	Get Tag Names	Tag:Get Names	Property
ProjectItems	Get XML Tag	Tag:Get XML Tag	Property
ProjectItems	Set Tag	Tag:Set Tag	Property
ProjectItems	Set XML Tag	Tag:Set XML Tag	Property
ProjectItems	Library Item Type String	Library Item Type:String	Property
ProjectItems	Library Item Type	Library Item:Type	Property

Upgrading from LabVIEW 7.x

You might encounter the following compatibility issues when you upgrade to LabVIEW 8.2 from LabVIEW 7.*x*. Refer to the *Upgrading from LabVIEW 8.0* section of this document for information about other upgrade issues you might encounter.

Refer to the *LabVIEW Upgrade Notes* for each version of LabVIEW between versions 7.*x* and 8.0 at `ni.com/manuals` for more information about the new features and changes in each version.

Note The *LabVIEW Quick Reference Card* and the *LabVIEW Fundamentals Manual* did not change for LabVIEW 8.2. You can access the PDF versions of these documents in the `labview\manuals` directory. Refer to the *Upgrading from LabVIEW 8.0* section of this document for more information about these documents.

Platforms Supported

LabVIEW 8.*x* includes the following changes in platforms supported:

- LabVIEW 7.1 and later do not support Windows Me/98/95. LabVIEW 8.*x* does not support Windows NT.
- LabVIEW 8.*x* does not support Mac OS X 10.2 or earlier.
- LabVIEW 8.*x* does not support Sun Solaris.

System Requirements

LabVIEW 7.*x* requires a minimum of 128 MB of RAM, but National Instruments recommends 256 MB of RAM. LabVIEW 8.*x* requires a minimum of 256 MB of RAM, but National Instruments recommends 512 MB of RAM.

LabVIEW 7.*x* requires a screen resolution of 800 × 600 pixels, but National Instruments recommends a screen resolution of 1,024 × 768 pixels. LabVIEW 8.*x* requires a screen resolution of 1,024 × 768 pixels.

Windows

LabVIEW 7.*x* requires a minimum of a Pentium III or greater or Celeron 600 MHz or equivalent processor, but National Instruments recommends a Pentium 4 or equivalent processor. LabVIEW 8.*x* requires a minimum of a Pentium III or Celeron 866 MHz or equivalent processor, but National Instruments recommends a Pentium 4/M or equivalent processor.

LabVIEW 7.*x* requires at least 130 MB of disk space for the minimum LabVIEW installation or 550 MB disk space for the complete LabVIEW installation. LabVIEW 8.*x* 1.2 GB disk space for the complete LabVIEW installation.

Mac OS

LabVIEW 7.*x* requires at least 280 MB of disk space for the minimum LabVIEW installation or 350 MB disk space for the complete LabVIEW installation. LabVIEW 8.2 requires at least 500 MB of disk space for the minimum LabVIEW installation or 700 MB disk space for the complete LabVIEW installation.

Linux

LabVIEW 7.*x* requires a minimum of a Pentium III or greater or Celeron 600 MHz or equivalent processor, but National Instruments recommends a Pentium 4 or equivalent processor. LabVIEW 8.*x* requires a minimum of a Pentium III or Celeron 866 MHz or equivalent processor, but National Instruments recommends a Pentium 4/M or equivalent processor.

LabVIEW 7.*x* requires at least 200 MB of disk space for the minimum LabVIEW installation or 300 MB disk space for the complete LabVIEW installation. LabVIEW 8.2 requires at least 430 MB of disk space for the minimum LabVIEW installation or 620 MB disk space for the complete LabVIEW installation.

LabVIEW 7.*x* requires GNU C Library (`glibc`) version 2.1.3 or later, but National Instruments recommends GNU C Library version 2.2.4 or later. LabVIEW 8.*x* requires GNU C Library version 2.2.4 or later.

LabVIEW 7.*x* runs on Red Hat Linux 7.0 or later, Mandrake Linux 8.0 or later, SuSE Linux 7.1 or later, or Debian Linux 3.0 or later. LabVIEW 8.*x* runs on Red Hat Enterprise Linux WS 3 or later, MandrakeLinux/Mandriva 10.0 or later, or SuSE Linux 9.1 or later.

Custom Palette Views

LabVIEW 8.*x* does not support custom palette views. You can edit a palette set without using a custom palette view. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `lv8palette` for more information about palette changes in LabVIEW 8.0.

VI and Function Behavior Changes

The behavior of the following VIs and functions changed in LabVIEW 7.1 or 8.0.

.NET VIs and Applications

You must have the .NET Framework 1.1 Service Pack 1 or later to use .NET functions and applications in LabVIEW 8.*x*. You must remove Microsoft .NET Framework 1.1 Hotfix KB886904 before installing the .NET Framework 1.1 Service Pack 1.

If you load a .NET VI last saved in LabVIEW 7.*x*, LabVIEW 8.*x* might prompt you to find the assemblies to which that VI refers even if the assembly files are in the same directory as the VI or if you registered them by selecting **Tools»Advanced».NET Assembly References** in LabVIEW 7.*x*.

Analyze VI Algorithms

In LabVIEW 7.1 and later, the Analyze VIs use the BLAS/LAPACK algorithms. These VIs now produce more accurate results. In LabVIEW 8.*x*, these VIs are on the **Mathematics** and **Signal Processing** palettes.

Append Signals Express VI

In LabVIEW 7.*x*, if **Input Signal A** of the Append Signals Express VI is empty or not wired and you wire a single signal or a combined signal to **Input Signal B**, the **Appended Signals** output is empty. In LabVIEW 8.*x*, if **Input Signal A** is empty or not wired and you wire a single signal to **Input Signal B**, the Express VI returns **Input Signal B**. If you wire only a combined signal to **Input Signal B**, each signal in the combined signal appends the following signal to create one signal as a result.

Comparison Functions

In LabVIEW 7.*x* and earlier, when you use the Comparison functions to compare variant data, LabVIEW first compares the length of the two variants and then compares the variants bit by bit. LabVIEW 8.*x* begins the comparison of variant data with the type codes, which encode the actual type information of the variants, and then compares other type-specific attributes.

Dot Product VI

In LabVIEW 7.0, the Dot Product VI calculates the dot product of input vectors *X* and *Y* using the following equation:

$$\mathrm{X*Y} = \sum_{i=0}^{n-1} x_i y_i$$

In LabVIEW 7.1 and later, the Dot Product VI calculates the dot product of complex inputs using the following equation:

$$X^*Y = \sum_{i=0}^{n-1} x_i y_i^*$$

where y_i^* is the complex conjugate of y_i.

Easy Text Report VI (Mac OS and Linux)

The connector pane of the Easy Text Report VI changed. In LabVIEW 8.*x*, when you open a VI last saved in LabVIEW 7.*x* or earlier that uses the Easy Text Report VI, you must right-click the subVI and select **Relink To SubVI** from the shortcut menu.

Format Into String Function

In LabVIEW 7.*x*, using the `%o` or `%x` format specifier syntax elements with the Format Into String function rounds a floating-point input to a 32-bit integer before converting that input to a string.

In LabVIEW 8.*x*, these format specifier syntax elements cause this function to round floating-point inputs to 64-bit integers before converting the inputs to strings.

Join Numbers Function

In LabVIEW 7.*x* and earlier, the Join Numbers function coerces 32-bit integer inputs to 16-bit integers to create one 32-bit integer. In LabVIEW 8.*x*, the Join Numbers function joins 32-bit integer inputs to create one 64-bit integer.

Note If you open a LabVIEW 7.*x* VI in LabVIEW 8.*x*, LabVIEW coerces 32-bit integer inputs to 16-bit integers.

Mathematics VIs and Matrices

In LabVIEW 8.*x*, **Mathematics** VIs support the matrix data type. If you load a VI from LabVIEW 7.*x* in LabVIEW 8.*x* and the VI contains a Mathematics VI wired to a function that can use the matrix data type, a red 7.*x* glyph appears on the function to indicate that the function uses behavior from LabVIEW 7.*x*.

Number to String Conversion Functions

In LabVIEW 7.*x*, the Number to Hexadecimal String, Number to Octal String, and Number to Decimal String functions round a floating-point input to a 32-bit integer before converting that input to a string.

In LabVIEW 8.*x*, these functions round floating-point inputs to 64-bit integers before converting the inputs to strings. However, if you open a LabVIEW 7.*x* VI in LabVIEW 8.*x*, LabVIEW maintains compatibility and functionality by rounding floating-point inputs to 32-bit integers.

Open VI Reference Function

In LabVIEW 7.*x*, if the **vi path** input of the Open VI Reference function is a path and a VI in memory exists with the same name, LabVIEW returns a reference to the VI in memory, even if the path to the VI in memory does not match the path you specified.

In LabVIEW 8.*x*, if the **vi path** input of the Open VI Reference function is a string, LabVIEW opens the VI only if **vi path** matches the qualified filename of a VI in memory on that target. If **vi path** is a path, LabVIEW searches for a VI in memory with the same path on the same target. If LabVIEW does not find a VI with a matching path, LabVIEW tries to load the VI from disk at the specified path. An error occurs if LabVIEW cannot find the file or if the file conflicts with another VI in memory and targets.

Quick Scale VI

In LabVIEW 7.1 and earlier, if the **X** input of the Quick Scale 1D VI or the Quick Scale 2D VI is an array of zeros, this VI returns **max|X|** as **`0`** and **Y[i]=X[i]/Max|X|** or **Yij=Xij/Max|X|** as an array of `NaN`. In LabVIEW 8.*x*, if the **X** input of the Quick Scale VI is an array of zeros, this VI returns **max|X|** as **`0`** and **Y[i]=X[i]/Max|X|** or **Yij=Xij/Max|X|** as an array of zeros.

Read Key VI

In LabVIEW 7.*x* and earlier, you can use the Read Key VI to read a Japanese multibyte-character string encoded in Shift-JIS. You must wire `1` or `<Shift-JIS>` to the **multibyte encoding** input. In LabVIEW 8.*x*, the Read Key VI reads multibyte-character, encoded strings by default if you set the operating system locale to the appropriate encoding.

Scale VI

In LabVIEW 7.1 and earlier, if the **X** input of the Scale 1D VI or the Scale 2D VI is an array of zeros, this VI returns **scale** as **`0`**, **offset** as **`0`**, and **Y=(X–offset)/scale** as an array of `NaN`. In LabVIEW 8.*x*, if the **X** input of the Scale VI is an array of zeros, this VI returns **scale** as **`1`**, **offset** as **`0`**, and **Y=(X–offset)/scale** as an array of zeros.

Semaphore VIs

In LabVIEW 7.*x*, the Release Semaphore VI and the Acquire Semaphore VI do not attempt to run when the **error in** input receives an error. In LabVIEW 8.*x*, these VIs attempt to run even if the **error in** input receives an error. However, if you open a LabVIEW 7.*x* VI in LabVIEW 8.*x*, LabVIEW maintains the LabVIEW 7.*x* functionality.

SMTP Email VIs

In LabVIEW 7.*x* and earlier, you can specify a character set by wiring a value to the **character set** input of the SMTP Email VIs. In LabVIEW 8.*x*, the SMTP Email VIs assume the message is in the system character set. These VIs encode the message into UTF-8 format before sending the email. The SMTP Email VIs no longer have the **character set** or **translit** parameters.

Sort Complex Numbers VI

In LabVIEW 7.*x* and earlier, if you set the **method** input of the Sort Complex Numbers VI to **Magnitude**, LabVIEW does not change the sequence of elements with the same magnitude. In LabVIEW 8.*x*, if you set **method** to **Magnitude**, LabVIEW sorts elements of the same magnitude first with respect to their real parts and then with respect to their imaginary parts.

Unit Vector VI

In LabVIEW 7.*x* and earlier, the Unit Vector VI calculates the norm of an input vector using the following equation:

$$\|X\| = \sqrt{x_0^2 + x_1^2 + \ldots + x_{n-1}^2}$$

In LabVIEW 8.*x*, the Unit Vector VI calculates the norm of an input vector using the following equation:

$$\|X\| = \left| |x_0|^y + |x_1|^y + \ldots + |x_{n-1}|^y \right|^{\frac{1}{y}}$$

where X is the input vector, $\|X\|$ is the norm, and y is the norm type.

User VIs

VIs that you place in the `labview\help`, `labview\project`, or `labview\wizard` directories appear in the **Help**, **Tools**, and **File** menus, respectively. VIs that you place in these directories in LabVIEW 7.*x* and earlier might not work as expected in LabVIEW 8.*x* because LabVIEW 8.0 and later opens these VIs in a private application instance.

Use the VIMemory Get VIs in Memory VI in the `labview\vi.lib\Utility\allVIsInMemory.llb` to generate a list of all user VIs in memory in all application instances. Use the Get User Application Reference VI in the `labview\vi.lib\Utility\allVIsInMemory.llb` to create a reference to the current application instance. Refer to the *LabVIEW Help* for more information about application instances.

