LabVIEW™ Upgrade Notes

These upgrade notes describe the process of upgrading LabVIEW for Windows, Mac OS, and Linux to LabVIEW 2009, issues you might encounter when you upgrade, and new features. To learn about any potential compatibility issues, read these upgrade notes prior to loading any VIs you saved in a previous version of LabVIEW in this new version of LabVIEW. Consider creating backup copies of all LabVIEW files you saved in a previous version of LabVIEW before you load the files in this new version of LabVIEW.

If you are upgrading from LabVIEW 7.1 or earlier to LabVIEW 2009, National Instruments recommends that you review the following documents in addition to these upgrade notes for more information about enhancements, changes, and features added to LabVIEW between LabVIEW 7.1 and LabVIEW 2009:

- **LabVIEW 8.0 Upgrade Notes**—The *Upgrade and Compatibility Issues* section and the *LabVIEW 8.0 Features and Changes* section provide important information for upgrade users. 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*.
- **LabVIEW 8.2 Upgrade Notes**—The *Upgrade and Compatibility Issues* section and the *LabVIEW 8.2 Features and Changes* section provide important information for upgrade users. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `upnote82` to access the *LabVIEW 8.2 Upgrade Notes*.
- **LabVIEW 8.5 Upgrade Notes**—The *Upgrade and Compatibility Issues* section and the *LabVIEW 8.5 Features and Changes* section provide important information for upgrade users. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `upnote85` to access the *LabVIEW 8.5 Upgrade Notes*.
- **LabVIEW 8.6 Upgrade Notes**—The *Upgrade and Compatibility Issues* section and the *LabVIEW 8.6 Features and Changes* section provide important information for upgrade users. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `upnote86` to access the *LabVIEW 8.6 Upgrade Notes*.

Refer to the *LabVIEW Help* for more information about LabVIEW 2009 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

Upgrading to LabVIEW 2009

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

1. To verify that you are aware of all compatibility issues before you install LabVIEW, refer to the following sections of this document prior to installing the new version of LabVIEW:
   - **Upgrading to LabVIEW 2009**—This section includes instructions for upgrading toolkits and modules, copying environment settings and `user.lib` files from a previous version of LabVIEW, and converting VIs to LabVIEW 2009.
   - **Upgrade and Compatibility Issues**—This section includes compatibility issues that might affect VIs you upgrade from a previous version of LabVIEW to the new version of LabVIEW. Specifically refer to the subsection that applies to the version of LabVIEW from which you are upgrading.

     Note You also can refer to the National Instruments Web site at `ni.com/info` and enter the info code `ex36rv` to download tests that can evaluate VIs for some compatibility issues.

   - **LabVIEW 2009 Features and Changes**—This section includes brief descriptions of the new features in this version of LabVIEW. Refer to the *LabVIEW Help* for more complete information about using these features. Access the *LabVIEW Help* by selecting **Help»Search the LabVIEW Help**.
2. (Optional) Uninstall any previous version(s) of LabVIEW.

3. Install and activate the upgrade version of LabVIEW. To verify that you complete all tasks associated with installing LabVIEW, refer to the following sections of the *LabVIEW Release Notes*:
   - *System Requirements*
   - *Installing LabVIEW 2009* and the appropriate subsection for the platform on which you are installing
   - *Installing LabVIEW Add-Ons* if you are installing LabVIEW toolkits or modules from media other than the LabVIEW Platform DVD
   - **(Windows)** *Activating the LabVIEW License* and all subsections
   - (Optional) *Installing and Configuring Hardware* and the appropriate subsection for the platform on which you are installing
   - *Where to Go from Here*
4. Refer to the *LabVIEW Readme* for issues fixed in the new version of LabVIEW, information about known issues in the new version of LabVIEW, and documentation additions that are not reflected in the *LabVIEW Help*. To access the *LabVIEW Readme*, navigate to the `labview\readme` directory and open the `readme.html` file.
5. Copy environment settings from a previous version of LabVIEW. Refer to the *Copying Environment Settings from a Previous Version of LabVIEW* section of this document for more information about copying environment settings.
6. Copy `user.lib` files from a previous version of LabVIEW. Refer to the *Copying user.lib Files from a Previous Version of LabVIEW* section of this document for more information about copying `user.lib` files.
7. Convert VIs to LabVIEW 2009. Refer to the *Converting VIs* section of this document for more information converting VIs saved in a previous version of LabVIEW.

Converting VIs

You cannot open a VI saved in LabVIEW 3.*x* or earlier without contacting an NI representative for information about upgrading your code to VI formats compatible with LabVIEW 2009. To open a VI saved in LabVIEW 4.0 to 5.*x*, you first must open and save the VI in LabVIEW 8.2 and then reopen the VI in LabVIEW 2009. When you open a VI last saved in LabVIEW 6.0 or later, LabVIEW 2009 automatically converts and compiles the VI. You must save the VI in LabVIEW 2009, 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.

Note VIs you save in LabVIEW 2009 do not load in earlier versions of LabVIEW. Before you save VIs in LabVIEW 2009 after you convert them, keep a backup copy of VIs you plan to use in LabVIEW 8.6, 8.5, 8.2, or 8.0. Select **File»Save for Previous Version** to save VIs so they can run in LabVIEW 8.6, 8.5, 8.2, or 8.0.

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

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

Upgrading Modules, Toolkits, and Instrument Drivers

If you are upgrading from a previous version of LabVIEW, you must install current, compatible versions of any modules, toolkits, or instrument drivers that you installed for the previous version of LabVIEW. The LabVIEW Platform DVDs include most modules and toolkits that are compatible with LabVIEW 2009. For those modules and toolkits that are not on the LabVIEW Platform DVDs, refer to the National Instruments Web site at `ni.com/info` and enter the info code `compat` for more information about which LabVIEW modules and toolkits are compatible with the current version of LabVIEW.

NI Modules and Toolkits

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

Operating System	Media to Use	Important Notes
Windows	DVD	Use the LabVIEW Platform DVDs to install LabVIEW 2009 and versions of modules and toolkits that are compatible with LabVIEW 2009. Additionally, you can choose to evaluate modules or toolkits you have not purchased. The LabVIEW Platform DVDs allow you to install new versions of a toolkit with LabVIEW 2009 without uninstalling or modifying previous versions. Refer to the *LabVIEW Release Notes* for information about installing LabVIEW, modules, and toolkits.
Mac/Linux; Windows, if the LabVIEW Platform DVDs do not include the module or toolkit	CD	Use the installation CD you received when you purchased the module or toolkit. Before using the installation CD, make sure you have a compatible version of the module or toolkit you want to install. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `compat` for more information about which LabVIEW modules and toolkits are compatible with the current version of LabVIEW. Then install the compatible modules and toolkits in the LabVIEW 2009 directory. Mass compile any VIs that you saved in previous versions of LabVIEW. Refer to the *Mass Compiling LabVIEW* section of this document for more information.

Note Some versions of toolkits do not work with LabVIEW 2009. Installing an incompatible toolkit might cause some features in the toolkit or LabVIEW to behave incorrectly. National Instruments recommends that you verify compatibility before attempting to install toolkits. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `compat` for more information about which LabVIEW modules and toolkits are compatible with the current version of LabVIEW. If you install an incompatible version and corrupt your LabVIEW 2009 installation, first uninstall the toolkit and then repair the LabVIEW installation using the Add/Remove Programs applet in the Control Panel.

Instrument Drivers

You must install current instrument drivers to control and communicate with any instrument hardware you plan to use. If you installed an instrument driver with a previous version of LabVIEW, you must reinstall that instrument driver with LabVIEW 2009 support using one of the following methods:

- **NI Modular Instrument drivers**—Use the NI Device Drivers DVD or CD to install NI Modular Instrument drivers.
- **Plug and Play instrument drivers**—Use the NI Instrument Driver Finder to search for and install LabVIEW Plug and Play instrument drivers without leaving the LabVIEW development environment.
- **IVI driver or non-certified instrument drivers**—Use the Instrument Driver Network on the National Instruments Web site to search for and install an IVI driver or non-certified drivers.

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

Third Party Add-Ons

Contact the vendor of third-party LabVIEW add-ons to determine whether the add-on is compatible with LabVIEW 2009 for your operating system. Make sure you mass compile any VIs that are related to the add-on.

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

Mass Compiling LabVIEW

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

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

Upgrading Additional National Instruments Software

You must use NI TestStand 3.5 or later with LabVIEW 2009. 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.

NI TestStand 3.5 and later returns an error when you attempt to configure the following LabVIEW 2009 Express VIs:

- Group Digital Signals
- Get Dynamic Data Attributes
- Set Dynamic Data Attributes

Refer to the National Instruments Web site at `ni.com/info` and enter the info code `rdtf10` for more information about the error.

Note NI TestStand 4.1 or later resolves this issue.

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

You must use NI Spy 2.3 or later in LabVIEW 2009. NI Spy 2.5 is available on the National Instruments Device Drivers CD.

LabVIEW 2009 supports Measurement Studio 8.0 or later. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `exd8yy` to access the Upgrade Advisor and purchase Measurement Studio 8.0 or later.

Upgrading from Previous Versions of LabVIEW

You can install LabVIEW 2009 without uninstalling previous versions of LabVIEW. While versions of LabVIEW might share components, upgrading to new versions of LabVIEW does not affect the performance of 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` directory, where *x* is the version number.

