LabVIEW™ Upgrade Notes

These upgrade notes describe the process of upgrading LabVIEW for Windows, Mac OS X, and Linux to LabVIEW 2010, 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 a previous version of LabVIEW to LabVIEW 2010, National Instruments recommends that you also review the upgrade notes for each LabVIEW version between the version from which you are upgrading and LabVIEW 2010. The following upgrade notes documents contain information about enhancements, changes, and features added in recent versions of LabVIEW:

- **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*.
- **LabVIEW 2009 Upgrade Notes**—The *Upgrade and Compatibility Issues* section and the *LabVIEW 2009 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 `upnote9` to access the *LabVIEW 2009 Upgrade Notes*.

Refer to the *LabVIEW Help* for more information about LabVIEW 2010 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»LabVIEW Help**.

Contents

Upgrading to LabVIEW 2010

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 2010**—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 2010.
   - **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 `lvupgradetests` to download tests that can evaluate VIs for some compatibility issues.

   - **LabVIEW 2010 Features and Changes**—This section includes brief descriptions of the new features in this version of LabVIEW. Refer to the *LabVIEW Help* for more complete information about using these features. Access the *LabVIEW Help* by selecting **Help»LabVIEW Help**.
2. (Optional) Uninstall any previous version(s) of LabVIEW.
3. Install and activate the 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 2010* and the appropriate subsection for the platform on which you are installing

- *Installing LabVIEW Add-Ons* if you are installing LabVIEW toolkits or modules from media other than the LabVIEW Platform DVDs
- **(Windows)** *Activating the LabVIEW License* and all subsections
- (Optional) *Installing and Configuring Hardware* and the appropriate subsection for the platform on which you are installing
- *Where to Go from Here*

4. Refer to the *LabVIEW Readme* for issues fixed in the new version of LabVIEW, information about known issues in the new version of LabVIEW, and documentation additions that are not reflected in the *LabVIEW Help*. To access the *LabVIEW Readme*, navigate to the `labview\readme` directory and open the `readme.html` file.
5. Copy environment settings from a previous version of LabVIEW. Refer to the *Copying Environment Settings from a Previous Version of LabVIEW* section of this document for more information about copying environment settings.
6. Copy `user.lib` files from a previous version of LabVIEW. Refer to the *Copying user.lib Files from a Previous Version of LabVIEW* section of this document for more information about copying `user.lib` files.
7. Convert VIs to LabVIEW 2010. Refer to the *Converting VIs* section of this document for more information converting VIs saved in a previous version of LabVIEW.

Upgrading from Previous Versions of LabVIEW

You can install LabVIEW 2010 without uninstalling previous versions of LabVIEW. While versions of LabVIEW might share components, upgrading to a new version of LabVIEW does not affect the performance of previous versions of LabVIEW on the computer because the new version installs in a different directory. LabVIEW 5.*x* and earlier install in the `labview` directory. LabVIEW 6.0 and later install in the `labview x` directory, where *x* is the version number.

Replacing an Existing Version of LabVIEW

To replace your existing version of LabVIEW, uninstall the existing version of LabVIEW, run the LabVIEW 2010 installer, and set the installation directory to the `National Instruments` directory where you installed the previous version of LabVIEW.

(Windows) You also can replace the existing version of LabVIEW with LabVIEW 2010 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 top-level `labview` directory.

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

Copying Environment Settings from a Previous Version of LabVIEW

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

Caution If you replace the LabVIEW 2010 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 2010, copy the LabVIEW preferences file to the LabVIEW 2010 directory.

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

(Mac OS X) LabVIEW stores preferences in the LabVIEW preferences text file at `~/Library/Preferences/LabVIEW 10.0 Preferences`.

(Linux) LabVIEW stores preferences at `/home/<username>/natinst/.config/LabVIEW-2010/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 2010 directory, you must manually change the preferences to match the new format if necessary.

Copying user.lib Files from a Previous Version of LabVIEW

To use files from the `user.lib` directory of a previous version of LabVIEW, copy the files from the `labview` directory in which the previous version is installed. After you install LabVIEW 2010, copy the files to the `user.lib` directory in the LabVIEW 2010 directory.

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 2010. To open a VI saved in LabVIEW 4.0 to 5.*x*, you first must open and save the VI in any LabVIEW version from 6.0 to 8.2.1 and then reopen the VI in LabVIEW 2010. When you open a VI last saved in LabVIEW 6.0 or later, LabVIEW 2010 automatically converts and compiles the VI. You must save the VI in LabVIEW 2010, 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 2010 do not load in earlier versions of LabVIEW. To maintain compatibility with an earlier version, before you save VIs in LabVIEW 2010 after you convert them, keep a backup copy of VIs you plan to use in the previous version. You also can select **File»Save for Previous Version** to save VIs so they can run in a previous version.

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

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

Upgrading Modules, Toolkits, and Instrument Drivers

If you are upgrading from a previous version of LabVIEW, you must install current, compatible versions of any modules, toolkits, or instrument drivers that you installed for the previous version of LabVIEW. The LabVIEW Platform DVDs include most modules and toolkits that are compatible with LabVIEW 2010. 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.