Deprecated VIs and Functions

LabVIEW 8.*x* does not support the following VIs and functions:

- LabVIEW 7.1 and later do not install the Polynomial Real Zero Counter VI. Use the Polynomial Real Zeros Counter VI instead.
- **(Mac OS)** LabVIEW 7.1 and later do not install the PPC VIs. Use the TCP VIs instead.
- LabVIEW 8.*x* does not support the QR Factorization VI. Use the QR Decomposition VI instead.
- LabVIEW 8.*x* does not support the Levenberg Marquardt or the Nonlinear Lev-Mar Fit VIs. Use the Nonlinear Curve Fit VI instead.
- In LabVIEW 8.*x*, the VISA Status Description function is not on the **Functions** palette. Use the Simple Error Handler or General Error Handler VIs instead.
- LabVIEW 8.*x* does not support the Chi Square Distribution, F Distribution, Normal Distribution, and T Distribution VIs. Use the Chi-Squared, F, Normal, and Student t instances, respectively, of the Continuous CDF VI instead.
- LabVIEW 8.*x* does not support the Inv Chi Square Distribution, Inv F Distribution, Inv Normal Distribution, and Inv T Distribution VIs. Use the Chi-Squared, F, Normal, and Student t instances, respectively, of the Continuous Inverse CDF VI instead.
- In LabVIEW 8.*x*, the 1D Linear Evaluation VI and the 2D Linear Evaluation VI are not on the **Functions** palette. Use the Linear Evaluation VI instead.
- In LabVIEW 8.*x*, the 1D Polynomial Evaluation VI and the 2D Polynomial Evaluation VI are not on the **Functions** palette. Use the Polynomial Evaluation VI instead.
- In LabVIEW 8.*x*, the 1D Rectangular to Polar VI and the 1D Polar to Rectangular VI are not on the **Functions** palette. Use the Re/Im To Polar function and the Polar To Re/Im function instead.
- In LabVIEW 8.*x*, the Harmonic Analyzer VI is not on the **Functions** palette. Use the Harmonic Distortion Analyzer VI instead to measure the **THD** or **component levels** outputs, or use the SINAD Analyzer VI to measure the **SINAD** or **THD Plus Noise** outputs.
- In LabVIEW 8.*x*, the Network Functions (avg) VI is not on the **Functions** palette. Use the Frequency Response Function (Mag-Phase), Frequency Response Function (Real-Im), Cross Spectrum (Mag-Phase), or Cross Spectrum (Real-Im) VIs instead.

- In LabVIEW 8.*x*, the Pulse Parameters VI is not on the **Functions** palette. Use the Transition Measurements VI instead to measure the **slew rate**, **duration**, **overshoot**, or **preshoot** outputs, the Pulse Measurements VI to measure the **period**, **pulse duration**, or **duty cycle** outputs, or the Amplitude and Levels VI to measure the **amplitude**, **high state level**, or **low state level** outputs.
- In LabVIEW 8.*x*, the Transfer Function VI is not on the **Functions** palette. Use the Frequency Response Function (Mag-Phase) or Frequency Response Function (Real-Im) VIs instead.
- In LabVIEW 8.*x*, the NI DIAdem Report Wizard Express VI is not on the **Functions** palette. Use the DIAdem Report Express VI instead.
- In LabVIEW 8.*x*, the VISA resource name constant and the IVI logical name constant are not on the **Functions** palette. To specify a VISA resource name, use the **VISA resource name** input of the VISA VIs. To specify an IVI logical name, use the appropriate input of the appropriate driver VI that initializes the instrument.
- In LabVIEW 8.*x*, the error ring constant is not on the **Functions** palette. Use a 32-bit signed integer constant instead to enter the error code that you want.
- **(Windows and Linux)** In LabVIEW 8.*x*, the Sound VIs available on the **Sound** palette in LabVIEW 7.*x* are not on the **Functions** palette. Use the Sound VIs in LabVIEW 8.*x* instead. The examples shipped with LabVIEW 7.*x* do not ship with LabVIEW 8.*x*.
- **(Mac OS)** LabVIEW 8.*x* continues to ship with the Sound VIs shipped with LabVIEW 7.1. The examples shipped with LabVIEW 7.*x* do not ship with LabVIEW 8.*x*.

File I/O VIs and Functions

In LabVIEW 8.*x*, the Read Characters From File VI is not on the **Functions** palette. Use the Read from Text File function instead.

In LabVIEW 8.*x*, the Open/Create/Replace File VI is not on the **Functions** palette. Use the Open/Create/Replace File function instead. The following functions include some of the functionality of the Open/Create/Replace File VI in LabVIEW 7.*x* and earlier.

- Use the Get File Size function to determine the size of a file.
- Use the File Dialog Express VI to specify the start path, file pattern, and default name of a file or directory for a file dialog box.
- Use the Refnum to Path function to convert a reference to a path.
- Use the Write to Binary File function to create platform-independent text files or other types of binary files, and use the Read from Binary File function to read the resulting binary files.

In LabVIEW 8.*x*, the Read File and Write File functions are not on the **Functions** palette. Use the Read from Binary File and Write to Binary File functions instead.

In LabVIEW 8.*x*, the Write Characters To File VI is not on the **Functions** palette. Use the Write to Text File function instead.

In LabVIEW 8.*x*, the Access Rights function is not on the **Functions** palette. Use the Get Permissions and Set Permissions functions instead.

In LabVIEW 8.*x*, the EOF function is not on the **Functions** palette. Use the Get File Size and Set File Size functions instead.

In LabVIEW 8.*x*, the List Directory function is not on the **Functions** palette. Use the List Folder function instead.

In LabVIEW 8.*x*, the Lock Range function is not on the **Functions** palette. Use the Deny Access function instead.

If you open a VI built in LabVIEW 7.*x* that includes the New Directory function on the block diagram, LabVIEW 8.*x* replaces that function with the Create Folder function. If the folder you specified in the **path** input does not exist, the Create Folder function creates the directory rather than returning an error, as the New Directory function did.

In LabVIEW 8.*x*, the Seek function is not on the **Functions** palette. Use the Get File Position and Set File Position functions instead.

In LabVIEW 8.*x*, the Type and Creator function is not on the **Functions** palette. Use the Get Type and Creator and Set Type and Creator functions instead.

In LabVIEW 8.*x*, the Volume Info function is not on the **Functions** palette. Use the Get Volume Info function instead.

In LabVIEW 8.*x*, the Open File and New File functions are not on the **Functions** palette. The Read Lines From File VI is not on the **Functions** palette but ships with LabVIEW for compatibility.

In LabVIEW 8.*x*, the Read From I16 File, Read From SGL File, Write To I16 File, and Write To SGL File VIs are not on the **Functions** palette. Use the Read from Binary File and Write to Binary File VIs instead.

Property, Method, and Event Behavior Changes

The behavior of the following properties, methods, and events changed in LabVIEW 7.1 or 8.0.

Application Properties and Methods

In LabVIEW 8.*x*, the behavior of some Application properties and methods depends on the application instance to which they belong. For example, the behavior of the Application:All VIs in Memory property depends on the application instance in which you use it. This property returns a list of all VIs in memory in the same application instance as the property. However, the behavior of the Application:Directory Path property does not depend on the application instance in which you use it. This property returns the absolute path to the directory in which the application is located. This information does not change with each application instance.

Refer to the *LabVIEW Help* for more information about application instances.

Front Panel:Open Method

The LabVIEW 7.0 Open FP method was renamed to Old Open FP in LabVIEW 7.1. LabVIEW 7.1 includes a different Open FP method that does not return an error if the front panel is already open. The LabVIEW 7.1 Open FP method was renamed to Front Panel:Open in LabVIEW 8.*x*. If you have VIs that use the Old Open FP method from LabVIEW 7.0, replace the method with the Front Panel:Open method.

Run VI Method

In LabVIEW 7.1, if you set the **Auto Dispose Ref** input of the Run VI method to TRUE, LabVIEW automatically disposes the reference after the VI stops running. In LabVIEW 8.*x* and later, LabVIEW also immediately disposes the reference if the method returns an error. This behavior might break a VI at run time if part of the block diagram depends on the reference.

Key Down and Key Repeat Events

The **VKey** data field of the Key Down, Key Down?, Key Repeat, and Key Repeat? events for VIs and controls now has separate values for the <Return> key on the alphanumeric section of the keyboard and the <Enter> key on the numeric keypad. In LabVIEW 7.*x* and earlier, when the <Enter> key or the <Return> key generates one of these events, LabVIEW returns **`<Enter>`** in the **VKey** data field. In LabVIEW 8.*x*, when the <Enter> key or the <Return> key generates one of these events, LabVIEW returns **`<Enter>`** or **`<Return>`**, respectively, in the **VKey** data field.

(Mac OS) LabVIEW 8.*x* accepts only <Control>-click for shortcut menus and does not receive the <Command>-click key combination. If you are emulating this behavior with an Event structure, modify your VIs to emulate the new behavior.

ListBox Properties

In LabVIEW 7.*x* and earlier, if you set the Top Row property of a listbox to a row that is below the bottom item of the listbox, LabVIEW pins the row to the last visible item. In LabVIEW 8.*x*, the number of visible items in the listbox does not limit the row number you can wire to this property.

LabVIEW 8.*x* does not support the Double-Click property for single-column listboxes. Use the Get Double-Clicked Row method instead.

Owning VI Property

In LabVIEW 7.*x* and earlier, the Owning VI property returns a reference to the VI to which the object belongs. This reference keeps the VI in memory. In LabVIEW 8.*x*, the reference the Owning VI property returns does not keep the VI in memory. If the owning VI is removed from memory, this reference becomes invalid. Use the Open VI Reference function to obtain a reference to a VI that stays in memory until you explicitly close the reference.

Text Property

In LabVIEW 7.*x* and earlier, the Text property returns a string in normal display. In LabVIEW 8.*x*, the Text property returns a string in the same text display as the front panel object. For example, if you display a string control in password display, the Text property returns the string in password display.

TreeControl Properties

In LabVIEW 7.*x* and earlier, the Active Cell Properties:Cell Size:Height and Active Cell Properties:Cell Size:Width properties return 17 pixels for each line in the tree control. In LabVIEW 8.*x*, the Active Cell:Cell Size: Height and Active Cell:Cell Size:Width properties return 16 pixels for each line in the tree control.

VI Strings Methods

Strings that you export from previous versions of LabVIEW using the Export VI Strings method might not import properly in LabVIEW 8.*x* when you use the VI Strings:Import method.

Deprecated Properties, Methods, and Events

LabVIEW 8.*x* does not support the following properties, methods, and events.

Cursor Properties

LabVIEW 8.*x* does not support the Cursor Lock Style property. Use the Cursor Mode property instead.

ListBox, Table, DigitalTable, and TreeControl Properties and Events

LabVIEW 8.*x* does not support the Cell Foreground Color property for multicolumn listboxes. Use the Active Cell:Cell Font:Color property instead.

LabVIEW 8.*x* does not support the Cell FG Color property for tables or digital tables. Use the Active Cell:Cell Font:Color property for tables and digital tables instead.

LabVIEW 8.*x* does not support the Active Cell Properties:Foreground Color property for tree controls. Use the Active Cell:Cell Font:Color property instead.

LabVIEW 8.*x* does not support the Drag, Drag?, Drop, and Drop? events in the TreeControl class. Use the Drag Ended, Drag Enter, Drag Leave, Drag Over, Drag Source Update, Drag Starting, Drag Starting?, and Drop events in the Control class instead.

NamedNumeric Properties

LabVIEW 8.*x* does not support the Named Numeric Colors, Named Numeric Colors:BG Color, or Named Numeric Colors:Text Color properties for named numeric objects. Use the Text Colors, Text Colors:BG Color, and Text Colors:Text Color properties, respectively, instead.

Panel Properties

LabVIEW 8.*x* does not support the Color property in the Panel class. If you use this property in LabVIEW 8.*x*, the property applies only to the upper-leftmost pane. Use the Pane Color property in the Pane class instead.

Subpanel Properties

In LabVIEW 8.*x*, use the pane of a subVI in a subpanel to configure the visibility of scroll bars for subpanel controls and to scale the front panel in subpanel controls.

LabVIEW 8.*x* does not support the X Scrollbar Visible property for subpanel controls. Use the Horizontal Scrollbar Visibility property for panes instead.

LabVIEW 8.*x* does not support the Y Scrollbar Visible property for subpanel controls. Use the Vertical Scrollbar Visibility property for panes instead.

LabVIEW 8.*x* does not support the Scale Panel property for subpanel controls. Use the Set Scaling Mode method for panes instead.

VI Properties, Methods, and Events

LabVIEW 8.*x* does not support the Front Panel Window:Auto Center property. Use the Front Panel:Center method instead.

LabVIEW 8.*x* does not support the Front Panel Window:Size to Screen property. Use the Front Panel Window:State property instead.

LabVIEW 8.*x* does not support the Front Panel Window:Origin property in the VI class. If you use this property in LabVIEW 8.*x*, the property applies only to the upper-leftmost pane. Use the Origin property in the Pane class instead.

LabVIEW 8.*x* does not support the Front Panel Window:Show Scroll Bars property in the VI class. If you use this property in LabVIEW 8.*x*, the property applies only to the upper-leftmost pane. Use the Horizontal Scrollbar Visibility and Vertical Scrollbar Visibility properties in the Pane class instead.

LabVIEW 8.*x* does not support the Get Front Panel Scaling Mode or Set Front Panel Scaling Mode methods in the VI class. If you use these methods in LabVIEW 8.*x*, the methods apply only to the upper-leftmost pane. Use the Get Scaling Mode and Set Scaling Mode methods in the Pane class instead.