Replacing an Existing Version of LabVIEW

To replace your existing version of LabVIEW, uninstall the existing version of LabVIEW, run the LabVIEW 2009 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 2009 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 If you replace the LabVIEW 2009 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 2009, copy the LabVIEW preferences file to the LabVIEW 2009 directory.

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

(Mac OS) LabVIEW stores preferences in the LabVIEW preferences text file at `~/Library/Preferences/LabVIEW 9.0 Preferences`.

(Linux) LabVIEW stores preferences at `/home/<username>/natinst/.config/LabVIEW-2009/labview.conf`, where <username> is the username of the user running the current instance of LabVIEW.

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

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 2009, copy the files to the `user.lib` directory in the LabVIEW 2009 directory.

Upgrade and Compatibility Issues

Refer to the following sections for upgrade and compatibility issues specific to different versions of LabVIEW. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `lvupgrade` for general information about upgrading to the latest version of LabVIEW.

Refer to the `readme.html` file in the `labview` directory for information about known issues in the new version of LabVIEW, additional compatibility issues, and information about late-addition features in LabVIEW 2009. You also can refer to the Developer Zone on `ni.com` for general information about upgrading to the latest version of LabVIEW.

Upgrading from LabVIEW 8.6

You might encounter the following compatibility issues when you upgrade to LabVIEW 2009 from LabVIEW 8.6.

Note You also can refer to the National Instruments Web site at `ni.com/info` and enter the info code `upnote86` for more information about additional issues you might encounter upgrading from LabVIEW 8.6.*x*.

System Requirements

(Windows) LabVIEW 2009 requires at least 1.6 GB of disk space for the LabVIEW installation.

(Mac OS) LabVIEW 2009 requires at least 563 MB of disk space for the minimum LabVIEW installation or 1.2 GB of disk space for the complete LabVIEW installation.

(Linux) LabVIEW 2009 requires at least 630 MB of disk space for the minimum LabVIEW installation or 835 MB of disk space for the complete LabVIEW installation.

VI and Function Behavior Changes

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

Bluetooth VIs and Functions

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

Signal Generation VIs

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

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

Miscellaneous VI and Function Behavior Changes

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

- If you wire a value that has a unit with an odd exponent to the square root function, the wire breaks because LabVIEW does not support units with fractional exponents.
- The Bessel Coefficients VI is rewritten to implement cutoff frequencies more correctly. As a result, the Bessel Coefficients VI and any calling VIs might run more slowly than in previous versions of LabVIEW.
- LabVIEW deploys Web services to version-specific directories For example, a typical root location for deployed Web services in LabVIEW 2009 is `C:\Documents and Settings\All Users\Application Data\National Instruments\Web Services 2009 32-bit`. You must redeploy any Web services created in a previous version of LabVIEW to use the Web services in LabVIEW 2009. To delete a Web service deployed by a previous version of LabVIEW, you must manually remove it from the deployed location.

- The Integral x(t) VI is rewritten in LabVIEW 2009. To use the new functionality, replace this VI with the Integral x(t) VI from the **Functions** palette.

Deprecated VIs and Functions

LabVIEW 2009 does not support the following VIs and functions:

- **LToCStr**—Use the LToCStrN function instead. The LToCStrN function differs from the LToCStr function because it takes a parameter specifying the size of the C string buffer to which LabVIEW copies the string. These function are Code Interface Node (CIN) functions.
- **Sound VIs (Mac OS)**—Use the **Sound** VIs instead. LabVIEW 2009 supports the same API for Windows, Mac OS, and Linux.

Deprecated Properties, Methods, and Events

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

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

Renamed Properties, Methods, and Events

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

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

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

Application Builder Changes

In LabVIEW 8.6, the Application Builder saves VIs and library files in a flat list within the application and saves VIs with conflicting filenames outside the application in separate folders. In LabVIEW 2009, the Application Builder stores source files within the application using a layout similar to the directory structure of the source files on disk. This internal file layout preserves source file hierarchy inside the application.

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

Case Structure Output Tunnel Changes

LabVIEW 2009 determines the data type from a Case structure output tunnel by using a data type that can handle all cases in the structure, including cases that never execute. For example, consider a Case structure with two cases, TRUE and FALSE. In the TRUE case, a U8 data type is wired to an output tunnel. In the FALSE case, a U32 data type is wired to the output tunnel. In LabVIEW 8.6.*x*, if you wire a constant to select the TRUE case, the data type from the output tunnel is U8 because the constant prevents the FALSE case from executing. In LabVIEW 2009, if you wire a constant to select the TRUE case, the data type from the output tunnel is U32.

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

Custom Icon Editor VI Changes

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

Custom Probes Changes (Linux)

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

LabVIEW MathScript Changes

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

.NET Changes

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

Upgrading from LabVIEW 8.5

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

Note You also can refer to the National Instruments Web site at `ni.com/info` and enter the info code `upnote85` for more information about additional issues you might encounter upgrading from LabVIEW 8.5.*x*.

Platforms Supported

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

System Requirements

(Windows) LabVIEW 8.6 and LabVIEW 2009 require at least 1.6 GB of disk space for the LabVIEW installation.

(Mac OS) LabVIEW 8.6 requires at least 262 MB of disk space. LabVIEW 2009 requires at least 563 MB of disk space for the minimum LabVIEW installation or 1.2 GB of disk space for the complete LabVIEW installation.

(Linux) LabVIEW 8.6 requires at least 365 MB of disk space for the minimum LabVIEW installation or 651 MB of disk space for the complete LabVIEW installation. LabVIEW 2009 requires at least 630 MB of disk space for the minimum LabVIEW installation or 835 MB of disk space for the complete LabVIEW installation.

VI and Function Behavior Changes

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

Report Generation VIs

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

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

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

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

External Code (DLLs and CINS)

The memory manager functions include only one zone of memory, DS (data space). If you work with C or C++ CINs or DLLs that manage LabVIEW memory, replace all references to AZ (application zone) memory functions with the DS equivalent function.

Miscellaneous VI and Function Behavior Changes

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

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

- If you wire an empty path to the **path in** input of the Call Library Function Node, LabVIEW no longer returns an error.
- The **output element** output of the Get Report Type VI changed to **report type**. LabVIEW automatically renames and reconnects code you take from previous LabVIEW versions and insert into LabVIEW 8.6 and later. However, the VI breaks if you use the Call By Reference Node function to call the Get Report Type VI.
- The **report type** input of the New Report VI is a required input. You must wire data to this input. You can right-click the VI and create a constant or a control for the **report type** input.

Deprecated VIs and Functions

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

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

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

- **Xmath script node**—Use the MathScript Node instead. Because the MathScript syntax is different from the Xmath syntax, you might need to modify existing scripts to work in the MathScript Node.

Property, Method, and Event Behavior Changes

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

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

Deprecated Properties, Methods, and Events

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

- Control Value:Set [Flattened] method of the VI class. Use the Control Value:Set method instead.

- Control Value:Get All [Flattened] method of the VI class. Use the Control Value:Get All method instead.
- Control Value:Get [Flattened] method of the VI class. Use the Control Value:Get method instead.
- VIModificationBitSet property of the VI Properties (ActiveX) class. Use the VIModificationBitSet2 property instead.
- Modifications:VI Modifications Bitset property of the VI class. Use the new Modifications:VI Modifications Bitset property instead. In LabVIEW 8.5 and earlier, the Modifications:VI Modifications Bitset property returns a 32-bit value. In LabVIEW 8.6 and later, the new Modifications:VI Modifications Bitset property returns a 64-bit value.

Renamed Properties, Methods, and Events

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

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

Shared Variable Changes

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

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

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

Upgrading Remote Front Panel Licenses (Windows)

You can allow clients to view an application or front panel remotely using LabVIEW. LabVIEW supports licenses that allow 5, 20, 50, or unlimited clients to connect to a remote front panel at one time. You can have only one license on the server computer. Purchase a license that allows enough connections to accommodate the number of clients you want to allow. If you upgraded the remote front panel license for LabVIEW 8.5.1 or earlier, you must use your existing serial number to activate a new license of equal value in NI License Manager.

Shared Components of the Application Builder

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

Saving Password-Protected VIs for Previous Versions

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

Upgrading from LabVIEW 8.2

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

Note You also can refer to the National Instruments Web site at `ni.com/info` and enter the info code `upnote82` for more information about additional issues you might encounter upgrading from LabVIEW 8.2.*x*.

Platforms Supported

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

- LabVIEW 8.5 and later supports Windows Vista and Windows Vista 64-bit.
- LabVIEW 8.5 and 8.5.1 support Macintosh computers with both Intel and PowerPC processors. LabVIEW 8.6 and later does not support Macintosh computers with PowerPC processors.

System Requirements

(Windows) LabVIEW 8.5 requires at least 1.2 GB of disk space for the LabVIEW installation. LabVIEW 2009 requires at least 1.6 GB of disk space for the LabVIEW installation.

(Mac OS) LabVIEW 8.5 requires at least 502 MB of disk space for the minimum LabVIEW installation or 734 MB of disk space for the complete LabVIEW installation. LabVIEW 2009 requires at least 563 MB of disk space for the minimum LabVIEW installation or 1.2 GB of disk space for the complete LabVIEW installation.