LabVIEW 8.*x* does not support the Mouse Down, Mouse Down?, Mouse Enter, Mouse Leave, Mouse Move, or Mouse Up events in the VI class. Use the Mouse Down, Mouse Down?, Mouse Enter, Mouse Leave, Mouse Move, and Mouse Up events in the Pane class, respectively, instead.

Application Item Tags

The following application item tags do not exist in LabVIEW 8.*x*:

- `APP_BUILD_STANDALONE_APP`
- `APP_DN_ASSEMBLY_REFS`
- `APP_EDIT_VI_LIBRARY`

- `APP_SAVE_WITH_OPTIONS`
- `APP_SHOW_CLIPBOARD`
- `APP_SRC_CODE_CTRL`
- `APP_SWITCH_EXEC_TARGET`
- `APP_UPDATE_VXI`
- `APP_VIEW_PRINTED_MANUALS`

When you use a run-time menu (`.rtm`) file that was saved in a previous version of LabVIEW and the file contains a deleted tag, LabVIEW 8.*x* automatically removes the tag from the `.rtm` file when you save the file in the **Menu Editor** dialog box. The deleted application item tags are reserved by LabVIEW and you cannot use them as user tags.

HiQ Support

National Instruments does not support HiQ functionality in LabVIEW 8.*x*. If an application uses HiQ VIs, consider replacing them with the Mathematics and Signal Processing VIs.

Error List Window

In LabVIEW 7.*x* and earlier, the **VI List** section of the **Error list** window shows errors for all VIs in memory. In LabVIEW 8.*x*, the **Items with errors** section of the **Error list** window shows errors for all items in memory, such as VIs and libraries. If two or more items have the same name, this section shows the specific application instance for each ambiguous item. Refer to the *LabVIEW Help* for more information about application instances.

VI String File Syntax

LabVIEW 8.*x* searches for a new set of tags, `<GROUPER></GROUPER>`, when you import VI string files by selecting **Tools»Advanced»Import Strings** or by using the VI Strings:Import method. This set of tags denotes front panel objects that are grouped together. Therefore, in LabVIEW 8.*x*, you cannot import VI string files saved in previous versions of LabVIEW.

LabVIEW 7.1 and earlier lists listbox strings in the `<ITEMS>` section of its private data. LabVIEW 8.*x* lists listbox strings in the `<STRINGS>` section of its private data. Also, in LabVIEW 7.1 and earlier, a listbox can have only one font, which LabVIEW lists in the `<LBLABEL>` section of its private data. In LabVIEW 8.*x*, the listbox can have multiple fonts, which LabVIEW lists in the `<CELL_FONTS>` section of its private data.

LabVIEW 7.1 and earlier lists multicolumn listbox strings in its default data. However, the default data for a multicolumn listbox is an integer or array of integers. LabVIEW 8.*x* lists multicolumn listbox strings in its private data.

LabVIEW 7.1 and earlier exports neither strings nor fonts for tree controls. LabVIEW 8.*x* can export both tree control strings and fonts, and it exports them in the same format as the listbox and multicolumn listbox.

In LabVIEW 8.*x*, each line of an export file contains no more than two tags for private or default data. LabVIEW 8.*x* also indents items once for each nesting level.

Complete the following steps to convert VI string files to the LabVIEW 8.*x* format.

1. Import the VI string file in the previous version of LabVIEW.
2. Save the VI.
3. Load the VI in LabVIEW 8.*x*.
4. Select **Tools»Advanced»Export Strings** to save the VI string file in the LabVIEW 8.*x* format.

Converting Type Descriptor Data to and from LabVIEW 7.x

The format in which LabVIEW stores type descriptors changed in LabVIEW 8.*x*. LabVIEW 7.*x* stores type descriptors in 16-bit flat representation. LabVIEW 8.*x* stores type descriptors in 32-bit flat representation. This change eliminates the 64 KB size limitation of type descriptors.

LabVIEW 8.*x* provides a mechanism for reading type descriptors written in LabVIEW 7.*x* and writing type descriptors that LabVIEW 7.*x* can read. The Flatten To String function has a **Convert 7.x Data** shortcut menu item. If you right-click the function and select this menu item, the function treats input data as if it were written for LabVIEW 7.*x*. When you select the **Convert 7.x Data** shortcut menu item and the **data string** output is wired, LabVIEW 8.*x* places a red `7.x` glyph on the function to indicate that it is converting data to or from LabVIEW 7.*x* format. To avoid the conversion of data, select the **Convert 7.x Data** shortcut menu item again to remove the checkmark.

In LabVIEW 8.*x*, when you load a VI last saved in LabVIEW 7.*x* or earlier, LabVIEW 8.*x* automatically sets the **Convert 7.x Data** attribute on the Flatten To String function. The function continues to operate as in LabVIEW 7.*x* and earlier. If you want a VI to use the LabVIEW 8.*x* type descriptor format, right-click the Flatten To String function and select

Convert 7.x Data from the shortcut menu to remove the checkmark. Use the LabVIEW 8.*x* type descriptor format if VIs do not need to manipulate files that contain data written in LabVIEW 7.*x* or earlier and do not send or receive data to or from VIs running in LabVIEW 7.*x* or earlier. Support for the previous type descriptor format might be discontinued in future versions of LabVIEW.

Migrating from the LabVIEW Built-In Source Control Provider

The built-in source control provider from LabVIEW 7.*x* and earlier is not available in LabVIEW 8.*x*. If you want to use source control in LabVIEW, you must select a third-party source control provider. If you used the built-in provider in previous versions, you must migrate the files to another provider to use source control in LabVIEW. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `exgucn` for the most current list of third-party source control providers supported in LabVIEW.

When you migrate files to a new source control provider, you lose the revision history stored in the built-in provider. You cannot transfer the previous versions of the files to the new provider.

Complete the following steps to migrate files from the built-in source control provider to a third-party source control provider.

1. In the previous version of LabVIEW, make sure that the files included in the LabVIEW built-in source control provider are checked in by all users.
2. On the computer where you want to add the files to the new source control provider, use the built-in provider to get the latest versions of all the files.
3. Use the built-in provider to check out the files from source control.
4. In the third-party source control provider, configure the settings you want for the new source control project.
5. Configure LabVIEW to work with the third-party source control provider. Refer to the **Fundamentals»Organizing and Managing a Project»How-to»Using Source Control in LabVIEW** book on the **Contents** tab of the *LabVIEW Help* for information about configuring LabVIEW to work with a third-party source control provider.
6. Create a LabVIEW project. Add the files included in the built-in provider to the project. When LabVIEW prompts you, add the files to source control. You also can add the files directly in the third-party provider. Refer to the **Fundamentals»Organizing and Managing a Project»How-to»Creating a LabVIEW Project** book on the

Contents tab of the *LabVIEW Help* for information about creating a LabVIEW project.

Converting NaN Strings to Integer Types (Windows)

In LabVIEW 7.*x*, when you explicitly or implicitly convert `NaN` to an integer, the value becomes the smallest value for that integer data type. For example, converting `NaN` to a 16-bit signed integer produces the value –32,768, the smallest possible value for a 16-bit signed integer.

In LabVIEW 8.*x*, when you explicitly or implicitly convert `NaN` to an integer, the value becomes the largest value for that integer data type. For example, converting `NaN` to a 16-bit signed integer produces the value 32,767, the largest possible value for a 16-bit signed integer.

Constants Wired to Case Structures

In LabVIEW 7.*x* and earlier, you can keep subVIs in memory by wiring a constant to a Case structure and placing the subVI in a case that does not execute. For example, if you wire a TRUE constant to a Case structure and place a subVI in the FALSE case of the Case structure, LabVIEW loads the subVI along with the calling VI. LabVIEW 8.*x* removes any code that does not execute. Therefore, if you load a VI in LabVIEW 8.*x* that was saved in an earlier version of LabVIEW with a constant wired to a Case structure, LabVIEW changes the constant to a hidden control to maintain the behavior from the earlier version of LabVIEW.

Delaying Operating System Messages

In LabVIEW 7.*x*, LabVIEW processes operating system messages while running callback VIs for handling .NET and ActiveX events. In LabVIEW 8.*x*, LabVIEW delays the processing of operating system messages until the callback VI stops execution or until you load a modal dialog box. This delay allows callback VIs to execute without interruption and prevents LabVIEW from firing an event within another event, which can result in a deadlock state.

You cannot make synchronous calls to non-modal dialog boxes from a callback VI. You must asynchronously call a non-modal dialog box from a callback VI by invoking a Run VI method on the dialog and wiring a FALSE Boolean constant to the **Wait Until Done** input of the method.

In LabVIEW 7.*x*, LabVIEW processes operating system messages while running DLL or shared library functions. In LabVIEW 8.*x*, LabVIEW delays the processing of operating system messages until the end of calls to DLL functions or until you load a modal dialog box from the DLL. This delay allows DLL functions to execute without interruption and prevents

LabVIEW from calling the same DLL while a DLL function is running, which can result in a deadlock state.

If you use this default behavior, you cannot make synchronous calls to non-modal dialog boxes while a DLL runs. You must call a non-modal dialog box asynchronously from a DLL by invoking a Run VI method on the dialog and wiring a FALSE Boolean constant to the **Wait Until Done** input of the method.

You can choose whether to delay operating system messages in DLLs that you build. Right-click the DLL in the **Project Explorer** window, select **Properties** from the shortcut menu, select **Advanced** from the **Category** list, and remove the checkmark from the **Delay operating system messages in shared library** checkbox to process operating system messages while DLL functions run.

Resource Manager (Mac OS)

LabVIEW 7.*x* and earlier provide undocumented capabilities with which you can read and write Macintosh resource files. In LabVIEW 8.*x*, these methods do not exist. Utilities that make use of these undocumented capabilities do not work, and you therefore cannot read or write Macintosh resource files from VIs.

One- and Two-Button Dialog Boxes

In LabVIEW 7.*x* and earlier, you cannot abort programmatically a VI displaying a one-button dialog box or two-button dialog box. In LabVIEW 8.*x*, you can abort programmatically a VI displaying these dialog boxes by using the Abort VI method.

Property and Invoke Nodes

If you create an implicitly linked Property Node or Invoke Node from a cursor legend in LabVIEW 7.*x*, LabVIEW deletes the node when you open the VI in LabVIEW 8.*x*.

Updating Shared Libraries

If you build a shared library (DLL) in LabVIEW 7.*x* or earlier that links to `labview.lib`, link the shared library to `labviewv.lib` instead in LabVIEW 8.*x*. Refer to the *LabVIEW Help* for more information about linking shared libraries to `labviewv.lib`.

Margin Values for Printing

In LabVIEW 7.*x* and earlier, the **Margins** option on the **Printing** page of the **Options** dialog box uses centimeters for margin values. In LabVIEW 8.*x*, the **Margins** option uses millimeters for margin values.

Upgrading from LabVIEW 6.x

You might encounter the following compatibility issues when you upgrade to LabVIEW 8.2 from LabVIEW 6.*x*. Refer to the *Upgrading from LabVIEW 7.x* and *Upgrading from LabVIEW 8.0* sections of this document for information about other upgrade issues you might encounter.

Refer to the *LabVIEW Upgrade Notes* for each version of LabVIEW between versions 6.*x* and 8.0 at `ni.com/manuals` for more information about the new features and changes in each version.

Changes to the Waveform Data Type

In LabVIEW 7.0, the waveform data type uses the time stamp data type for the **t0** component rather than a double-precision, floating-point number. If you save data in the waveform data type to a file without including information about the data type in LabVIEW 6.*x*, you might encounter an error if you try to retrieve that data in LabVIEW 7.*x* and later.

In the LabVIEW 7.*x* and later, the Read Waveform from File VI converts the old waveform data type format in a file to the new waveform data type format. This VI displays a dialog box that prompts you to accept the conversion. In the LabVIEW Run-Time Engine, the Read Waveform from File VI cannot perform this conversion and returns an error instead. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `exd9zq` for more information about migrating waveform data from LabVIEW 6.*x* to LabVIEW 7.*x* and later.

Serial Compatibility VIs

In LabVIEW 7.*x* and later, the Serial Compatibility VIs do not appear on the **Functions** palette. Use the VISA VIs and functions to build VIs that communicate with VXI devices.

In LabVIEW 7.*x* and later, LabVIEW does not use the `serpdrv` driver to communicate with the serial driver of the operating system. LabVIEW includes compatible VIs based on VISA. For new applications, use the VISA and Serial VIs and functions to control serial devices. Any VIs built in previous versions of LabVIEW that include Serial VIs continue to work in LabVIEW 7.1 and later.

If you reconfigured the mapping of port numbers to ports, you must specify a mapping to those ports. Use the set serial alias ports VI in the `labview\vi.lib\Instr\_sersup.llb` to specify the serial port mappings. Wire a string array to the **VISA Aliases** input of the VI and enter the port names you use in the input array. Each element in the array should correspond to a port. For example, if you configured port 0 to map to the VISA alias MySerialPort, enter `MySerialPort` as the first element of the **VISA Aliases** input array. You must call the set serial alias ports VI before you call the VISA Configure Serial Port VI.

Refer to the `labview\examples\instr\smplserl.llb` for examples of using the VISA VIs and functions to control serial instruments.

Default Data in Loops

In LabVIEW 6.0 and earlier, For Loops produce undefined data if the loop does not execute. In LabVIEW 6.1 and later, For Loops produce default data if you wire `0` to the count terminal of the For Loop or if you wire an empty array to the For Loop as an input with auto-indexing enabled. The loop does not execute, and any output tunnel with auto-indexing disabled contains the default value for the tunnel data type.