(Linux) LabVIEW 8.5 requires at least 450 MB of disk space for the minimum LabVIEW installation or 640 MB of disk space for the complete LabVIEW installation. LabVIEW 2009 requires at least 630 MB of disk space for the minimum LabVIEW installation or 835 MB of disk space for the complete LabVIEW installation.

Windows Vista Compatibility Issues

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

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

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

VI and Function Behavior Changes

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

Enhancements to Analysis VIs and Functions

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

Mathematics VIs

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

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

Numeric Functions

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

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

Signal Processing VIs

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

Hyperbolic Functions

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

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

Libraries & Executables VIs and Functions

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

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

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

Miscellaneous VI and Function Behavior Changes

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

- The Instr Get Attribute VI and Instr Set Attribute VI no longer ship with LabVIEW. If you use either of these VIs in an application, replace them with the Property Node on the **VISA Advanced** palette for equivalent functionality.

- The **All Folders** parameter of the Recursive File List VI can contain folder shortcuts, but the VI does not recurse them.

Case Structure Output Tunnel Changes

The behavior of the Case structure output tunnel changed between LabVIEW 8.2 and LabVIEW 8.5. LabVIEW 8.2 determines the data type from a Case structure output tunnel by using a data type that can handle all cases in the structure, including cases that never execute. However, LabVIEW 8.5 to 8.6.*x* determine the data type from a Case structure output tunnel by using the data type of the case that has a constant wired to it.

In LabVIEW 2009, the Case structure output tunnel behaves the same as in LabVIEW 8.2. That is, LabVIEW 2009 determines the data type from a Case structure output tunnel by using a data type that can handle all cases in the structure, including cases that never execute. Refer to the *LabVIEW Help* for more information about the behavior of the Case structure output tunnel in LabVIEW 2009.

Property, Method, and Event Behavior Changes

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

- The Data Binding:Path property of the Control class is read/write and settable when the VI is running. To write this property, you must bind the control to an NI Publish-Subscribe-Protocol URL before you begin writing.
- The Target:CPU property of the Application class includes the value `AMD/Intel x64`.
- The Target:Operation System property of the Application class includes the values `Windows x64` and `Linux x64`.
- The Point to Row Column method of the TreeControl class returns the tag `TREE_COLUMN_HEADERS` when you wire a point within the column headers of the tree.
- The LabVIEW Class:Create method includes a name input. If you do not wire the **name** input, LabVIEW prompts the user to name the class at run time.
- The Control Value:Get [Variant], Control Value:Get [Flattened], Control Value:Set [Variant], and Control Value:Set [Flattened] methods no longer trim leading and trailing whitespace when searching for controls.

Deprecated Properties, Methods, and Events

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

- Default Instance property of the LVClassLibrary class. Use the Get LV Class Default Value VI instead.
- Geometry property of the SceneObject class. Use the Drawable property instead.
- Grid Colors property of the GraphChart class. Use the Grid Colors property of the GraphScale class instead.
- Grid Colors:X Color property of the GraphChart class. Use the Grid Colors:Major Color and Grid Colors:Minor Color properties instead.
- Grid Colors:X Color property of the GraphChart class. Use the Grid Colors:Major Color and Grid Colors:Minor Color properties instead.
- Grid Colors:Y Color property of the GraphChart class. Use the Grid Colors:Major Color and Grid Colors:Minor Color properties instead.
- Legend:Plots Shown property of the WaveformChart class. Use the Legend:Number of Rows property instead.
- Legend:Plots Shown property of the WaveformGraph class. Use the Legend:Number of Rows property instead.
- Pixel Width property of the ListBox class. Use the Bounds:Area Width property instead.

- Scrollbars Visible property of the Picture class. Use the Horizontal Scrollbar Visible and Vertical Scrollbar Visible properties instead.
- Set Geometry method of the SceneObject class. Use the Set Drawable method instead.
- Scene:Geometry:New Mesh method of the Application class. Use the Scene:Drawable:Geometry:New Mesh method instead.
- Drag Starting event of the Control class. Use the Drag Starting event of the appropriate control class instead.
- Drag Starting? event of the Control class. Use the Drag Starting? event of the appropriate control class instead.

Renamed Properties, Methods, and Events

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

Class	LabVIEW 8.2 Name	LabVIEW 8.5 Name	Type
AbsTime, Numeric	Data Range	Data Entry Limits	Property
AbsTime, Numeric	Data Range:Increment	Data Entry Limits:Increment	Property
AbsTime, Numeric	Data Range:Maximum	Data Entry Limits:Maximum	Property
AbsTime, Numeric	Data Range:Minimum	Data Entry Limits:Minimum	Property
AbsTime, Numeric	Out of Range Action	Response to Value Outside Limits	Property
AbsTime, Numeric	Out of Range Action:Increment	Response to Value Outside Limits:Increment	Property
AbsTime, Numeric	Out of Range Action:Maximum	Response to Value Outside Limits:Maximum	Property
AbsTime, Numeric	Out of Range Action:Minimum	Response to Value Outside Limits:Minimum	Property
Application	Library:Get Project Library File Version	Library:Get File LabVIEW Version	Method
Application	Scene:Geometry:New Box	Scene:Drawable:Geometry:New Box	Method
Application	Scene:Geometry:New Cone	Scene:Drawable:Geometry:New Cone	Method
Application	Scene:Geometry:New Cylinder	Scene:Drawable:Geometry:New Cylinder	Method
Application	Scene:Geometry:New Height Field	Scene:Drawable:Geometry:New Height Field	Method
Application	Scene:Geometry:New Mesh	Scene:Drawable:Geometry:New Mesh	Method
Application	Scene:Geometry:New Sphere	Scene:Drawable:Geometry:New Sphere	Method
Application (ActiveX)	LibraryGetProjectLibFileVersion	LibraryGetFileLVVersion	Method
Digital, NumericText, and Scale	Format & Precision	Display Format	Property
Digital, NumericText, and Scale	Format & Precision:Format	Display Format:Format	Property
Digital, NumericText, and Scale	Format & Precision:Precision	Display Format:Precision	Property
DigitalTable	Column Headers Visible	Signal Number Visible	Property
DigitalTable	Row Headers Visible	Transitions Visible	Property

Class	LabVIEW 8.2 Name	LabVIEW 8.5 Name	Type
SceneGraphDisplay and SceneWindow	Clear Color	Background Color	Property
SceneObject	Set Geometry	Set Drawable	Method
VI	Connector Pane	Connector Pane:Set	Property

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

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

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

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

LabVIEW Class Icons

If you created a LabVIEW class icon in LabVIEW 8.2 and you want the icon displayed when you place a class control or indicator on the block diagram, you must update the class icon to occupy a smaller space so that the class mask does not obscure any part of the class icon. Use an image no larger than 32 pixels wide by 19 pixels high.

Opening LLBs in LabVIEW

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

Timed Loop Priority Level Restriction

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

Waveform Data Type

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

Enum Coercion

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

Upgrading from LabVIEW 8.0

You might encounter the following compatibility issues when you upgrade to LabVIEW 2009 from LabVIEW 8.0. Refer to the *Upgrading from LabVIEW 8.2*, *Upgrading from LabVIEW 8.5*, and *Upgrading* *from LabVIEW 8.6* sections of this document for information about other upgrade issues you might encounter.

Note You also can refer to the National Instruments Web site at `ni.com/info` and enter the info code `upnote8` for more information about additional issues you might encounter upgrading from LabVIEW 8.0.

Platforms Supported

LabVIEW 8.2 and later includes the following changes in platforms supported:

- LabVIEW 8.2 and later 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. LabVIEW 8.6 and later does not support Macintosh computers with PowerPC processors.

System Requirements

(Windows) LabVIEW 8.2 requires at least 1.2 GB of disk space for the LabVIEW installation. LabVIEW 2009 requires at least 1.6 GB of disk space for the LabVIEW installation.

(Mac OS) LabVIEW 8.2 requires at least 500 MB of disk space for the minimum LabVIEW installation or 700 MB of disk space for the complete LabVIEW installation. LabVIEW 2009 requires at least 563 MB of disk space for the minimum LabVIEW installation or 1.2 GB of 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 of disk space for the complete LabVIEW installation. LabVIEW 2009 requires at least 630 MB of disk space for the minimum LabVIEW installation or 835 MB of disk space for the complete LabVIEW installation.

VI and Function Behavior Changes

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

Communicating between Application Instances

In LabVIEW 8.2 and later, 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 **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 or later 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 or later.

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 or later, 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 display a loading dialog box when searching for missing subVIs of the referenced VI. The user is not prompted to find VIs that LabVIEW cannot find automatically. A new value, `0x20`, specifies not to display the loading dialog box to find missing subVIs of the referenced VI. To prompt the user to find missing subVIs, use option the `0x10` value.

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 and later, 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 and later 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 and later 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 and later:

- 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. In LabVIEW 8.2 and later, .NET methods that pass array data types by reference pass 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 and later, 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 and later does not support the following properties, methods, and events:

- LabVIEW 8.2 and later 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 and later.

Class	LabVIEW 8.0 Name	LabVIEW 8.2 and Later 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
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

Class	LabVIEW 8.0 Name	LabVIEW 8.2 and Later Name	Type
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

Application Builder Changes

In LabVIEW 8.2 and later, you cannot view the contents of a stand-alone application (EXE) or shared library (DLL) by renaming the application or shared library to have a `.llb` file extension. You also cannot access a VI in a stand-alone application or shared library by specifying the path to the VI from outside of the application or shared library. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `exjk3b` for more information about viewing and accessing applications and shared libraries.