Remote Front Panel License

The LabVIEW Full Development System and the Application Builder include a remote front panel license that allows one client to view and control a front panel remotely. The LabVIEW Professional Development System includes a remote front panel license that allows five clients to view and control a front panel remotely.

You can upgrade the remote front panel license to support more clients.

Multiple Thread Allocation

LabVIEW 7.1 and later allocate more threads for executing VIs than in versions earlier than LabVIEW 7.1. Because of this change, you might encounter errors with multiple threads if you incorrectly mark Call Library Function Nodes as reentrant when the DLL you call is not actually reentrant. Refer to the *LabVIEW Help* for more information about the Call Library Function Node and reentrancy.

To change how LabVIEW allocates threads, use the threadconfig VI in the `labview\vi.lib\Utility\sysinfo.llb`. You also can disable reentrancy for VIs by selecting **File»VI Properties**, selecting **Execution** from the **Category** pull-down menu, and removing the checkmark from the **Reentrant execution** checkbox.

Refer to the *LabVIEW Help* for more information about thread allocation.

Instrument Drivers

The LabVIEW package in LabVIEW 7.*x* and later does not include the LabVIEW Instrument Driver Library CD, which contains instrument drivers. Download instrument drivers from the National Instruments Instrument Driver Network at `ni.com/idnet`. The National Instruments Device Drivers CD includes NI-DAQ, NI-VISA, and other National Instruments drivers.

Units and Conversion Factors

In LabVIEW 7.*x* and later, you do not need to use the Convert Unit function to remove the extra unit after using the Compound Arithmetic function.

The unit conversion factors in LabVIEW 7.1 and later more closely match the guidelines published by the National Institute for Standards and Technology (NIST) in the *Guide for the Use of the International System of Units (SI)*. Also, the `calorie` unit now is `calorie (thermal)`, and `horse power` now is `horsepower (electric)`. The abbreviations for these units did not change. The following table details the changes in unit conversion factors between LabVIEW 6.1 and 7.*x* and later.

Unit	6.1 Definition	7.*x* and Later Definition
astronomical unit (AU)	149,498,845,000 m	149,597,900,000 m
British Thermal Unit (mean)	1055.79 J	1055.87 J
electron volt (eV)	1.602e–19 J	1.60217642e–19 J
foot-candle	10.764 lx	10.7639 lx
horse power versus horse power (electric)	745.7 W	746 W. The new conversion is exact.
imperial gallon	4.54596 l	4.54609 l
light year	9.4605 Pm	9.46073 Pm
pound force	4.448 N	4.448222 N
rod	16.5 ft	5.029210 m
slug	32.174 lb	14.59390 kg
unified atomic mass (u)	1.66057e–27 kg	1.66053873e–27 kg

Defer Panel Updates Property

In LabVIEW 6.1 and earlier, LabVIEW waits until the Defer Panel Updates property is FALSE to redraw any front panel objects with pending changes. In LabVIEW 7.0 and later, when you set this property to TRUE, LabVIEW redraws any front panel objects with pending changes and then defers all new requests for front panel updates. In some cases, this change can cause LabVIEW to redraw the changed elements of the front panel an extra time.

Data Ranges for Numeric Controls

In LabVIEW 6.1 and earlier, some numeric controls have a default minimum value of `0.00`, maximum value of `0.00`, increment value of `0.00`, and out of range action of **Ignore**. In LabVIEW 7.*x* and later, these numeric controls use the default data range values for the data type.

Coercion Dots and Type Definitions

In LabVIEW 6.1 and later, wires include information about type definitions, so you might notice more coercion dots on block diagrams. If you wire a type definition to a VI or function terminal that is not a type definition terminal, a coercion dot appears. A coercion dot also appears if you wire an output terminal that is a type definition to an indicator that is not a type definition. These coercion dots indicate where you are not using type definitions consistently in the VIs. In this case, coercion dots do not affect run-time performance.

Refer to the *LabVIEW Help* for information about using the Flatten To String function to flatten type definitions.

File Dialog Box Button Label

In LabVIEW 6.1 and earlier, the file dialog box that the File Dialog function displays has a button label of **Save** if the user can enter a new filename. Otherwise, the button label is **Open**. In LabVIEW 8.*x*, the button label on the file dialog box that the File Dialog Express VI displays is **OK** in all cases unless you change it. Use the **button label** input of the File Dialog Express VI to change the label of the button. If you use the File Dialog Express VI in an existing VI, consider reviewing the behavior of the VI to make sure the default label of **OK** is appropriate to the functionality of the VI.

Control Online Help Function

The **Path to the help file** input of the Control Online Help function now is required. You can wire a compiled help filename (`.chm` or `.hlp`) or the full path to a compiled help file to the input. If you wire only a compiled help filename, LabVIEW searches the `labview\help` directory for that file.

Displaying the Front Panel When Loaded

In LabVIEW 7.*x* and later, if you configure a VI to display the front panel when LabVIEW loads the VI and you load the VI using the VI Server, LabVIEW does not display the front panel. You must use the Front Panel: Open method to display the front panel programmatically.

Open VI Reference Function

In LabVIEW 6.1 and earlier, if you do not wire a value to the **options** parameter of the Open VI Reference function, LabVIEW instantiates a VI from a template if the template is not already in memory. If the template is in memory, LabVIEW opens a reference to the template. In LabVIEW 7.0 and 7.1, if you use the Open VI Reference function to create a reference to a template that is already in memory, the function returns an error unless you specify `0x02` in the **options** parameter. In LabVIEW 8.0 and later, if you use the Open VI Reference function to create a reference to a template, LabVIEW instantiates a VI from the template even if that template is already in memory.

Exponential Representation

In LabVIEW 6.0 and earlier, the ^ operator represents exponentiation in the Formula Node. In LabVIEW 6.1 and later, the operator for exponentiation is `**`—for example, `x**y`. The ^ operator represents the bitwise exclusive or (XOR) operation.

IVI Configuration Store File

The IVI Configuration Store file format now requires that all names be case-sensitive. If you use logical names, driver session names, or virtual names in your application, make sure that the name you use matches the name defined in the IVI Configuration Store file exactly, without any variations in the case of the characters in the name.

Technical Support Form

In LabVIEW 7.*x* and later, the LabVIEW installation program does not install `techsup.llb`. Refer to the National Instruments Web site at `ni.com/support` to solve installation, configuration, and application problems and questions.

Upgrading from LabVIEW 5.x or Earlier Versions

Refer to the National Instruments Web site at `ni.com/info` and enter the info code `ext8h9` for information about upgrading to LabVIEW 8.2 from LabVIEW 5.*x* or earlier.

LabVIEW 8.2 Features and Changes

Refer to the *LabVIEW Help* for more information about LabVIEW 8.2 features, including programming concepts, step-by-step instructions, and reference information. Access the *LabVIEW Help* by selecting **Help» Search the LabVIEW Help**.

Refer to the `readme.html` in the `labview` directory for known issues, additional compatibility issues, and information about late addition features in LabVIEW 8.2.

LabVIEW Documentation

LabVIEW 8.2 includes the following documentation enhancements:

- A **Submit feedback on this topic** link appears at the bottom of all *LabVIEW Help* topics. To provide feedback on a help topic, click and complete the **Documentation Suggestion Details** form. This link only appears in English versions of the *LabVIEW Help*.
- Concept topics include a navigation table in the upper, right-hand corner of the help topic. Click a link in the table to jump to the related subtopic.

New Example VIs

Refer to the **New Examples for LabVIEW 8.*x*** folder on the **Browse** tab of the NI Example Finder to view descriptions for and launch example VIs added to LabVIEW 8.*x*.

Launch Time Improvement

LabVIEW 8.2 launches faster than LabVIEW 8.0 due to performance optimizations.

Block Diagram Enhancements

LabVIEW 8.2 includes the following enhancements to the block diagram and related functionality.

Default Color Changes

The default colors of the following block diagram components changed to improve visibility:

- Error cluster wires and terminals appear dark yellow instead of pink on the block diagram.
- Coercion dots appear red instead of gray by default. To change the color of coercion dots, select **Tools»Options** and select **Colors** from

the **Category** list. Remove the checkmark from the **Use default colors** checkbox and click the **Coercion Dots** color box to select a different color.

Removing Breakpoints from a VI Hierarchy

From a VI, select **Edit»Remove Breakpoints from Hierarchy** option to remove all breakpoints from the VI hierarchy. You must manually remove breakpoints in dynamically called VIs or VIs referenced by the Static VI Reference function.

Performance Optimized with Constants

LabVIEW uses constant folding to optimize the performance of VIs. With constant folding, LabVIEW stores constant values when it compiles VIs instead of calculating them at run time. For constants wired to structures, LabVIEW calculates the output values of the structures when it compiles VIs and stores the values so they are available at run time.

You can display constant folding hash marks on the block diagram by selecting **Tools»Options**, selecting **Block Diagram** from the **Category** list, and placing checkmarks in the **Show constant folding of wires** and **Show constant folding of structures** checkboxes. When you place a checkmark in the **Show constant folding of wires** checkbox, gray hash marks appear on the wires attached to constants that are constant folded. When you place a checkmark in the **Show constant folding of structures** checkbox, gray hash marks appear inside structures that are wired to constants. The hash marks might not appear in a VI until after you run or save the VI.

LabVIEW 8.2 also folds computed constants you wire to Case structure selector terminals and While Loop conditional terminals.

Miscellaneous Block Diagram Enhancements

LabVIEW 8.2 includes the following miscellaneous enhancements to the block diagram.

- When you right-click the **reference out** output of a Property or Invoke Node and select **Create** from the shortcut menu, the **Create** menu displays the properties or methods in the same class as the Property or Invoke Node.
- The Visible Items:Hierarchy Lines Visible property accepts the following values: If **Visible**, LabVIEW always displays lines to outline the hierarchy of the items as vertical and horizontal lines to the left of the items in the tree control. If **Invisible**, the hierarchy lines are always invisible. If **OS Default**, LabVIEW displays the hierarchy lines if trees in the operating system show hierarchy lines.

Front Panel Enhancements

LabVIEW 8.2 includes the following enhancements to the front panel and related functionality.

Setting Background Images for Panes

You can set and import background images for panes. Right-click the scroll bar of a pane and select **Properties** from the shortcut menu. In the **Pane Properties** dialog box, select an image from the **Background** list.

To select an image that does not appear in the **Background** list, click the **Browse** button. LabVIEW supports BMP, JPEG, and PNG graphic formats for background images. You also can use the Background Image property to set a pane background image programmatically.

Note If you select an image that does not appear in the **Background** list, LabVIEW does not add that image to the **Background** list permanently. To add an image to the list permanently, you must save the image in the `labview\resource\backgrounds` directory.

When you save a VI that contains a pane with a background image, LabVIEW saves the pane background image with the VI.

Locking Knobs and Dials at Minimum and Maximum

By default, knobs and dials cannot rotate past their minimum or maximum values.

To disable this locking behavior, right-click the knob or dial, select **Properties** from the shortcut menu, and remove the checkmark from the **Lock at minimum and maximum** checkbox. Locking prevents a knob or dial from jumping from minimum to maximum or maximum to minimum values. Disabling this behavior might cause unintended jumps between values.

In LabVIEW 8.2, when you open a VI last saved in LabVIEW 8.0 or earlier, locking is disabled. To enable locking, place a checkmark in the **Lock at minimum and maximum** checkbox.

Multiple-Item Dragging within Tree Controls and Listboxes

You can drag and drop multiple items from and to tree controls and listboxes. Right-click the tree control or listbox and select **Selection Mode»0 or More Items** or **Selection Mode»1 or More Items** from the shortcut menu to enable multiple-item dragging and dropping. Initiating a drag with multiple items selected moves all the selected items.

Digital Waveform Graph Enhancements

Refer to the **Fundamentals»Graphs and Charts** book on the **Contents** tab in the *LabVIEW Help* for more information about digital waveform graphs.

LabVIEW 8.2 includes the following enhancements to digital waveform graphs.

Setting Line Thickness

Line Style replaces **Thick Line Location** in the shortcut menu of the digital waveform graph plot legend. Right-click the plot in the plot legend and select **Line Style** from the shortcut menu to set the line thickness. LabVIEW 8.2 includes a new **Line Style** option to thicken the entire line.

Darkening Compare Data

If a digital waveform graph includes digital data in both drive and compare logic states, by default the compare data appears darker on the plot than the drive data. If you do not want to darken compare data, right-click the plot and select **Advanced»Darken Compare Data** from the shortcut menu to remove the checkmark. You also can use the Darken Compare Data property to darken compare data programmatically.

Note This feature primarily applies to users generating digital I/O signals. Refer to the **Fundamentals»Graphs and Charts»Concepts»Customizing Graphs and Charts** book on the **Contents** tab of the *LabVIEW Help* for more information about compare data.

Miscellaneous Front Panel Enhancements

LabVIEW 8.2 includes the following miscellaneous enhancements to the front panel.

- In LabVIEW 8.0, the first time you display a caption for a front panel object, LabVIEW moves the label to the side. In LabVIEW 8.2, LabVIEW hides the label and displays only the caption.

- In the **XY Graph Properties** dialog box, the **Show optional plane** pull-down menu and the options to configure the plane you select moved from the **Scales** page to the **Appearance** page.
- You can use the Coloring tool to change the background color of a system table.
- You can change the color of the headers and cells of a system multicolumn listbox, listbox, table, or tree control.
- The **Reinitialize to Default Value**, **Cut Data**, and **Paste Data** options are not available in the shortcut menu of an indicator when the VI is in run mode. These shortcut menu options are available only for controls in run mode.
- To size a tab control to fit its contents, right-click the tab control and select **Advanced»Size To Fit** from the shortcut menu.
- When you right-click an enumerated type control, ring control, or combo box control and select **Edit Items** from the shortcut menu, you must double-click in a cell to edit an item in the **Items** or **Values** column. This change also applies when you right-click the text labels for a slide control or knob and select **Edit Items** from the shortcut menu.
- In LabVIEW 8.0, when you select **File»Apply Changes** from the Control Editor window, LabVIEW preserves the label, caption, and value of the original control. In LabVIEW 8.2, LabVIEW applies all the changes you make in the Control Editor window and does not preserve any information from the original control. This behavior applies only to custom controls and not to type definitions.
- LabVIEW does not include hidden plots when you autoscale the axes of a graph or chart. If you want to include the hidden plots when you autoscale, make the hidden plots transparent instead. Right-click the plot legend and select **Color** from the shortcut menu to change the color of the plots.
- **(Windows)** Radio buttons and checkboxes in LabVIEW are consistent with the behavior of radio buttons and checkboxes in Windows. You can toggle radio buttons and checkboxes only using the spacebar. You can toggle dialog box buttons with the <Enter> key on the alphanumeric keyboard, the <Enter> key on the numeric keypad, or the spacebar. The LabVIEW keyboard shortcuts <T> and <F> do not toggle dialog box buttons.
- LabVIEW 8.2 includes new tokens for advanced editing of the time stamp control. Right-click the control and select **Properties** from the shortcut menu to display the **Time Stamp Properties** dialog box. On the **Format and Precision** page, select the **Advanced editing mode**

option to display the **Absolute time format codes** list. LabVIEW 8.2 includes following new format codes.

Format Code	Value
<%^<>T>	Universal time container
<%z>	Difference between locale time and universal time

Environment Enhancements

LabVIEW 8.2 introduces the following enhancements to the LabVIEW environment.

Automatic Saving for Recovery

In the event of an irregular shutdown or system failure, LabVIEW backs up any modified VI (`.vi`), VI template (`.vit`), control (`.ctl`), or control template (`.ctt`) files open at the time of the shutdown or failure to a temporary location. LabVIEW does not back up projects (`.lvproj`), project libraries (`.lvlib`), XControls (`.xctl`), or LabVIEW classes (`.lvclass`).

If LabVIEW automatically saves files before an irregular shutdown or system failure, the **Select Files to Recover** window appears the next time you launch LabVIEW. Select the files you want to recover and click the **Recover** button. If you do not want to recover any files, deselect all files and click the **Discard** button. Click the **Cancel** button to move all selected files to the `LVAutoSave\archives` subdirectory of the default data directory.

Select **Tools»Options** and select **Environment** from the **Category** list to enable or disable saving for recovery and to specify how often LabVIEW should back up files.

Dialog Box Enhancements

LabVIEW 8.2 includes the following dialog box enhancements.

Options Dialog Box Enhancements and Changes

LabVIEW 8.2 includes the following **Options** dialog box enhancements. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `ex6rkc` for information about workarounds for these deprecated options.

- The **Performance and Disk** page no longer exists, including the **Check available disk space during launch** and **Run with multiple threads** options.

- On the **Front Panel** page, the **Override system default function key settings** and the **Use smooth updates during drawing** options no longer exists.
- On the **Block Diagram** page, the **Show wiring guides** option no longer exists.
- On the **Paths** page, the **Library Directory** option in the pull-down menu no longer exists.
- On the **Environment Options** page, the **Use abridged menus** option no longer exists.
- On the **Environment** page, the **Enable Just-In-Time Advice** checkbox no longer contains a checkmark by default.
- On the **Block Diagram Options** page, the **Show subVI names when dropped** checkbox also applies to global variables. Make sure this checkbox contains a checkmark if you want to display the label of the global variable when you place it on the block diagram.
- On the **Web Server: Visible VIs** page, the **Currently selected VI** option changed to the **Visible VI** option.

LLB Manager Enhancements

LabVIEW 8.2 includes the following enhancements to the **LLB Manager** window.

- A **Browse** button appears to the right of the **Directory** field. Click this button to navigate to the LLB you want to modify.
- A **Delete** button appears in the toolbar. Select a file in the **Files** list and click this button to remove the selected file from the LLB.
- You can sort by column in the **Files** list by right-clicking the columns.

Data Binding Page

LabVIEW 8.2 includes the following enhancements to the **Data Binding** page of the **Properties** dialog box of all front panel controls:

- The **Current Network Item Selected** and **Current Project Item Selected** text boxes changed to the **Path** text box.
- The **Mode** section changed to the **Access Type** pull-down menu.
- The **Blink while Alarm On** checkbox no longer appears.
- The **Browse** button that appears when you select **Shared Variable Engine (NI-PSP)** from the **Data Binding Selection** pull-down menu launches the **Select Source Item** dialog box instead of the **Select Network Item** dialog box. The **Select Network Item** dialog box no longer exists.
- The **Network-Published Source** pull-down menu appears in the **Select Source Item** dialog box instead of on the **Data Binding** page.

Miscellaneous Dialog Box Enhancements

LabVIEW 8.2 includes the following miscellaneous dialog box changes:

- The **VI Metrics** window includes a row in the **Metrics Statistics** table to display the average of all the values in a column. You also can exclude files located in a specific folder by placing a checkmark in the **Exclude files in this folder from statistics** checkbox.
- In the **VI Properties** dialog box, the **Security** page changed to the **Protection** page.
- The **Polymorphic VI** window includes a **Show License Warning** button that appears if the polymorphic VI belongs to a licensed project library where the license is either in evaluation mode or invalid. Click this button to display a warning message. Click the **Help** button in the warning message for more information about the license status.
- In the **Edit Format String** dialog box and **Edit Scan String** dialog box, you can select SI notation from the **Selected operation** pull-down menu as the conversion type.
- In the **Export Simplified Image** dialog box, the **Save to clipboard** option changed to **Export to clipboard**.
- In the **Create Instrument Driver VI** dialog box, **None** is no longer an option in the **Output data type** pull-down menu on the **Control setup** page.
- In the **Error Code File Editor** window, the **Current Error Code and Description** option changed to the **Error code** and **Error text** options.

Displaying Hidden Controls and Indicators

You can display all hidden controls and indicators on the front panel of custom controls and global variables by selecting **Edit»Show Hidden Controls and Indicators**. This option is available only in the **Edit** menu of a custom control or global variable.

You also can display controls and indicators for VIs that are not custom controls or global variables by running the ShowHidden Core VI in the `labview\project\_ShowHidden` directory.

Error List Window Enhancements

In the **Items with errors** list in the **Error list** window, broken items appear with a red `X` glyph beside the item name. LabVIEW sorts these items to the top of the **Items with errors** list. Items that cause errors in other items because you are editing them appear with a pencil icon beside the item name. Items that appear with no icons beside the item name have errors because an item on which they depend is has errors.

Find and Replace Enhancements

LabVIEW 8.2 includes the following enhancements to the find and replace operations and related functionality.

- In the **Text Search Options**, the **Ignore clones** checkbox ignores any front panel clones, or reentrant front panels, when you search for text in a VI.
- In the **Find** dialog box, the **Select Object** button appears with the title of the object you select instead of the filename of the object.
- The **Search Results Window** appears only if LabVIEW finds more than one object during a search. If LabVIEW finds only one object, LabVIEW highlights the object on the front panel window or block diagram window.
- The **Find** dialog box also finds and replaces Express VIs and nodes such as the Match Regular Expression function and variables.

Managing Open Windows

The **Window** menu displays a maximum of 10 open windows. You can manage all open windows by selecting **Window»All Windows** or by pressing <Ctrl-Shift-W> to display the **All Windows** dialog box. You can show or close a window or save the item that corresponds to the window by clicking the corresponding button on the right side of the dialog box.

Window items modified since you last saved them have an asterisk at the end of the corresponding window name in the **Title** column.

Palette Enhancements

LabVIEW 8.2 includes the following palette enhancements:

- You can use the **Organize Favorites** dialog box to change the order of the items in the **Favorites** palette category.
- The **Menu Documentation** dialog box changed to the **Palette Documentation** dialog box. The **Menu Description** field changed to **Palette Description**.
- If a palette belongs to a project library, you can view the path to the project library. Select **Tools»Advanced»Edit Palette Set**, right-click a palette and select **Display Path To Palette File** from the shortcut menu. LabVIEW displays the actual path of the palette and the path to the owning library, if the palette belongs to a library.
- The **Edit Controls and Functions Palette Set** dialog box includes the **Preview changes before saving** checkbox. Place a checkmark in the **Preview changes before saving** checkbox and click the **Save Changes** button to display the **Preview Palette Changes** dialog box.

- The **Drop VI** shortcut menu item changed to **Place VI**. This shortcut item appears if you right-click a VI on the **Functions** palette.
- LabVIEW 8.2 includes palettes that remain empty until you install an add-on. For example, the **Control Design & Simulation** subpalette on the **Controls** palette is empty until you install one of the LabVIEW Control Design & Simulation products.

Saving a Duplicate Hierarchy

You can save a VI and its subVIs as a duplicate hierarchy without having to create a source distribution. Select **File»Save As** and select the **Duplicate hierarchy to new location** option to save the VI and its hierarchy to a new location.

Miscellaneous Environment Enhancements

LabVIEW 8.2 includes the following miscellaneous enhancements to the front panel.

- LabVIEW no longer displays a watermark on the subVIs in a subpanel of a VI in the Evaluation or student versions of LabVIEW. LabVIEW also no longer displays a watermark on the debug deployment version of LabVIEW.
- If you do not have a valid license specific to the version of LabVIEW that you purchased, you might not be able to edit polymorphic VIs.
- You can create a reference to a control in a strict type definition by right-clicking the control in the type definition and selecting **Create»Reference** from the shortcut menu.
- The **View** menu includes the **This VI's Library**, **This VI's XControl**, or **This VI's Class** item which highlights the project library, XControl, or LabVIEW class to which the current VI belongs in the **Project Explorer** window. The menu item changes according to the type of library that owns the VI. If the library, XControl, or class is not in a LabVIEW project, LabVIEW opens a new window that contains only the library, XControl, or class to which the current VI belongs.
- In LabVIEW 8.0, the **Help»Explain Errors** menu item is not available in the **Getting Started** window. In LabVIEW 8.2 this menu item is available in the **Getting Started** window.
- **(Mac OS)** To display or hide the **Context Help** window, select **Help» Show Context Help** or press the <Command-Shift-H> keys. The <Command-H> keyboard shortcut hides the open LabVIEW application. Refer to the *LabVIEW Help* for more information about customizing keyboard shortcuts.

- Select **Tools»Instrumentation»Advanced Development** to access options for advanced development in LabVIEW using instrument drivers, including:
  - **Show Driver Guidelines** displays the Instrument Driver Guidelines from the National Instruments Instrument Driver Network at `ni.com/idnet` in a Web browser.
  - **Show Icon Art Glossary** displays the Icon Art Glossary from the National Instruments Instrument Driver Network at `ni.com/idnet` in a Web browser.
  - **Other Resources** displays Development Tools and Resources from the National Instruments Instrument Driver Network at `ni.com/idnet` in a Web browser.

New and Changed VI, Function, and Node Enhancements

LabVIEW 8.2 includes the following new and changed VIs and functions. 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 and Functions

LabVIEW 8.2 includes the following new VIs and functions.

Advanced File VIs

The **Advanced File Functions** palette includes the following new VIs:

- Check if File or Folder Exists VI
- Compare Two Paths VI
- Generate Temporary File Path VI
- Get File Extension VI
- MD5Checksum File VI
- Recursive File List VI

Digital Waveform Functions

The **Digital Waveform** palette includes the following new functions:

- Build Waveform function
- Build Digital Data function
- Get Waveform Components function
- Get Digital Data Components function

Mathematics VIs

The **Mathematics** palette includes the following new VIs in the LabVIEW Full and Professional Development Systems:

- Kronecker Product VI
- Lyapunov Equations VI

.NET VIs

The **.NET** palette includes the following new VIs:

- To .NET Object VI
- .NET Object To Variant VI

Scaling VIs

The **Scaling** palette includes the following new VIs:

- Convert RTD Reading VI
- Convert Strain Gauge Reading VI
- Convert Thermistor Reading VI
- Convert Thermocouple Reading VI

Signal Processing VIs

The **Signal Processing** palette includes the following new VIs in the LabVIEW Full and Professional Development Systems:

- Bohman Window VI
- Decimate (continuous) VI
- Decimate (single shot) VI
- FIR Filter VI
- FIR Filter with I.C. VI
- Gaussian Modulated Sine Pattern VI
- Gaussian Monopulse VI
- Inverse Chirp Z Transform VI
- Modified Bartlett-Hanning Window VI
- Parzen Window VI
- Periodic Sinc Pattern VI
- Pulse Train VI
- Rational Resample VI
- Savitzky-Golay Filter VI
- Savitzky-Golay Filter Coefficients VI

- Triangle Pattern VI
- Upsample VI
- Welch Window VI

TDM Streaming VIs and Functions

The **TDM Streaming** palette includes the following new VI and functions:

- TDM Streaming File Viewer VI
- TDMS Close function
- TDMS Defragment function
- TDMS Flush function
- TDMS Get Propertiesfunction
- TDMS List Contents function
- TDMS Open function
- TDMS Read function
- TDMS Set Properties function
- TDMS Write function

The **Storage** palette includes the following new VIs:

- Convert TDM to TDMS VI
- Convert TDMS to TDM VI

Refer to the *TDM Streaming File Format* section of this document for information about the TDM streaming file format.