Upgrading from LabVIEW 7.x

You might encounter the following compatibility issues when you upgrade to LabVIEW 2009 from LabVIEW 7.*x*. Refer to the *Upgrading from LabVIEW 8.0*, *Upgrading from LabVIEW 8.2*, *Upgrading from LabVIEW 8.5*, and *Upgrading from LabVIEW 8.6* sections of this document for information about other upgrade issues you might encounter.

Note You also can refer to the National Instruments Web site at `ni.com/info` and enter the info code `rd70un` for more information about additional issues you might encounter upgrading from LabVIEW 7.0. Enter the info code `exucme` for more information about additional issues you might encounter upgrading from LabVIEW 7.1.

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 screen resolution of 800 × 600 pixels, but National Instruments recommends a screen resolution of 1,024 × 768 pixels. LabVIEW 2009 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 2009 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 2009 requires at least 1.6 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 2009 requires at least 563 MB of disk space for the minimum LabVIEW installation or 1.2 GB 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 2009 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 2009 requires at least 630 MB of disk space for the minimum LabVIEW installation or 835 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 2009 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 2009 runs on Red Hat Enterprise Linux WS 4 or later and SuSE Linux 10.3 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:

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

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`, `%b`, 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 and is instead using a 2D array, a red 7.*x* glyph appears on the function. The red glyph indicates that LabVIEW replaced the 2D array with the matrix data type.

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 VI name 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. In LabVIEW 8.5 and later, an error occurs if LabVIEW cannot find the file or if the VI name of the file is the same as the qualified VI name as another VI in memory on that target.

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 string instance of 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 string instance of 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 updates the VI in order to maintain 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.

Note In LabVIEW 8.6 and later, **Magnitude** is renamed **Magnitude, Real, Imaginary**.

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** (the Transition Measurements VI equivalent is the **post-transition** output), or **preshoot** (the Transition Measurements VI equivalent is the **pre-transition** output) 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*.

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. If the Run VI Method generates an error, the reference is not automatically closed. In LabVIEW 8.0, LabVIEW automatically 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. In LabVIEW 8.2 and later, the behavior has been changed back to the 7.1 behavior.

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.

Modifications:VI Modifications Bitset Property

In LabVIEW 7.*x* and earlier, the Modifications:VI Modifications Bitset property changes the value to nonzero when you must relink a VI. In LabVIEW 8.*x* and later, the value remains zero if no changes were made even though you might need to relink a VI.

TreeControl Properties

In LabVIEW 7.*x* and earlier, the Active Cell Properties:Cell Size:Height and Active Cell Properties:Cell Size:Width properties return one less pixel for each line in the tree control than in LabVIEW 8.*x*. For example, if you load a LabVIEW 7.*x* VI in LabVIEW 8.*x* that contains a property node which returns the height and width of a tree control as 70 pixels and 16 pixels, any new property nodes you place to determine height and width return 69 pixels and 15 pixels.

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, MulitcolumnListbox, 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.

In LabVIEW 8.*x* you cannot select the Mouse Down, Mouse Down?, Mouse Move, or Mouse Up events of the VI class in the **Edit Events** dialog box. Use the Mouse Down, Mouse Down?, Mouse Move, and Mouse Up events in the Pane class, respectively, instead.

Changes from Previous Versions of the Application Builder

The Application Builder is integrated into the **Project Explorer** window. If you use the LabVIEW Base Package or Full Development System, you can purchase the Application Builder separately by visiting the National Instruments Web site at `ni.com/info` and enter the info code `rdlv21`.

Use **Build Specifications** in the **Project Explorer** window to create build specifications for and build stand-alone applications (EXEs), shared libraries (DLLs), and zip files. **(Windows)** You also can use **Build Specifications** to create build specifications for and build installers. Build specifications are equivalent to `.bld` files in previous versions of the Application Builder, but they now are part of a LabVIEW project instead of separate files.

Note You must use the Application Builder tools within a project.

You can convert a `.bld` file into a build specification in a new project. Select **Tools»Convert Build Script** to navigate to and select the `.bld` file to convert. LabVIEW uses the file to create a project that contains the source files and build specifications.

Application Item Tags

The following list includes application item tags that have been removed from LabVIEW either because the feature is no longer available or because it has been combined with another feature:

- `APP_SAVE_WITH_OPTIONS`
- `APP_UPDATE_VXI`
- `APP_DSC_TOOLBAR`
- `APP_DSC_TAGEDITOR`
- `APP_DSC_TAGMONITOR`
- `APP_DSC_HTV`
- `APP_DSC_ENGINE`
- `APP_DSC_SECURITY`
- `APP_DSC_LOGOUT`
- `APP_DSC_CPWD`

- `APP_DSC_USERINFO`
- `APP_DSC_USEREDITOR`
- `APP_DSC_ADVANCED`
- `APP_DSC_STARTUP`
- `APP_DSC_SRVBRW`
- `APP_DSC_IST`
- `APP_DSC_IMAGENAV`
- `APP_DSC_OPTIONS`
- `APP_SRC_CODE_CTRL`
- `APP_BUILD_STANDALONE_APP`
- `APP_EDIT_VI_LIBRARY`
- `APP_DN_ASSEMBLY_REFS`
- `APP_SHOW_CLIPBOARD`
- `APP_VIEW_PRINTED_MANUALS`
- `APP_RT_ENGINE_INFO`
- `APP_SWITCH_EXEC_TARGET`
- `APP_REALTIME`

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. Delaying the operating system messages, such as keyboard messages from the users, is useful in order to avoid calling the same shared library file while a shared library function runs. For example, if the shared library function is called in response to the user pressing a button, the user should not be able to press the button again until the shared library function has completed.

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 2009 from LabVIEW 6.*x*. Refer to the *Upgrading from LabVIEW 7.x*, *Upgrading from LabVIEW 8.0*, *Upgrading from LabVIEW 8.2*, *Upgrading from LabVIEW 8.5*, and *Upgrading from LabVIEW 8.6* sections of this document for information about other upgrade issues you might encounter.

Note Refer to the *LabVIEW Upgrade Notes* for each version of LabVIEW between LabVIEW 6.*x* and LabVIEW 2009 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 serial 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.

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 converting VIs saved in previous versions of LabVIEW so they work with later versions and vice versa.

LabVIEW 2009 Features and Changes

Refer to the *LabVIEW Help* for more information about LabVIEW 2009 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, a partial list of bugs fixed, additional compatibility issues, and information about late-addition features in LabVIEW 2009.

Installing LabVIEW

(Windows) With LabVIEW 2009 you can install LabVIEW and most modules and toolkits from the LabVIEW Platform DVDs. Refer to the *Upgrading Modules, Toolkits, and Instrument Drivers* section of this document or the *Installing LabVIEW 2009* section of the *LabVIEW Release Notes* for more information.

New Example VIs

Refer to the **New Examples for LabVIEW 2009** folder on the **Browse** tab of the NI Example Finder to view descriptions for and launch example VIs added to LabVIEW 2009.

Block Diagram Enhancements

LabVIEW 2009 includes the following enhancements to the block diagram and related functionality.

Cleaning Up Sections of the Block Diagram Automatically

You can select wires and objects on the block diagram and then select **Edit»Clean Up Selection** to reroute the wires and reorder the objects you selected automatically. To prevent LabVIEW from rerouting wires and reordering objects within a structure, right-click the structure and select **Exclude from Diagram Cleanup** from the shortcut menu.

Enabling Parallel For Loop Iterations

When you use a For Loop, LabVIEW executes the loop iterations sequentially. If a For Loop is computationally intensive, consider running the loop iterations parallel to each other to improve performance. Select **Tools»Profile»Find Parallelizable Loops** to analyze a VI for For Loop iterations you can parallelize. LabVIEW displays results in the **Find Parallelizable Loops Results** window. If you launch this window from the **Project Explorer** window, LabVIEW analyzes all VIs in the project.

Right-click a For Loop border and select **Configure Iteration Parallelism** from the shortcut menu to display the **For Loop Iteration Parallelism** dialog box. Use this dialog box to enable and configure parallelism on the iterations of a For Loop.

Shared Variable Enhancements

LabVIEW 2009 includes the following shared variable enhancements.

Fixed-Point Data Type Support

Shared variables, including I/O variables, support the fixed-point data type.

Programmatically Finding, Reading, and Writing Variables

LabVIEW 2009 includes new functions, classes, properties, and methods you can use to find, read and write network-published shared variables and I/O variables programmatically. Although you still can use the **Datasocket** VI and functions to read and write shared variables programmatically, the new shared variable API offers enhanced flexibility. Unlike the Datasocket API, the new shared variable API supports I/O variables and includes classes you can use to work with variable engines and variable containers.

Scanned I/O Support

LabVIEW 2009 includes support for scanned I/O with the NI Scan Engine and I/O variables, which previously required the LabVIEW Real-Time Module. The NI Scan Engine enables efficient single-point access to sets of data channels, such as I/O channels, using a scan that stores data in a global memory map and updates all values at a single rate, known as the scan period. An I/O variable is a type of shared variable that uses the NI Scan Engine for single-point access to I/O data. At the time of the LabVIEW 2009 release, CompactRIO, DeviceNet , EtherCAT, and Wireless Sensor Network devices offer drivers with NI Scan Engine support. However, additional I/O drivers with NI Scan Engine support might become available before the next release of LabVIEW. Refer to the specific I/O driver documentation for more information about NI Scan Engine support.