Changed VIs, Functions, and Nodes

The following VIs, functions, and nodes changed in LabVIEW 8.2.

DataSocket functions

The **DataSocket** palette includes the following changed VIs:

- You can enable synchronous notifications for NI-PSP data items when you use the DataSocket Write function. By appending `?sync="true"` to the end of the `psp` URL, when you enable synchronous notifications, you can specify a nonzero **ms timeout** value. The function waits until the operation completes or the timeout expires. Enabling synchronous notifications can cause slower performance, particularly on RT targets.
- The DataSocket Read function has a **status** output, which reports warnings or errors from a PSP server or FieldPoint controller.

Mathematics VIs

In the LabVIEW Base Package, the A x B VI includes the new instances Vector x A and Complex Vector x A.

The **Mathematics** palette includes the following changed VIs in the LabVIEW Full and Professional Development Systems:

- The Exponential Fit VI includes a new **Weight** input that specifies the array of weights for the observations (**X**, **Y**). The negative values that often result from signal noise no longer cause the exponential fit to fail.
- The GCD of p(x) and q(x) VI includes a new **algorithm** input that specifies the algorithm the VI uses to compute the polynomial greatest common divisor.
- The General Polynomial Fit VI includes a new **Weight** input that specifies the array of weights for the observations (**X**, **Y**).
- The Histogram, Histogram PtByPt, General Histogram, and General Histogram PtByPt VIs include a **Histogram Graph** output that displays the bar graph of the histogram of the input sequence **X**. The *y*-axis is the histogram count, and the *x*-axis is the histogram center values of the intervals (bins) of the histogram.
- The LCM of p(x) and q(x) VI includes a new **algorithm** input that specifies the algorithm the VI uses to compute the polynomial least common multiple.
- The Partial Fraction Expansion VI includes a new **option** input that specifies how the VI handles the co-factors of **Numerator** and **Denominator**.
- The Sylvester Equations VI includes a new **matrix type** input that specifies the types of inputs **A** and **B**, which speeds up the computation of **X**.
- The **Direction Cosines** input of the Array instance of the 3D Cartesian Coordinate Rotation (Direction) VI changed to **Rotation Matrix**.

Numeric Functions

The **Numeric** palette includes the following changed functions:

- The Quotient & Remainder function produces accurate answers when you use large negative 64-bit divisors.
- The **anything** input of the Swap Bytes and Swap Words functions changed to **data**. You can wire string, tag, path, Boolean, non-integer numeric, enumerated type, error cluster, picture, matrix, and array and cluster data of such types and you can pass the data unchanged. For any 16-, 32-, and 64-bit integers wired to the input the function swaps every byte pair in each word or every word pair in each longword. You cannot wire occurrences, all refnum types, or variants to the input.

- On x86-based platforms, the Scale By Power Of 2 function produces the correct answer when both the **x** and **n** inputs are floating-point numeric.

Protocols VIs and Functions

The **Protocols** palette includes the following changed VIs and functions:

- The UDP Open function and UDP Multicast Open VI have a new **port** output that returns the port number the function used.
- The UDP Open function and UDP Multicast Open VI have a new **net address** input, on which network address to listen.
- The **address** input of the TCP Open Connection function changed from optional to recommended.
- The TCP Listen VI includes a **resolve remote address** input that indicates whether to call the IP to String function on the remote address. The default is TRUE.

Quadratic Programming VI

The Quadratic Programming VI includes the following changes:

- The Quadratic Programming VI is a polymorphic VI with the following instances: Quadratic Programming IP and Quadratic Programming AS. The Quadratic Programming IP instance has the same functionality as the Quadratic Programming VI in LabVIEW 8.0. The Quadratic Programming AS instance finds the minimum using an active set algorithm.
- Both instances of the Quadratic Programming VI include a **start** input that specifies the point in *n* dimension at which the optimization process starts.
- Both instances of the Quadratic Programming VI include a **max time (sec)** input that specifies the maximum amount of time LabVIEW allows between the start and the end of the optimization process.
- The Quadratic Programming AS instance of the Quadratic Programming VI includes a **warm start?** input that indicates whether to allow a warm start of the optimization.

Signal Processing VIs

The **Signal Processing** palette includes the following changed VIs in the LabVIEW Full and Professional Development Systems:

- The Median Filter VI includes new **left rank** and **right rank** inputs that replace the **rank** input in LabVIEW 8.0. This VI applies a median filter of rank to the input sequence **X**, where rank is **right rank** if **right rank** is greater than zero, or **left rank** if **right rank** is less than zero.

Note If you open a LabVIEW 8.0 or earlier VI that includes the Median Filter VI with a value wired to the **rank** input, LabVIEW uses that value for **left rank** in LabVIEW 8.2.

- The **f** input of the Sine Wave, Sine Wave PtByPt, Triangle Wave, Triangle Wave PtByPt, Sawtooth Wave, Sawtooth Wave PtByPt, Square Wave, Square Wave PtByPt, and Arbitrary Wave VIs changed to **frequency**.
- The **# side points** input of the Savitzky Golay Filter PtByPt VI changed to **side points**.
- The Point By Point VIs have updated icons.
- The **window** input of the Symmetric Window and the Scaled Time Domain Window VIs includes the following new values: Modified Bartlett-Hanning, Bohman, Parzen, and Welch.

Sound VIs (Linux)

You must have the Open Sound System (OSS) driver to use the **Sound** VIs. Refer to the **Important Information»Copyright** book on the **Contents** tab of the *LabVIEW Help* for applicable OSS copyright information.

Note LabVIEW probes for devices by looking for files named `/dev/dsp` or `/dev/dsp`*`X`*, where *`X`* is an integer between `0` and `16`. LabVIEW attempts to open each device for input and output. If LabVIEW cannot detect the sound card, check that a device file named `/dev/dsp` or `/dev/dsp`*`X`* exists on the local system and that you have permission to read from and write to the device. If you moved this device to a location other than the default, LabVIEW can work with a symbolic link.

The Sound VIs on Linux include the following changes:

- The VIs support monophonic and stereophonic sound.
- A waveform represents sound data. You can use elements of 8-bit unsigned, 16-bit signed, or 32-bit integers signed, or single and double-precision data types to represent the **Y** array data. Each waveform defines one channel.
- The format of the sound data is Pulse Code Modulated (PCM).
- The VIs can produce continuous sound output.
- The VIs allow for a streaming view of wave files.
- The VIs have improvements to error checking.

String Functions

The **String** palette includes the following changed functions:

- The **format string** input of the Scan From String function accepts System International (SI) notation values.

- The Spreadsheet String to Array function works correctly when the **array type** input is complex numeric.

VISA Functions

When you right-click terminals of certain VISA functions and select **Create»Constant**, **Create»Control**, or **Create»Indicator** from the shortcut menu, a ring object appears. The object appears as a pull-down menu that you can cycle through to make selections. All values that LabVIEW supports appear in the pull-down menus.

This change affects the following functions and terminals.

Function	Terminal(s)
VISA Assert Interrupt Signal	mode
VISA Assert Trigger	protocol
VISA Assert Utility Signal	bus signal
VISA Disable Event	event type
VISA Discard Events	event type
VISA Enable Event	event type
VISA Find Resource	search mode
VISA GPIB Control ATN	mode
VISA GPIB Control REN	mode
VISA Map Trigger	trigger source, trigger destination
VISA Open	access mode
VISA Unmap Trigger	trigger source, trigger destination
VISA VXI Cmd or Query	mode
VISA Wait on Event	event type in, event type out

Call Library Function Node

The Call Library Function Node includes error terminals. The **Call Library Function** dialog box also includes the following changes:

- The dialog box includes multiple pages, which you can use to configure parameters more easily.
- On the **Function** page, you can configure the node to allow you to specify the library path programmatically by placing a checkmark in the **Specify path on diagram** checkbox.

- On the **Parameters** page, when you configure string or array parameters, you can use the **Minimum size** text box to allocate memory correctly.
- On the **Callbacks** page, you can specify user-defined callbacks.

Miscellaneous VI, Function, and Node Changes

LabVIEW 8.2 includes the following miscellaneous VI, function, and node changes:

- The **Mathematical Operation** options in the configuration dialog box of the Time Domain Math Express VI changed from **Differential** to **Derivative (dX/dt)**, from **Difference** to **Difference (dX)**, from **Integral** to **Integral (Sum[Xdt])**, and from **Summation** to **Summation (Sum[X])**.
- **(Windows)** The **refnum in** input of the Sound File Info (refnum) instance of the Sound File Info VI changed to **sound file refnum**.
- **(Mac OS, Linux)** The Set Report Footer Text VI generates footers in a slightly different size than in previous versions of LabVIEW. The Set Report Footer Text VI also includes a new **HTML footer size** input.
- The Synchronize Timed Structure Starts VI includes a **clear** input that removes all timed structures and deletes the entire group before adding the timed structures you specify to the group. Use this input to remove any timed structures that do not correspond to a Timed Loop. This VI also includes a **timed structures names out** output that returns the names of all timed structures in the group after LabVIEW adds the names to the synchronization group.
- **(Windows)** In LabVIEW 8.0, the List Folder function appends `.*` to the **pattern** input if **pattern** does not already include an extension. In LabVIEW 8.2, the function does not alter the **pattern** input, but it appends `.` to the filename if the filename does not have an extension but **pattern** does have an extension.
- After you place the Static VI Reference function on a block diagram, double-click the function to display a file dialog box where you can select a VI. In LabVIEW 8.0, double-clicking the function displayed a missing file error dialog box.
- The **format string** input of the Format Into File, Format Into String, Scan From File, and Scan From String functions is no longer a required input.
- Opening a VI containing a Shared Variable node in a project where the Shared Variable node cannot find its associated shared variable in the **Project Explorer** window causes the Shared Variable node to break. Any front panel controls associated with the missing shared variable also break. This behavior is specific to Windows, and only occurs when you open the VI node in a project. If you open the VI in the main

application instance, you do not receive notification of missing shared variables. Previous versions of LabVIEW did not indicate that the Shared Variable node could not find its associate variables in the Project Explorer window.

- In LabVIEW 8.0.1, the **Wakeup Reason** output of the Timed Loop is a ring. In LabVIEW 8.2, the output is an enumerated type, which is consistent with LabVIEW 8.0 behavior.
- The names of the duplicate outputs of the following VIs and functions changed from **dup [output]** to **[output] out**. This change does not impact the functionality of these VIs and functions:
  - Call By Reference Node
  - Execute Query Expression
  - File/Directory Info
  - Get Object Info
  - Get Property
  - Invoke Node
  - List Folder
  - Property Node
  - Scan String For Tokens
  - Refnum to ID
  - Set Property

New Properties, Methods, and Events

LabVIEW 8.2 includes new VI server classes, properties, methods, and events. Refer to the **LabVIEW 8.2 Features and Changes»New VI Server Classes, Properties, Methods, and Events** book on the **Contents** tab of the *LabVIEW Help* for a list of new class, properties, methods, and events.

LabVIEW 8.2 also includes the following new VISA properties: PXI/PCI Settings:Is PCI Express, PXI/PCI Settings:Maximum Link Width, PXI/PCI Settings:Actual Link Width, PXI/PCI Settings:Slot Link Width, PXI/PCI Settings:D-Star Bus Number, and PXI/PCI Settings:D-Star Set.

LabVIEW MathScript Enhancements

Refer to the **Fundamentals»Formulas and Equations** book on the **Contents** tab in the *LabVIEW Help* for more information about LabVIEW MathScript.

LabVIEW 8.2 introduces the following enhancements and changes to MathScript.

New MathScript Functions

LabVIEW 8.2 includes the following new MathScript functions. You can use these functions in the **LabVIEW MathScript Window** or the MathScript Node.

- plots class: `area`, `bar`, `bar3`, `bar3h`, `barh`, `contour`, `contour3`, `contourf`, `errorbar`, `ezcontour`, `ezcontourf`, `ezmesh`, `ezmeshc`, `ezplot`, `ezplot3`, `ezpolar`, `ezsurf`, `ezsurfc`, `feather`, `fill`, `fplot`, `gplot`, `meshc`, `pie`, `plotmatrix`, `plotyy`, `polar`, `quiver`, `scatter`, `scatter3`, `shg`, `stem3`, `strips`, `surfc`, `treeplot`, and `waterfall`.
- dsp class: `ac2poly`, `ac2rc`, `arburg`, `arcov`, `armcov`, `aryule`, `barthannwin`, `bartlett`, `besselap`, `besself`, `bilinear`, `bitrevorder`, `blackman`, `blackmanharris`, `bohmanwin`, `buttap`, `cceps`, `cheb1ap`, `cheb2ap`, `chebwin`, `chirp`, `conv2`, `convmtx`, `corrmtx`, `czt`, `dct`, `dftmtx`, `digitrevorder`, `diric`, `downsample`, `dst`, `ellipap`, `eqtflength`, `filternorm`, `filtic`, `firgauss`, `firrcos`, `flattopwin`, `freqspace`, `gauspuls`, `gausswin`, `gmonopuls`, `goertzel`, `hann`, `icceps`, `iczt`, `idct`, `idst`, `impinvar`, `intfilt`, `invfreqs`, `invfreqz`, `is2rc`, `kaiserord`, `lar2rc`, `latc2tf`, `levinson`, `lp2bp`, `lp2bs`, `lp2hp`, `lp2lp`, `lpc`, `lsf2poly`, `maxflat`, `medfilt1`, `nuttallwin`, `parzenwin`, `phasedelay`, `phasez`, `poly2ac`, `poly2lsf`, `poly2rc`, `polyscale`, `polystab`, `prony`, `pulstran`, `rc2ac`, `rc2is`, `rc2lar`, `rc2poly`, `rceps`, `rectpuls`, `rectwin`, `resample`, `residuez`, `rlevinson`, `schurrc`, `seqperiod`, `sgolay`, `sgolayfilt`, `sinc`, `sos2ss`, `sos2tf`, `sos2zp`, `sosfilt`, `spline`, `ss2sos`, `ss2tf`, `ss2zp`, `stepz`, `stmcb`, `tf2latc`, `tf2sos`, `tf2ss`, `tf2zp`, `tf2zpk`, `triang`, `tripuls`, `tukeywin`, `udecode`, `uencode`, `upfirdn`, `upsample`, `vco`, `xcorr`, `xcorr2`, `xcov`, `zerophase`, `zp2sos`, `zp2ss`, and `zp2tf`.
- support class: `csvread`, `csvwrite`, `dlmread`, `dlmwrite`, `labviewroot`, `type`, `uiload`, and `what`.
- string class: `eval`.
- libraries class: `loadlibrary`, `calllib`, `unloadlibrary`, `libisloaded`, and `libfunctionsview`. You can use these functions to call shared libraries from the **LabVIEW MathScript Window** or the MathScript Node. Refer to the `MathScript Shared Libraries.lvproj` in the `labview\examples\MathScript\MathScript Shared Libraries` directory for examples of calling shared libraries from MathScript.

Miscellaneous MathScript Enhancements and Changes

LabVIEW 8.2 includes the following miscellaneous changes to MathScript:

- Overall performance improved in the **LabVIEW MathScript Window**. Compile time improved in the MathScript Node.
- The LabVIEW Run-Time Engine supports MathScript. You can include MathScript Nodes in stand-alone applications and shared libraries that you build with the Application Builder. The LabVIEW Run-Time Engine currently does not support some MathScript functions. If a script includes these unsupported functions, you might need to modify the script before you build an application or shared library. Refer to the *MathScript Functions Not Supported in the LabVIEW Run-Time Engine* topic in the *LabVIEW Help* for a complete list of unsupported MathScript functions.
- When you call the MathScript `help` command, LabVIEW displays help in an HTML Help window. To display help in the **Output Window** of the **LabVIEW MathScript Window**, select **File» MathScript Preferences** and remove the checkmark from the **Display HTML Help?** checkbox. If you define a function, LabVIEW always displays the help for the function you defined in the **Output Window**.
- The MathScript Node behaves differently than other script nodes when you wire an unsupported data type to an input terminal. On a MathScript Node, LabVIEW either converts the data type to a supported type or displays a broken wire. If LabVIEW converts the data type, a coercion dot appears on the terminal where the conversion takes place. After you wire an input to a MathScript Node, right-click the input terminal and select **Show Data Type** from the shortcut menu to see the data type of the input.
- MathScript supports all non-Unicode characters in text strings but not in variable names. You can use only ASCII characters in variable names. For example, you can use `á` in a text string, but you cannot call the following script from the **LabVIEW MathScript Window** or the MathScript Node:

  ```
  á=rand(50, 1)
  plot(á)
  ```

  You can save data to paths that contain non-Unicode characters. Also, if you install LabVIEW in a directory whose path contains any non-Unicode characters, MathScript functions correctly.
- LabVIEW uses short-circuit evaluation to evaluate compound logical expressions in MathScript. For example, if you execute the command `if 0 == 0 || foo(a) == 2`, LabVIEW does not execute `foo(a)` because the first part of the expression already is TRUE. Similarly, if

you execute the command `if 0 ~= 0 && foo(a) == 2`, LabVIEW does not execute `foo(a)` because the first part of the expression already is FALSE.

Because LabVIEW 8.0 evaluates all parts of compound logical expressions in MathScript regardless of whether the expressions are TRUE or FALSE, LabVIEW 8.0 scripts that contain compound logical expressions might not run as expected in LabVIEW 8.2. For example, because LabVIEW 8.0 executes `foo(a)` in the command `if 0 == 0 || foo(a) == 2`, you can use the result of `foo(a)` to define a variable in the/a script. This same script executes differently in LabVIEW 8.2. Because LabVIEW 8.2 does not execute `foo(a)` in the command `if 0 == 0 || foo(a) == 2`, you cannot use the result of `foo(a)` in the/a script. If you do not want LabVIEW to use short-circuit evaluation, remove compound logical expressions from existing code.

- MathScript supports the `nargin` and `nargout` functions within user-defined functions.
- MathScript supports the `return` keyword. MathScript also supports the `end` keyword in matrix indexing.
- The `prod` and `sum` functions include a **b** input that you can use to specify the dimension along which to compute the product or sum.

3D Picture Control

Refer to the **Fundamentals»Graphics and Sound VIs** book on the **Contents** tab in the *LabVIEW Help* for more information about the 3D picture control.

The 3D Picture Control VIs, properties, and methods convert a collection of 3D objects into a 3D scene that you can view and manipulate. You can generate multiple 3D objects and specify their size, shape, movement, appearance, and relationship to other objects within the scene.

Use the following VIs, properties, and methods to specify the appearance of a 3D object:

- Use the Object VIs to create or find 3D objects. You also can use the SceneObject properties and SceneObject methods to place 3D objects in the scene and assign the objects characteristics programmatically.
- Use the File Loading VIs to add existing model or scene files to the 3D scene.
- Use the Geometries VIs with the SceneGeometry properties and SceneGeometry methods to specify the geometric form a 3D object takes.
- Use the Transformations VIs to position objects in a 3D scene.

Use the Helpers VIs to perform common 3D scene operations. You can configure a separate window for the scene, create new clip planes within the scene, add light sources, apply textures to 3D objects, and convert LabVIEW color values to appear in the 3D picture control.

You also can use the following properties and methods to configure a 3D scene programmatically:

- Use the SceneWindow properties to render the scene in a separate window, configure the window, and set the interaction of the camera controller with the scene.
- Use the SceneClipPlane properties to specify planes in the scene in which an object appears or is cut off.
- Use the SceneLight properties to configure a light source for the scene.
- Use the SceneTexture properties and SceneTexture methods to apply textures to a 3D object.

Refer to the solarsystem VI in the `labview\examples\picture\3D Picture Control` directory for an example of creating a 3D scene with the 3D picture control.

LabVIEW Object-Oriented Programming

Refer to the **Fundamentals»LabVIEW Object-Oriented Programming** book on the **Contents** tab in the *LabVIEW Help* for more information about object-oriented programming in LabVIEW.

LabVIEW object-oriented programming uses concepts from other object-oriented programming languages such as C++ and Java, including class structure, encapsulation, and inheritance. You can use these concepts to create code that is easier to maintain and modify without affecting other sections of code within the application. You can use object-oriented programming in LabVIEW to create user-defined data types.

Creating LabVIEW Classes

You create user-defined data types in LabVIEW by creating LabVIEW classes. LabVIEW classes define data associated with an object, as well as the methods that define the actions you can perform on the data.

In LabVIEW, the data of a class is private, which means only VIs that are members of the class can access the data. You define the data of the class in the private data control. When you create and save a LabVIEW class, LabVIEW creates a class library file (`.lvclass`) that defines a new data type. The class library file records the private data control and information about any member VIs you create, such as a list of the VIs and various properties of the VIs. The class library is similar to the project library (`.lvlib`). However, the class library defines a new data type.

You can create a class in one of the following ways:

- Right-click **My Computer** in the **Project Explorer** window and select **New»Class** from the shortcut menu.
- Select **File»New** to display the **New** dialog box and select **Other Files»Class** from the **Create New** list.

Defining Private Data Controls

LabVIEW creates a private data control of the class automatically when you create a LabVIEW class.

You use the Control Editor window to customize the private data control of a class. LabVIEW displays the Control Editor window when you double-click the private data control of the class in the **Project Explorer** window. You can place controls and indicators in the **Cluster of class private data** to define the private data type of a LabVIEW class. The default values you set for the controls in the **Cluster of class private data** are the default values for the class.

Creating Member VIs

Member VIs implement the methods you create for the LabVIEW class. You create member VIs to perform operations on the private data of the class. Member VIs are members of the LabVIEW class in which you create them and appear in the **Project Explorer** window under the private data control of the class. You can define most methods using a single member VI in one class, but some methods you might define by creating multiple member VIs throughout the class hierarchy.

You can create a member VI from a VI template that includes error handling and class objects, from a blank VI, or from an ancestor member VI. Right-click the class and select among the following shortcut menu items:

- **New»VI**—Opens a blank member VI.
- **New»Dynamic VI**—LabVIEW populates the new member VI with **error in** and **error out** clusters, a Case structure for error handling, the input LabVIEW class, and the output LabVIEW class.
- **New»Override VI**—Creates a member VI that overrides an ancestor member VI.

Distributing a LabVIEW Class to Other Developers and Users

You can distribute the LabVIEW class you develop to other LabVIEW class developers and LabVIEW class users. You can distribute the class in several ways so choose the manner that most suits your needs. You can use the Application Builder to create a zip file to distribute the class or classes. You also can lock the LabVIEW class before you distribute it to limit the access the LabVIEW class user has to the private data and member VIs. Locking the class can help prevent users from introducing errors in the application.

LabVIEW Project Enhancements

LabVIEW 8.2 includes the following enhancements to the LabVIEW project and related functionality:

Application Builder Enhancements

LabVIEW 8.2 includes the following enhancements to the Application Builder in the LabVIEW Professional Development System:

- **Automatically incrementing product version**—If you build an installer multiple times, LabVIEW can increment the product version number automatically for each new version of the installer. The **Auto increment product version** checkbox appears on the **Product Information** page of the **Installer Properties** dialog box.
- **Specifying media size for storing the installer**—You can customize media size when saving installer components. The **Enable media spanning** checkbox appears on the **Advanced** page of the **Installer Properties** dialog box. The pull-down menu of media includes a **Custom** option. Use this option to enter an arbitrary media size value in the **Media size (MB)** text box.
- **Requiring Windows 2000 or later**—You can require that users have Microsoft Windows 2000 Service Pack 3 or a later version to run the installer. The **Windows 2000 or later** option appears in the **System Requirements** section of the **Advanced** page of the **Installer Properties** dialog box.
- **Caching installer components**—If you build an installer more than once and the installer contains additional installers or components, caching eliminates the need to specify a location for the additional components each time you build the installer. The first time you build the installer, the **Locate Distribution** dialog box prompts you to locate the distribution that contains the additional components. Place a checkmark in the **Cache component from this distribution** checkbox to copy files from the distribution into a permanent location on the local system. The next time you build an installer that includes these

components, the Application Builder automatically copies the components from the local system instead of prompting you for a distribution CD.

- **Dynamic VIs and dependencies**—If a VI specified as a dynamic VI on the **Source File Settings (Application)** page is going to a destination other than the built application, Application Builder moves all dependencies of the dynamic VI to the new destination with the dynamic VI, rather than keep the dependencies in the built application. If two or more dynamic or top-level VIs call a VI and try to move it to two different locations, Application Builder moves the VI and all subVIs to the built application. To make a VI that is specified as **Include only if referenced** on the **Source File Settings** page move to a new location, you must specify the VI as a dynamic VI.
- **Using project alias files with shared libraries**—The **Advanced** page of the **Shared Library Properties** dialog box includes the **Use the default project alias file** checkbox, which associates the project alias file with the shared library. If you remove the checkmark from the checkbox, specify an alias file to use in the **Alias file in project** textbox.
- **Including additional LabVIEW header files with shared libraries**—The **Advanced** page of the **Shared Library Properties** dialog box includes the **Include additional LabVIEW header files** checkbox, which copies any additional LabVIEW header files that the header file, generated during the builder process, references. You can include additional header files to use a LabVIEW built shared library in C or another language that requires those header files.
- The **New Destination** button on the **Distribution Settings** page of the **Source Distribution** dialog box changed to the **Add** button. Also, all options related to excluding files from a source distribution moved to the **Additional Exclusions** page.
- The **New Destination** button on the **Destinations** page of the **Application Properties** dialog box and **Shared Library Properties** dialog box changed to the **Add** button.
- You can build an application or source distribution without saving VIs first.
- If you do not add the `.zip` extension to zip files you create in the Application Builder, LabVIEW adds the extension automatically.

New Dialog Box Pages

LabVIEW 8.2 includes the following new **Application Builder** dialog box pages have been added:

- **Installer Properties** dialog box

  - **Dialog Information**—Use the this page to design the user interface for the installer. You can set the language for the text, display a custom readme and license agreement, and set a welcome title and message. This page replaces the **Dialog information** section on the **Product Information** page in LabVIEW 8.0.