Timed Structure Timing Sources

LabVIEW 2009 includes the following built-in timing sources you can select as the timing source for a Timed Loop, Timed Loop with Frames, or Timed Sequence structure:

- **1 kHz <absolute time>**
- **1 MHz <absolute time>**

You can use the new **1 kHz <absolute time>** and **1 MHz <absolute time>** timing sources to synchronize the start of timed structures running on multiple computing devices.

Miscellaneous Block Diagram Enhancements

LabVIEW 2009 includes the following miscellaneous block diagram enhancements:

- You can right-click a Shared Variable node and select **Show Variable In Project** from the shortcut menu to highlight the corresponding shared variable in the **Project Explorer** window.
- Custom probes are available in the LabVIEW Base Package.

Front Panel Enhancements

LabVIEW 2009 includes the following enhancements to the front panel and related functionality.

2D Graph Enhancements

The **Graph** palette includes the following new 2D graphs:

- 2D Compass Plot
- 2D Error Plot
- 2D Feather Plot
- XY Plot Matrix

3D Graph Enhancements

The **3D Graph** palette includes the following new 3D graphs:

- 3D Bar Plot
- 3D Comet Plot
- 3D Contour Plot
- 3D Mesh Plot
- 3D Pie Plot
- 3D Quiver Plot
- 3D Ribbon Plot
- 3D Scatter Plot
- 3D Stem Plot
- 3D Surface Plot
- 3D Waterfall Plot

Miscellaneous Front Panel Enhancements

LabVIEW 2009 includes the following miscellaneous front panel enhancements:

- Digital waveform graphs automatically display the line and bus names you specify for the digital signals.
- The name of the 3D Picture Control changed to 3D Picture.
- The name of the Picture indicator changed to 2D Picture. In addition, 2D Picture indicators are available in the LabVIEW Base Package.

- Subpanel controls and tree controls are available in the LabVIEW Base Package.
- You can right-click a front panel control bound to a shared variable and select **Show Variable In Project** from the shortcut menu to highlight the corresponding shared variable in the **Project Explorer** window.
- In LabVIEW 8.6 and earlier, the default range of the front panel scroll bar allows you to scroll past all visible objects by one-fourth of the window dimensions during both run time and edit time. In LabVIEW 2009, the default range of the scroll bar behaves differently during run time than during edit time. During run time the default range of the scroll bar extends 10 pixels past the bounding rectangle of all visible front panel objects.

Environment Enhancements

LabVIEW 2009 includes the following enhancements to the LabVIEW environment.

Access Scope Enhancements

Libraries, LabVIEW classes, and XControls include a community access scope. Use community access scope to allow only members of a library and any VI or library designated as a friend to access the library. Use the **Item Settings** page of the **Project Library Properties** dialog box to specify the access scope for members of a library. Use the **Friends** page of the **Project Library Properties** dialog box to specify the friends of a project library.

If you specify a VI as a friend of a library, the VI can access any member of the library that is in community scope. You also can specify a library as a friend of another library.

Use the **Class Properties** and **XControl Properties** dialog boxes to set classes and XControls to community access scope and specify friends.

Create New Instrument Driver Project Wizard Enhancements

The **Create New Instrument Driver Project** wizard includes the following new templates:

- Counter
- Power Meter

Domain Account Manager Enhancements (Windows)

LabVIEW 2009 includes the following enhancements to the **Domain Account Manager**:

- Use the **Domain Account Manager** to export a local domain. Then import the domain to a computer you use to deploy LabVIEW applications. When you import a domain, the **Domain Account Manager** checks whether a local domain already exists on the computer. If a local domain exists, the **Domain Account Manager** prompts you to save a copy of the existing local domain before you import the new domain. The **Domain Account Manager** does not allow you to import a domain if it already exists on the network.
- The **Import Lookout 4.x Security File** menu option is renamed to **Import Lookout Security File** and is no longer available in the **File»New** menu of the **Domain Account Manager**. Instead, select **File»Import and Export»Import Lookout Security File** or right-click the domain list and select **Import and Export»Import Lookout Security File** from the shortcut menu to import a Lookout account.

Getting Started Window Enhancements

LabVIEW 2009 updates the **Latest from ni.com** section of the **Getting Started** window with news, technical content, example programs, and training information from the National Instruments Web Site at `ni.com`. LabVIEW dynamically populates these categories with a list of articles that relate to LabVIEW and any modules and toolkits you have installed. LabVIEW notifies you of new content by displaying the number of unread articles next to the category name.

Note LabVIEW disables updates from `ni.com` by default in localized versions of LabVIEW. Localized versions of LabVIEW do not support automatic updates from `ni.com`. In localized versions of LabVIEW, the links in this category display static content from `ni.com` in a Web browser.

LabVIEW Class Hierarchy Window Enhancements

LabVIEW 2009 includes the following enhancements to the **LabVIEW Class Hierarchy** window:

- The color of the background, the lines that connect items, and the VI borders in the **LabVIEW Class Hierarchy** window changed to improve visibility of items in the hierarchy.
- The **LabVIEW Class Hierarchy** window arranges nodes in the hierarchy into groups according to the libraries to which the nodes belong. Nodes in the same library share the same border color.
- The **View** menu in the **LabVIEW Class Hierarchy** window includes the following new menu items: **Always Show Labels**, **Zoom In**, **Zoom Out**, **Actual Size**, and**Fit to Window**.

Performance Enhancements

LabVIEW 2009 includes an upgraded version of Math Kernel Library (MKL) 10.0 software for Windows and Linux. MKL is third-party software that LabVIEW uses to improve performance of linear algebra VIs. For more information regarding MKL, refer to the *Copyright* topic in the *LabVIEW Help*.

Searching Enhancements

The **Quick Drop** and **Search Palettes** dialog boxes include a new search algorithm that searches all controls, VIs, and functions by keywords and sorts the search results by relevance, instead of alphabetically by name.

VI Hierarchy Window Enhancements

LabVIEW 2009 includes the following enhancements to the **VI Hierarchy** window:

- The color of the background, the lines that connect items, and the VI borders in the **VI Hierarchy** window changed to improve visibility of items in the hierarchy.
- The **VI Hierarchy** window displays shared variables.
- The **VI Hierarchy** window shows recursive relationships by connecting the related VIs with a dotted line.
- The **VI Hierarchy** window arranges nodes in the hierarchy into groups according to the libraries to which the nodes belong. Nodes in the same library share the same border color.
- The **View** menu in the **VI Hierarchy** window includes the following new menu items: **Always Show Labels**, **Zoom In**, **Zoom Out**, **Actual Size**, and **Fit to Window**.

Dialog Box Enhancements

LabVIEW 2009 includes the following dialog box enhancements.

Class Properties Dialog Box Enhancements

LabVIEW 2009 includes the following changes to the **Class Properties** dialog box:

- On the **Item Settings** page, place a checkmark in the **Require overrides of this dynamic dispatch VI to always invoke the Call Parent Method node** checkbox to require that all dynamic dispatch VIs that override the specified VI invoke the Call Parent Method node.
- On the **Item Settings** page, place a checkmark in the **Require descendant classes to override this dynamic dispatch VI** checkbox to require all child classes to define an override member VI of the VI you select.

- On the **Inheritance** page, place a checkmark in the **Transfer all Must Override requirements to descendant classes** checkbox to allow the class to transfer all override requirements to any descendant classes instead of overriding the dynamic dispatch VI itself.
- On the **Inheritance** page, place a checkmark in the **Restrict references of this class type to member VIs of this class** checkbox to allow only member VIs to create or delete data value references to a class.
- On the **Inheritance** page, place a checkmark in the **Restrict references of descendant class types to member VIs of this class** checkbox to allow only member VIs to create or delete data value references to descendant classes of a class.

Options Dialog Box Enhancements

LabVIEW 2009 includes the following enhancements to the **Options** dialog box:

- The pages in the **Options** dialog box are organized into sections to improve usability.
- The options on the **Colors**, **Fonts**, and **Debugging** pages moved to the **Environment** page.
- The options on the **Alignment Grid** page moved to the **Front Panel** page and the **Block Diagram** page.
- The **VI Server** pages are combined into one page.
- The **Web Server** pages are combined into one page.
- Rarely used options no longer exist in the **Options** dialog box. Refer to the KnowledgeBase for a list of these deprecated options and information about alternative ways to set the options.
- The **Use numbers in icons of new VIs (1 through 9)** checkbox on the **Front Panel** page allows you to specify whether LabVIEW automatically places numbers on the icons of the first nine VIs you create after you launch LabVIEW. This option does not affect VIs you create from templates.
- The **Disable ni.com updates in Getting Started window** checkbox on the **Environment** page allows you to disable the automatic check for the latest articles on the `ni.com` Web site.
- The default number of steps for the **Maximum undo steps per VI** option on the **Environment** page changed from 9 to 30.
- The **Maximum entries in recent files lists** option on the **Environment** page allows you to specify the number of files LabVIEW lists when you select **File»Recent Files**.
- The **Advanced Settings** section of the **Web Server** page allows you to enable SSL on a non-default instance of the LabVIEW Web server.

Quick Drop Dialog Box Enhancements

LabVIEW 2009 includes the following enhancements to the **Quick Drop** dialog box:

- The **Quick Drop** dialog box displays project items by name from active LabVIEW projects. You can search for project items without navigating the **Project Explorer** window.
- While the **Quick Drop** dialog box is active, you can access the following shortcuts:
  - <Ctrl-D>—Creates controls and indicators for all unwired inputs and outputs of the selected block diagram object(s).
  - <Ctrl-Shift-D>—Creates constants for all unwired inputs of the selected block diagram object(s).
  - <Ctrl-R>—Removes the selected block diagram object(s) and any wires and constants connected to the selected object(s). This shortcut also connects wires of identical data types that were wired to the inputs and outputs of the deleted object(s).

- <Ctrl-T>—Moves the labels for all front panel control terminals to the left of the terminal on the block diagram and moves the labels for all front panel indicator terminals to the right of the terminal on the block diagram.

Miscellaneous Dialog Box Enhancements

LabVIEW 2009 includes the following miscellaneous dialog box enhancements:

- The **Navigation** window and the **Show Buffer Allocations** window are available in the LabVIEW Base Package.
- The **Servers** page of the **Web Service Properties** dialog box is renamed to the **Service Settings** page. The **Web Service Properties** dialog box configures settings for the Web Server, service aliases, and default request support.
- The **Advanced** page of the **Web Service Properties** dialog box allows you to enable remote debugging of Web service and generate log files of the Web service build.
- The **Scan Engine** page, which previously required the LabVIEW Real-Time Module, is available in the **My Computer Properties** dialog box in the LabVIEW Base Package.
- The **Browse Variables** dialog box displays shared variables you can use in a variable refnum control or constant when finding, reading, or writing shared variables programmatically.
- The **Configuration** page of the **Variable Refnum Properties** dialog box includes options you can set to configure the data type, access, and timestamp settings of a variable refnum control or constant.
- The **Edit Events** dialog box includes a new layout to improve usability.
- The **Locate Distribution** dialog box no longer exists. To copy all necessary components from a distribution into a permanent location on the computer, place a checkmark in the **Copy selected distributions to this computer during the build** checkbox on the **Additional Installers** page of the **Installer Properties** dialog box.
- The **Edit Controls and Functions Palette Set** dialog box supports the use of both LabVIEW (32-bit) and LabVIEW (64-bit) on the same computer. If you edit a palette set, LabVIEW 2009 saves the edits to the `*x*\Palettes\menus` directory, where *x* is the version of LabVIEW, in the default data directory. The version number includes either `(32-bit)` or `(64-bit)`, respectively. For example, if you edit a palette set in the 32-bit version of LabVIEW 2009, LabVIEW saves the edits to the `2009\(32-bit)\Palettes\menus` directory.

Controls Palette Enhancements

LabVIEW 2009 includes the following palette enhancements:

- The **System** palette includes the following subpalettes:
  - Numeric
  - Boolean
  - String & Path
  - Ring & Enum
  - Containers
  - List, Table & Tree
  - Decorations
- The **List & Table** palette is renamed the **List, Table & Tree** palette.
- The **Container** palette includes a subpanel control.

Miscellaneous Environment Enhancements

LabVIEW 2009 includes the following miscellaneous environment enhancements:

- The **View** menu includes the new menu item **This VI in Project**.
- The **Window** menu includes the new menu item **Show Project**.

- The **Tools»Instrumentation»Advanced Development»Show Icon Art Glossary** menu item no longer exists. Use the **Glyphs** page of the **Icon Editor** dialog box, which includes all glyphs in the Icon Library at `ni.com`, instead.
- Names for image files that LabVIEW generates include the full qualified name of the VI.

LabVIEW Application Builder Enhancements

LabVIEW 2009 includes the following enhancements to the LabVIEW Application Builder, which you can access by right-clicking **Build Specifications** in the **Project Explorer** window, and selecting the type of distribution from the shortcut menu that you want to create:

- The Application Builder no longer moves VIs with conflicting filenames outside of stand-alone applications, shared libraries, or Web services for build specifications you create in LabVIEW 2009. In LabVIEW 8.6 and earlier, the Application Builder saves VIs and library files in a flat list within the application and saves VIs with conflicting filenames outside the application in separate directories. If you build a stand-alone application or shared library using LabVIEW 2009, the Application Builder stores source files within the application using a layout similar to the directory structure of the source files on disk. For example, the following table lists the relative paths for a top-level VI, `foo.vi`, which calls `a.vi` and `b.vi`. `C:\..\Application.exe` represents the path to the application.

Path to source files	Path to files in application
`C:\Source\foo.vi`	`C:\..\Application.exe\foo.vi`
`C:\Source\xxx\a.vi`	`C:\..\Application.exe\xxx\a.vi`
`C:\Source\yyy\b.vi`	`C:\..\Application.exe\yyy\b.vi`

  To use the legacy file layout, place a checkmark in the **Use LabVIEW 8.x file layout** checkbox on the **Advanced** page of the **Application Properties**, **Shared Library Properties**, and **Web Service Properties** dialog boxes. LabVIEW enables this option by default for build specifications you load from previous LabVIEW versions.
- **(Windows)** The .NET interop assembly build specification allows you to build and distribute VIs in a LabVIEW project as .NET interop assemblies. Right-click **Build Specifications** in the **Project Explorer** window and select **New».NET Interop Assembly** from the shortcut menu to display the **.NET Interop Assembly Properties** dialog box.
- The **Build Application from VI** dialog box allows you to build an application from the VI you are currently editing.

Application Builder Dialog Box Enhancements

LabVIEW 2009 includes the following enhancements to the LabVIEW Application Builder:

- The **Generate build log file** option on the **Advanced** page of the **Application Properties**, **Shared Library Properties**, and **Web Service Properties** dialog boxes allows you to create a log file that includes information about the build, such as source files, build start and end time, and any errors that occur during the build.
- The **Shared Variable Deployment** page of the **Application Properties** dialog box allows you to specify dependent libraries containing shared variables to deploy at run time.
- The **Advanced** page of the **Source Distribution Properties** dialog box allows you to generate a log file that includes information about the build, such as source files, build start and end time, and any errors that occurr during the build.
- The **Windows Security** page of the **Application Properties**, **Shared Library Properties**, and **.NET Interop Assembly Properties** dialog boxes allows you to configure security settings for

computers running Windows. You can use this page to apply a digital signature and embed a manifest file with the build.

LabVIEW Project Enhancements

LabVIEW 2009 includes the following enhancements to the LabVIEW project and related functionality.

Miscellaneous Project Enhancements

LabVIEW 2009 includes the following miscellaneous project enhancements:

- The **Why is this item in Dependencies?** dialog box allows you to find all project items not listed under dependencies that depend on the selected item.
- The **Find Friends** dialog box allows you to to find all friends of a specific class in the project.
- The **Find Children** dialog box allows you to find all children of a specific class in the project.
- The **Find Callers** dialog box allows you to find all VIs in the project that use a shared variable node to reference a specific shared variable.
- The linkage between shared variables and their containing project libraries has been improved, eliminating problems that might occur when you save copies of complex project libraries.

New and Changed VI, Function, and Node Enhancements

LabVIEW 2009 includes the following new and changed VIs, functions, and nodes. Refer to the **VI and Function Reference** book on the **Contents** tab of the *LabVIEW Help* for more information about VIs, functions, and nodes.

New VIs and Functions

LabVIEW 2009 includes the following new VIs and functions.

Advanced Curve Fitting VIs

The **Advanced Curve Fitting** palette includes the following new VIs:

- General Polynomial Fit Coefficients
- Polynomial Fit Intervals

Application Control VIs and Functions

The **Application Control** palette includes the new **CPU Information** palette with the following new functions:

- CPU Information
- Data Cache Size
- Number of Cache Levels

Differential Equations VIs

The **Differential Equations** palette includes the new **Partial Differential Equations** palette with the following new VIs:

- Define PDE
- Define PDE Boundary Condition
- Define PDE Domain
- Define PDE Initial Condition
- PDE Rendering
- PDE Solver

File I/O VIs and Functions

The **File I/O** palette includes the following new VIs:

- Get System Directory
- Application Directory
- TDMS Convert Format
- TDMS Create Scaling Information

Fixed-Point Functions

The **Fixed-Point** palette includes the following new functions:

- Fixed-Point to Integer Cast
- Integer to Fixed-Point Cast

Matrix Functions

The **Array** and **Linear Algebra** palettes include the new **Matrix** palette with the following new functions:

- Build Matrix
- Get Matrix Diagonal
- Get Matrix Elements
- Get Submatrix
- Matrix Size
- Resize Matrix
- Set Matrix Diagonal
- Set Matrix Elements
- Set Submatrix
- Transpose Matrix

Memory Control VIs, Functions, and Nodes

The **Memory Control** palette includes the following new functions:

- Delete Data Value Reference
- New Data Value Reference

Shared Variable VI, Functions, and Node

The **Shared Variable** palette includes the following new functions:

- Close Variable Connection
- Open Variable Connection
- Read Variable
- Write Variable

The **Shared Variable** palette includes the new **I/O Variable** palette with the following new functions:

- Direct Variable Read
- Direct Variable Write
- Scanned Variable Read
- Scanned Variable Write

The **Shared Variable** palette includes the new **PSP Variable** palette with the following new functions:

- Open and Verify Variable Connection
- Open Variable Connection in Background

Signal Manipulation Express VIs

The **Signal Manipulation** palette includes the following new Express VIs:

- Get Dynamic Data Attributes
- Group Digital Signals
- Set Dynamic Data Attributes

Utility VIs

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

- LV Image to PNG Data
- PNG Data to LV Image

XML Parser VIs and Functions

The **XML Parser** palette includes the following new VIs:

- Get All Matched Nodes
- Get First Matched Node

Miscellaneous New VIs and Functions

LabVIEW 2009 includes the following miscellaneous new VIs and functions:

- The **Cluster, Class, & Variant** palette includes the Preserve Run-Time Class function.
- The **Signal Operation** palette includes the Z-Transform Delay Node, which is the Feedback Node in z-transform view.
- The **Fitting** palette includes the B-Spline Fit VI.
- The **Web Services** palette includes the new Read Uploaded Files Info VI.