Note The **Include custom license agreement** option of the **Dialog Information** page replaces the **License file** option which appeared on the **Product Information** page.

  - **Hardware Configuration**—Use the this page to specify the source of hardware configuration information to include in the installer. The new **Import Mode** section includes more options for importing hardware configuration files from Measurement & Automation Explorer. This page replaces the **Hardware configuration** section of the **Advanced** page in LabVIEW 8.0.

- **Application Properties**, **Shared Library Properties**, and **Source Distribution Properties** dialog boxes
  - **Additional Exclusions**—Use this page to configure settings for removing or disconnecting type definitions, unused polymorphic VI instances, and unreferenced members of project libraries. Use these settings to reduce the size of the build.

Duplicating and Rearranging Build Specifications

You can duplicate build specifications in the **Project Explorer** window. Right-click the build specification to duplicate and select **Duplicate** from the shortcut menu to create a copy of the build specifications under the **Build Specifications**.

You also can drag and drop build specification items to rearrange the build order within the same **Build Specification**.

Activating the Application Builder (Windows)

If you have an activated version of the LabVIEW Base Package or Full Development System, you can select **Help»Activate Application Builder** to activate the Application Builder. The license takes effect when you restart LabVIEW.

Creating Project Libraries and Adding Shared Variables Programmatically

Use the CreateOrAddLibrary VI in the `labview\vi.lib\Utility\Variable` directory to add a library to a project or a parent item such as a target, folder, or another library programmatically.

You also can use the AddSharedVariableToLibrary VI in the `labview\vi.lib\Utility\Variable` directory to add shared variables to a library programmatically.

Saving LabVIEW Projects and Project Libraries for a Previous Version

You can save LabVIEW projects and project libraries that are readable by LabVIEW 8.0. To save a LabVIEW project for a previous version, select **File»Save for Previous Version** in the **Project Explorer** window. To save a project library for a previous version, right-click the library file in the **Project Explorer** window and select **Save For Previous Version** from the shortcut menu, or open the project library and select **File»Save for Previous Version.**

Selecting an Application Instance

LabVIEW creates an application instance for each target in a LabVIEW project. When you open a VI from the **Project Explorer** window, the VI opens in the application instance for the target. LabVIEW also creates a main application instance, which contains open VIs that are not part of a project and VIs that you did not open from a project. You can open VIs in a specific application instance by using the application instance shortcut menu. Right-click the current instance name in the bottom left corner of the front panel window or block diagram window to display the shortcut menu and select among all application instances. Selecting a new application instance reopens the VI in the selected application instance. The VI also remains open in the original application instance.

Shared Variable Enhancements

LabVIEW 8.2 includes the following shared variable enhancements:

- LabVIEW uses the Network:BuffSize, Network:ElemSize, and Network:PointsPerWaveform properties as appropriate to calculate the network buffer size for a network-published shared variable.
  - For scalar network-published shared variables, LabVIEW uses the Network:BuffSize property, which indicates the number of values to buffer.

  - For array and string network-published shared variables, LabVIEW uses the Network:BuffSize and Network:ElemSize properties.
  - For waveform network-published shared variables, LabVIEW uses the Network:BuffSize and Network:PointsPerWaveform properties.
  - For array of waveform network-published shared variables, LabVIEW uses the Network:BuffSize, Network:ElemSize, and Network:PointsPerWaveform properties.
- The variable input of the Shared Variable node is required when the node is configured to write data.
- The **Variable** page of the **Shared Variable Properties** included the following changes:
  - The **Custom** item in the **Data Type** pull-down menu changed to **From Custom Control**.
  - The **Data Type** pull-down menu includes a new **Variant** item.
  - No longer displays the element size for the network buffer.
  - If you set **Access Type** to **read only** or **write only**, you can create shared variables that are configured only to read data or write data, respectively. When you right-click the shared variable that is bound to a source that is read or write only, LabVIEW disables the **Change to Write** and **Change to Read** options in the shortcut menu.
  - In LabVIEW 8.0, you right-click the Shared Variable node and select **Show timestamp** from the shortcut menu to obtain time stamp information about the single-process shared variable. In LabVIEW 8.2, you must first right-click the shared variable in a project, select **Properties** from the shortcut menu, and place a checkmark in the **Enable timestamp** checkbox on the **Variable** page to obtain time stamp information about the single-process shared variable. To view the time stamp information and add a **timestamp** output to the Shared Variable node, right-click the Shared Variable node and select **Show Timestamp** from the shortcut menu. If you load a single-process shared variable in LabVIEW 8.2 that you created in LabVIEW 8.0, the time stamp is enabled by default.
- In LabVIEW 8.0, LabVIEW dims items you cannot select in the list of the **Select Variable** dialog box. In LabVIEW 8.2, LabVIEW dims the **OK** button if you select an invalid item from the list.

Miscellaneous Project Enhancements

LabVIEW 8.2 includes the following miscellaneous project enhancements:

- In LabVIEW 8.0, if you create a type definition in a project library and set the access scope of the type definition as private, you still can reference the private type definition in a VI outside the project library. In LabVIEW 8.2, you cannot reference private type definitions outside of the library.
- Right-click an XControl and select **New»VI** from the shortcut menu to create a VI inside an XControl.
- In LabVIEW 8.0, you have to save an XControl ability, property, or method VI after you set a breakpoint when debugging. In LabVIEW 8.2, you no longer have to save abilities, properties, or method VIs after you set breakpoints.
- When you enter a password for a library in a LabVIEW project that does not have a valid license specific to the version of LabVIEW you purchased, you cannot drag and drop items into or out of the library.
- If a project library is password protected and the password is not in the LabVIEW password cache, you can right-click the project library and select **Enter Password** from the shortcut menu to unlock the project library.

Controlling VIs Remotely from Multiple Clients

Multiple clients can control an application or VI remotely at the same time. To allow simultaneous control of a VI, the VI must be reentrant. To make a VI reentrant, select **File»VI Properties**, select **Execution** from the **Category** list, and place a checkmark in the **Reentrant execution** checkbox. LabVIEW opens a clone of the reentrant VI for each client request for a remote front panel. You can use the Web Server:VI Access List property to programmatically limit access to clones already in memory for remote front panel connections.

Importing Functions from a Shared Library File

You can generate and update VIs for exported functions in a Windows `.dll` file, a Mac OS `.framework` file, or a Linux `.so` file. Select **Tools»Import»Shared Library** then follow the prompts to create wrapper VIs for shared library files. You must provide the name of a shared library file and a header `.h` file.

Refer to the **Fundamentals»Calling Code Written in Text-Based Programming Languages»Concepts»Importing Functions from Shared Library Files** on the **Contents** tab of the *LabVIEW Help* for more information about creating wrappers for shared library files.

Instrument Driver Templates

LabVIEW 8.2 includes the following new templates for the Create New Instrument Driver wizard:

- **General Purpose (register-based)**—Use for register-based instruments for which there is no class-specific template. Common register-based instruments include: VXI and PXI.
- **Spectrum Analyzer**—Controls basic operations such as setting the frequency range and sweep coupling. The template also includes advanced features such as configuring and querying the marker.

.NET and ActiveX Enhancements (Windows)

User interface thread dependency no longer exists for .NET controls. You do not have to set a VI or subVI to run in the user interface thread for .NET control operations. By default, LabVIEW sets any .NET controls to run in the correct thread, which in most cases is the user interface thread.

LabVIEW does not support debugging in ActiveX and .NET callback VIs. If you use breakpoints in ActiveX and .NET callback VIs, LabVIEW does not pause at the breakpoint.

In LabVIEW 8.0 and earlier, the .NET Refnum Probe displayed only the hex value of the reference. In LabVIEW 8.2, the .NET Refnum Probe displays the result of calling the ToString() method on the object represented by the .NET refnum, the type of the object, the hash code of the object, and the hex value of the reference. You can use this probe to set a breakpoint if the value is an invalid refnum.

LabVIEW 8.2 includes the following changes in the .NET and ActiveX menus:

- The **Tools».NET & ActiveX** menu items, **Add .NET Controls to Palette** and **Add ActiveX Controls to Palette**, changed to**Tools» Import».NET Controls to Palette** and **Tools»Import»ActiveX Controls to Palette**, respectively.
- The **Tools».NET & ActiveX»Browse ActiveX Properties** menu item changed to **View»ActiveX Property Browser**.

NI Example Finder Enhancements

The **Most Recent** examples folder appears on the **Browse** tab of the NI Example Finder by default. You can set the maximum number of examples to display in the **Most Recent** examples folder by clicking the **Setup** button in the NI Example Finder and clicking the **General** tab.

Refer to the *NI Example Finder Help* for more information about enhancements to the NI Example Finder. Click the **Help** button in the NI Example Finder to display the *NI Example Finder Help*.

Source Control Enhancements

Refer to the **Fundamentals»Organizing and Managing a Project** book on the **Contents** tab in the *LabVIEW Help* for more information about source control in LabVIEW.

The following third-party source control providers have been tested for integration and basic functionality with the LabVIEW 8.2 Professional Development System:

- Seapine Surround SCM
- Borland StarTeam
- Telelogic Synergy
- PushOK (CVS and SVN plugins)
- ionForge Evolution

Note Currently, ionForge Evolution 2.8 and later works with LabVIEW.

Also, additional third-party source control providers are compatible with the procedure LabVIEW uses to complete a graphical differencing of VIs than were compatible with LabVIEW 8.0.

Source Control Operations on LabVIEW Project Folders

You can perform source control operations on a folder of items in a LabVIEW project. LabVIEW performs the operation on all appropriate items within the hierarchy. For example, if you add files to source control, LabVIEW adds only files within the folder that you have not yet added to source control. When you perform source control operations on a folder of items, all source control configuration options that apply to any of the folder items appear.

Note If you have a project library within a folder, the source control operations you perform do not extend to items within the project library. To perform source control operations on items within a project library, you must select the items manually. You can perform source control operations on a folder within a project library.

Unsaved Files in Source Control Operations

If you attempt check-in operations on files that contain unsaved changes, the **Unsaved Files** dialog box prompts you to save or ignore the changes. This dialog box appears if you click the **OK** button in the **Source Control Operations** dialog box and a LabVIEW file type included in the source control operation has unsaved changes. This dialog box also appears if you select **Tools»Source Control»Show Differences** in a library or project file that has unsaved changes.

Miscellaneous Source Control Enhancements

LabVIEW 8.2 includes the following miscellaneous source control enhancements:

- In LabVIEW 8.0, the **Add to Source Control** menu item is available for unsaved files in the **Tools»Source Control** menu. In LabVIEW 8.2, the **Add to Source Control** menu item is not available until you save the file.
- Select **Tools»Source Control»Configure Source Control** and use the **Exclude vi.lib** and **Exclude instr.lib** checkboxes to exclude files in the `vi.lib` and `instr.lib` directories when adding files to source control. If you configure LabVIEW to include dependencies when adding files to source control, these options exclude unnecessary files from the operation.
- The **Perforce Revision History** dialog box includes the options **Sync to this revision** and **Describe changelist**. To access these options, right-click a revision in the **History** list.
- In the **Perforce Revision History** dialog box, the **Actions in changelist** field displays actions taken in the current changelist. Possible operations include add, edit, delete, branch, and integrate. This field also displays the type of file on which the action occurs.
- **(Windows)** If you install the Perforce core installer, **Perforce SCM** appears in the **Source Control Provider Name** pull-down menu of the **Source Control** page of the **Options** dialog box.

TDM Enhancements

LabVIEW 8.2 includes the following TDM enhancements.

TDM Streaming File Format

LabVIEW 8.2 includes the TDM Streaming (`.tdms`) file format for storing binary data. The `.tdms` file format provides faster writing performance than the `.tdm` file format available in LabVIEW 8.0 and earlier. The `.tdms` file format also provides a simpler interface for defining properties.

Refer to the *TDM Streaming VIs and Functions* section of this document for information about TDMS VIs and functions.

User-Defined TDM and TDMS Properties

You can create and customize user-defined properties and specify DAQmx properties for binary measurement files. Use the Write To Measurement File, Write Data, Set Properties, and Get Properties Express VIs to configure user-defined properties for `.tdm` or `.tdms` files.

Click the **Advanced** button in the Write To Measurement File Express VI configuration page to configure properties. Display the configuration page of the Write Data, Set Properties, and the Get Properties Express VIs to configure properties.

Importing Web Services (Windows)

You can use Web services in LabVIEW 8.2 without managing all the complexities behind Web services. You can transform any Web service into a project library of VIs that you then can use to program the Web service as if it were independently available on the local computer. You must provide a valid URL to a Web Service Description Language (WSDL). WSDL is an XML-formatted language used to describe a Web service and its functionality. Select **Tools»Import»Web Service** to launch a wizard that guides you through the process of importing the methods in a Web service and creating a library of VIs.

Note You must have the .NET Framework, version 1.1 or later, installed to use the Import Web Service wizard.

Refer to the **Fundamentals»Windows Connectivity»Concepts»Web Services** book on the **Contents** tab in the *LabVIEW Help* for more information about importing Web services.

External Code Functions Changes

The data type `size_t` replaces type `int32` in certain places in the Memory Manager interface for use by external source code. This change is compatible with existing DLLs.

Refer to the *Memory Manager Functions* topic in the *LabVIEW Help* for more information about the Memory Manager functions.

371780B-01 Aug06