Changed VIs, Functions, and Nodes

The following VIs, functions, and nodes changed in LabVIEW 2009.

Advanced IIR Filtering VIs

The **Advanced IIR Filtering** palette includes the following changed VIs.

The following VIs include a **filter structure option** input, which specifies the order of the IIR cascade filter:

- Bessel Coefficients
- Butterworth Coefficients
- Chebyshev Coefficients
- Elliptic Coefficients
- Inv Chebyshev Coefficients

Connectivity VIs and Functions (Windows)

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

- The Create Registry Key VI includes a **registry view** input, which specifies whether to create a 32-bit key or a 64-bit key on a 64-bit operating system.

Note The following VIs are available only in the LabVIEW Full and Professional Development Systems.

- **Set ESP Variable**—Includes polymorphic instances that accept string, Boolean, and numeric data types.

- **Write Session Variable**—Includes polymorphic instances that accept string, Boolean, and numeric data types.

Digital Waveform VIs and Functions

The **Digital Waveform** palette includes the following changed VIs:

- **Append Digital Signals**—The DWDT instance combines the names of the digital signals.
- **Digital Signal Subset**—The **DWDT** instance returns the names of the signals in the digital waveform subset if the signals in the subset have names.

File I/O VIs and Functions

The **File I/O** palette includes the following changed VIs:

- **Close Config Data**—The **write configuration file?** input is renamed **write file if changed**.
- **Get Key Names**—The **found?** output is renamed **section exists?**.
- **Open Config Data**—Includes a **file created?** output that returns TRUE if the configuration file operation creates a file. Also, the **create file if necessary?** input is renamed **create file if necessary**.
- **Recursive File List**—Includes a **Continue Recursing on Error?** input that specifies whether to continue recursing directories if an error occurs. This VI also includes a **Folders to Exclude** input that specifies folder names to exclude during the recursion.
- **Remove Section**—The **found?** output is renamed **section exists?**.
- Read From Measurement File—The **Reopen file** input is renamed **Reset**.
- **Write Key**—Includes a **found?** output that is TRUE if the VI found the key in the specified **section**.

Fitting VIs

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

- **Exponential Fit**—Includes a **parameter bounds** input that contains the upper and lower constraints for the **amplitude**, **damping**, and **offset**. This VI also includes an **offset** output that returns the offset of the fitted model. This VI no longer includes a **refine?** input.
- **Gaussian Peak Fit**—Includes a **parameter bounds** input that contains the upper and lower constraints for the **amplitude**, **center**, **standard deviation**, and **offset**. This VI also includes an **offset** output that returns the offset of the fitted model.
- **General Linear Fit**—Includes a **tolerance** input that determines when to stop the iterative adjustment of **Coefficients**. This VI also includes a **method** input that specifies the fitting method. The name of this VI also changed from General LS Linear Fit to General Linear Fit.
- **General Polynomial Fit**—Includes a **tolerance** input that determines when to stop the iterative adjustment of **Polynomial Coefficients**. This VI also includes a **method** input that specifies the fitting method.
- **Linear Fit**—Includes a **parameter bounds** input that contains the upper and lower constraints for the **slope** and **intercept**.
- **Logarithm Fit**—Includes a **parameter bounds** input that contains the upper and lower constraints for the **amplitude** and **scale**.
- **Power Fit**—Includes a **parameter bounds** input that contains the upper and lower constraints for the **amplitude**, **power**, and **offset**. This VI also includes an **offset** output that returns the offset of the fitted model.

Mathematics VIs

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

- **Integral x(t)**—Includes an **integration method** input, which specifies the method to use to perform the numeric integration. Also, the **Initial Condition** and **Final Condition** inputs changed to specify the initial and final conditions, respectively, of **X** in the integration calculation based on **integration method**. The data types of **Initial Condition** and **Final Condition** changed from double-precision, floating-point numbers to 1D arrays of double-precision, floating-point numbers.

- **Matrix Characteristic Polynomial**—Is a polymorphic VI with the following instances: Real Characteristic Polynomial and Complex Characteristic Polynomial.
- **Numeric Integration**—The data type of the **integration method** input changed to a 32-bit signed integer for all three instances of the VI.
- **Quadrature**—Includes the following new instances: 2D Quadrature (VI), 2D Quadrature (Formula), 3D Quadrature (VI), and 3D Quadrature (Formula).
- **(Riemann) Zeta Function**—The **x** input and the **z(x)** output changed to complex double-precision data types.
- **Test Matrix Type**—Allows you to specify a `Symmetric` or `Hermitian` type in the **matrix type** input.

Memory Control VIs, Functions, and Nodes

The **Memory Control** palette includes the following changed structure.

The In Place Element structure includes an Array Split / Replace Subarrays border node and a Data Value Reference Read / Write Element border node.

Report Generation VIs

The **Report Generation** VIs palette includes the following changed VIs:

- **Easy Text Report**—The **print or save? (Word/Excel)** input is renamed to **print or save? (Word/Excel/HTML)**.
- **Dispose Report**—The **save changes?** input is optional.

Signal Generation VIs

The **Signal Generation** palette includes the following changed VIs.

The following VIs include an **initialize?** input, which controls the reseeding of the noise sample generator. Also, the **seed** input changed to determine how to generate the internal seed state when **initialize?** is TRUE.

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

Sound VIs (Mac OS)

The VIs on the **Sound** palette on Mac OS include the following changes:

Note LabVIEW 2009 supports the same API for Windows, Mac OS, and Linux.

- 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 signed integers, or single and double-precision data 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 include improvements to error checking.

TDM Streaming VIs and Functions

The **TDM Streaming** palette includes the following changed functions:

- **TDMS Open**—Includes a **disable buffering** input that specifies whether LabVIEW opens, creates, or replaces a `.tdms` file without system buffering. This function also includes a **byte order** input that specifies the endian format of the data in a `.tdms` file.

  This function also includes a **file format version** input that specifies the version of a `.tdms` file. The file format version 2.0 includes all features from version 1.0, as well as the following additional features:

  – You can write interleaved data to a `.tdms` file.
  – You can write `.tdms` data in different endian formats, or byte orders.

  Use the **TDMS Convert Format** VI to convert the file format version of a `.tdms` file from 1.0 to 2.0 or vice versa.
- **TDMS Write**—Includes a **data layout** input that specifies the arrangement of the data you want to stream to a `.tdms` file.

Waveform Measurements VIs

The **Waveform Measurements** palette includes the following changed VIs:

- **Extract Multiple Tone Information**—Includes the following new instances: Extract Multiple Tone Information 1 Chan (CDB), Extract Multiple Tone Information N Chan (CDB), and Extract Multiple Tone Information N Channels - N Specs (CDB).
- **Extract Single Tone Information**—Includes the following new instances: Extract Single Tone Information 1 Chan (CDB) and Extract Single Tone Information N Chan (CDB).

Zeros VIs

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

- **Find All Zeros of f(x)**—Is a polymorphic VI with the following instances: Find All Zeros of f(x) (Formula) and Find All Zeros of f(x) (VI). The Find All Zeros of (fx) (Formula) instance has the same functionality as the Find All Zeros of f(x) VI in LabVIEW 8.6.
- **nD Nonlinear System Single Solution**—Is a polymorphic VI with the following instances: nD Nonlinear System Single Solution (Formula) and nD Nonlinear System Single Solution (VI). The nD Nonlinear System Single Solution (Formula) instance has the same functionality as the nD Nonlinear System Single Solution VI in LabVIEW 8.6.
- **nD Nonlinear System Solver**—Is a polymorphic VI with the following instances: nD Nonlinear System Solver (Formula) and nD Nonlinear System Solver (VI). The nD Nonlinear System Solver (Formula) instance has the same functionality as the nD Nonlinear System Solver VI in LabVIEW 8.6.
- **Newton Raphson Zero Finder**—Is a polymorphic VI with the following instances: Newton Raphson Zero Finder (Formula) and Newton Raphson Zero Finder (VI). The Newton Raphson Zero Finder (Formula) instance has the same functionality as the Newton Raphson Zero Finder VI in LabVIEW 8.6.

- **Ridders Zero Finder**—Is a polymorphic VI with the following instances: Ridders Zero Finder (Formula) and Ridders Zero Finder (VI). The Ridders Zero Finder (Formula) instance has the same functionality as the Ridders Zero Finder VI in LabVIEW 8.6.

Miscellaneous VI, Function, and Node Changes

LabVIEW 2009 includes the following miscellaneous VI, function, and node changes:

- The **n or n-1 dim array** input of the Insert Into Array function changed so that if **n-dim array** is an array of references, **n or n-1 dim array** must be a reference or array of references from the same child class as **n-dim array**.
- The **3D Graph Properties** VIs, the **Cursor** VIs, the **Graphics Formats** VIs, the **Picture Functions** VIs, the **Picture Plots** VIs, and the **SMTP Email** VIs are available in the LabVIEW Base Package.
- The icons for the New and Close VIs changed.
- The Feedback Node includes new configuration options. You can display the node in z-transform view, specify the number of block diagram executions or loop iterations to delay the output of the node, enable or disable the node, and specify when the node initializes.
- The behavior of the **Configuration File** VIs and how they read data from `.ini` files for section and key inputs changed. The key input does not accept a semicolon as the first character of the key name, and the VIs do not accept a key input without a value input. Also, a closed bracket is not supported in the section input, and the VIs do not produce modified comments.
- The **NI Scan Engine** VIs, which previously installed with the LabVIEW Real-Time Module, are available in the LabVIEW Base Package.

New and Changed Classes, Properties, Methods, and Events

LabVIEW 2009 includes new VI Server classes, properties, methods, and events. Refer to the **LabVIEW 2009 Features and Changes»New VI Server Objects** topic on the **Contents** tab of the *LabVIEW Help* for a list of new classes, properties, methods, and events.

3DPC_SurfacePlot Properties and Methods

The 3DPC_SurfacePlot class includes the new Cursor List property.

Math Plots Properties and Methods

Refer to the **Property and Method Reference»Math Plots** book on the **Contents** tab of the *LabVIEW Help* for a list of new properties and methods you can use with the new 2D and 3D graphs.

Variable Properties and Methods

LabVIEW 2009 includes new Variable properties and methods you can use to find, read, and write shared variables programmatically.

VI Server Properties and Methods

LabVIEW 2009 includes the following VI Server method changes:

- Get VI Version—Returns the LabVIEW file format version of the VI. This version might be different from the version of LabVIEW in which the VI was last saved. You can use the Get VI Editor Version method to retrieve the version of LabVIEW that last saved the VI.
- Library:Get File LabVIEW Version—Returns the LabVIEW file format version of the library. This version might be different from the version of LabVIEW in which the library was saved.
- Source Scope:Set—Includes the community access scope.

XML Parser Properties and Methods

LabVIEW 2009 includes the following new XML Parser properties:

- The Node class includes the new Get XML (Pretty Print) method.

- The Document class includes the new Save File (Pretty Print) method.

LabVIEW Object-Oriented Programming Enhancements

In LabVIEW 2009 you can deploy VIs that use LabVIEW classes or that call member VIs of a LabVIEW class to RT targets. If you deploy a VI that uses a class or calls a member VI, only the VIs and classes referenced in the application deploy to the target. If the deployed VIs reference any dynamic dispatch member VIs, all the VIs that override the dynamic dispatch member VIs in descendant classes also deploy.

LabVIEW Web Services Enhancements (Windows, Not in Base Package)

LabVIEW 2009 includes the following enhancements to Web services:

- The **Service aliases** option on the **Service Settings** page of the **Web Service Properties** dialog box can act in place of the **Service name** in the custom browsing URL.
- When you add a Web method VI to the **Service VIs** list on the **Source Files** page of the **Web Service Properties** dialog box, LabVIEW creates a default URL mapping on the **URL Mappings** page of the **Web Service Properties** dialog box. You can use, modify, or delete this default URL.
- Web services support additional output data types, including clusters, arrays, waveforms, and timestamps.
- You can upload files to Web services.
- Web services support JSON output formatting. Configure the output format type in the **Configure RESTful VI** dialog box.
- You can create a custom browsing URL that assigns multiple fragments of the URL to a single input terminal of a Web method VI. This technique is useful for passing file paths or other hierarchical values using a custom browsing URL.
- You can configure a default HTML document (`index.html`) to load from static document folders.
- You can enable debugging of a Web method VI on real-time targets.
- The NI Distributed System Manager includes enhanced statistics for deployed Web services.
- You can configure a default Web method VI for a Web service.

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

SSL Support for the LabVIEW Web Server

You can enable Secure Sockets Layer (SSL) encryption on the LabVIEW Web Server. SSL encryption allows you to create secure, encrypted connections for communication between clients and the Web Server. SSL is useful for Web server features such as Web services, remote front panels, and publishing static images and documents.

SSL uses X.509 certificates to establish encrypted connections between a client and the Web Server. You can use the default LabVIEW self-signed certificate, or using the NI Distributed System Manager, you can create custom self-signed certificates and certificate signing requests (CSR) to be digitally signed by a certificate authority (CA).

Comparing VIs Using the Command Line or a Third-Party Source Control Provider

`LVCompare.exe` is a command-line executable that compares two VIs you specify. You can invoke `LVCompare.exe` from the command line of a computer or from a third-party source control provider. Use the following command-line syntax: <*`absolute path to VI 1`*><*`absolute path to VI 2`*> [*`-lvpath <path to LabVIEW>`*][*`-noattr`*][*`-nofp`*][*`-nofppos`*][*`-nobd`*][*`-nobdcosm`*][*`-nobdpos`*]

Note `LVCompare.exe` is available only in the LabVIEW Professional Development System.

Refer to the **Fundamentals»Development Guidelines»How-To»Comparing VIs** book on the **Contents** tab in the *LabVIEW Help* for more information about comparing VIs.

Debugging with the Probe Watch Window

The **Probe Watch Window** replaces individual floating **Probe** windows as a unified tool to view and manage all probes in the current instance of LabVIEW.

Icon Editor Enhancements

In LabVIEW 8.6 and earlier, the **Icon Editor** dialog box includes basic editing tools with which you can create 256-color, 16-color, and black-and-white icons. You create each type of icon individually.

In LabVIEW 2009, the **Icon Editor** dialog box provides an enhanced set of editing tools for creating icons. The **Icon Editor** dialog box provides icon templates, glyphs from the Icon Library at `ni.com`, options for adding and formatting icon text, and support for editing with layers. The **Icon Editor** dialog box saves icons you create in both 256-color and black-and-white format. You also can use the **Icon Editor** dialog box to create and save icon templates or glyphs for later use. You can save these images as 256-color `.png` files.

To edit the icon for a VI or custom control, double-click the icon in the upper right corner of the front panel window, block diagram window, or Control Editor window to display the **Icon Editor** dialog box. You also can right-click the icon and select **Edit Icon** from the shortcut menu to display the **Icon Editor** dialog box. To edit the icon for a project library, statechart, class, or XControl, click the **Edit Icon** button on the **General Settings** page of the corresponding **Properties** dialog box to display the **Icon Editor** dialog box.

Refer to the **Fundamentals»Creating VIs and SubVIs** book on the **Contents** tab in the *LabVIEW Help* for more information about creating icons.

LabVIEW 2009 (64-bit)

LabVIEW 2009 introduces a 64-bit version of the LabVIEW Development System. When run on Windows Vista (64-bit version), LabVIEW 2009 (64-bit) provides access to more memory than either a 32-bit operating system or a 32-bit application can provide. LabVIEW 2009 (64-bit) includes nearly all of the development environment features of LabVIEW 2009 (32-bit), including the LabVIEW Application Builder.

Refer to the National Instruments Web site at `ni.com/info` and enter the info code `lv64bit` for information about obtaining a copy of LabVIEW 2009 (64-bit).

Supported Hardware

LabVIEW 2009 (64-bit) supports many hardware devices. Drivers are available for DAQ devices, VISA devices, GPIB devices, and image acquisition devices. For GPIB devices, you must use at least NI-488.2 version 2.6 for Windows. Refer to the specific hardware documentation for more information about compatibility with LabVIEW 2009 (64-bit).

Supported Modules

LabVIEW 2009 (64-bit) supports only the NI Vision Development Module. Refer to the Vision Development Module documentation for more information. LabVIEW 2009 (64-bit) does not support any additional modules, toolkits, or add-ons.

Recursion

You can use recursion in LabVIEW 2009 if all VIs in the VI hierarchy are reentrant and at least one of the VIs shares clones of itself between calls. After you configure a VI to be reentrant, you can drag the icon of the VI onto its own block diagram.

Reusing Sections of Code

You can save sections of code, or VI snippets, from the block diagram to reuse later or to share with other LabVIEW 2009 users. When you save a section of code, LabVIEW embeds the code into a `.png` image file. The image shows a picture of the code and also contains the actual code.

To save a section of code, select the section of code you want to save and then select **Edit»Create VI Snippet from Selection**. After you save the VI snippet as a `.png` file, you can drag the file from the directory where you saved it and drop the file onto the block diagram, or you can share the file with other LabVIEW 2009 users so they can use the code.

Using the TDM Excel Add-In (Windows)

Use the TDM Excel Add-In to load `.tdm` and `.tdms` files into Microsoft Excel. From a toolbar in Excel, choose which properties load into Excel at the file, group, and channel levels, including custom properties.

371780E-01 Jun09