Table 1. Types of LabVIEW Installation Media

Operating System	Media to Use	Important Notes
Windows	DVD	Use the LabVIEW Platform DVDs to install LabVIEW 2010 and versions of modules and toolkits that are compatible with LabVIEW 2010. 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 2010 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 2010 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 2010. 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 2010 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 2010 support using one of the following methods:

- **NI Modular Instrument drivers**—Use the NI Device Drivers DVD or CD to install NI Modular Instrument drivers.
- **Plug and Play instrument drivers—(Windows, Linux)** Use the NI Instrument Driver Finder to search for and install LabVIEW Plug and Play instrument drivers without leaving the LabVIEW development environment. Select **Help»Find Instrument Drivers** to launch the Instrument Driver Finder.
- **IVI driver or non-certified instrument drivers**—Use the Instrument Driver Network on the National Instruments Web site to search for and install an IVI driver or a non-certified driver.

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 2010 for your operating system. Make sure you mass compile any VIs that are related to the add-on.

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

Mass Compiling LabVIEW

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

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

Upgrading Additional National Instruments Software

There are known compatibility issues with LabVIEW 2010 and TestStand 4.2.1 and earlier verisons. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `exvaku` to access the KnowledgeBase article that details these issues.

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

You must use NI Spy 2.3 or later in LabVIEW 2010. NI Spy 2.7.2 is available on the National Instruments Device Drivers CD.

LabVIEW 2010 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.

Upgrade and Compatibility Issues

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

Refer to the `readme.html` file in the `labview` directory for information about known issues in the new version of LabVIEW, additional compatibility issues, and information about late-addition features in LabVIEW 2010. 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 2009

You might encounter the following compatibility issues when you upgrade to LabVIEW 2010 from LabVIEW 2009.

Platforms Supported

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

Disk Space Requirements

(Windows) LabVIEW 2010 requires at least 1.6 GB of disk space for the LabVIEW installation.

(Mac OS X) LabVIEW 2010 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 2010 requires at least 680 MB of disk space for the minimum LabVIEW installation or 890 MB of disk space for the complete LabVIEW installation.

Refer to the *LabVIEW Release Notes* for additional system requirements for LabVIEW 2010.

VI and Function Behavior Changes

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

- Elliptic Coefficients
- Elliptic Filter
- Elliptic Filter PtByPt

Deprecated VIs, Functions, and Nodes

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

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

Floating-Point Math Operations

Due to changes to the LabVIEW compiler, the results of several mathematical operations performed using floating-point numbers might differ from results returned in previous versions of LabVIEW. The accuracy of algorithms written in LabVIEW using floating-point numbers is the same and in many cases improved in LabVIEW 2010. However, in a few operations the results might be less accurate than in previous versions because LabVIEW 2010 implements functions internally with the same numeric precision as the input data types rather than a higher numeric precision than the input data types as in previous versions. The acceptable error for the results of these operations is still appropriate for the data types of the inputs.

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

Creating LabVIEW Classes

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

Building an Installer (Windows)

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

Upgrading from LabVIEW 8.6

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

Disk Space Requirements

(Windows) LabVIEW 2009 and 2010 both require at least 1.6 GB of disk space for the LabVIEW installation.

(Mac OS X) LabVIEW 2009 and 2010 both require 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. LabVIEW 2010 requires at least 680 MB of disk space for the minimum LabVIEW installation or 890 MB of disk space for the complete LabVIEW installation.

Refer to the *LabVIEW Release Notes* for additional system requirements for LabVIEW 2010.

VI and Function Behavior Changes

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

Bluetooth VIs and Functions

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

Signal Generation VIs

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

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

Miscellaneous VI and Function Behavior Changes

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

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

must redeploy any Web services created in a previous version of LabVIEW to use the Web services in LabVIEW 2009 or later. To delete a Web service deployed by a previous version of LabVIEW, you must manually remove it from the deployed location.

- The Integral x(t) VI is rewritten. To use the new functionality, replace this VI with the Integral x(t) VI from the **Functions** palette.
- The **section** and **refnum** inputs of the Get Key Names VI are required inputs.
- The **refnum** input of the Get Section Names VI is a required input.
- The **refnum** input of the Not A Config Data Refnum VI is a required input.

Deprecated VIs and Functions

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

- **LToCStr**—Use the LToCStrN function instead. The LToCStrN function differs from the LToCStr function because it takes a parameter specifying the size of the C string buffer to which LabVIEW copies the string. These functions are Code Interface Node (CIN) functions. The Code Interface Node is deprecated in LabVIEW 2010. Use the Call Library Function Node instead.
- **Open Config Data (compatibility)**—Use the Open Config Data VI instead. The Open Config Data VI differs from the Open Config Data (compatibility) VI because the Open Config Data VI includes the **file created?** output.
- **Sound VIs (Mac OS X)**—Use the **Sound** VIs instead. LabVIEW 2009 and later supports the same API for Windows, Mac OS X, and Linux.
- **Unconstrained Exponential Fit**—Use the Exponential Fit VI instead. The Exponential Fit VI differs from the Unconstrained Exponential Fit VI because the Exponential Fit VI does not include the **refine?** input but does include the **parameter bounds** input and **offset** output.
- **Unconstrained Gaussian Peak Fit**—Use the Gaussian Peak Fit VI instead. The Gaussian Peak Fit VI differs from the Unconstrained Gaussian Peak Fit VI because the Gaussian Peak Fit VI includes the **parameter bounds** input and **offset** output.

Deprecated Properties, Methods, and Events

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

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

Renamed Properties, Methods, and Events

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

Table 2. Renamed Properties, Methods, and Events in LabVIEW 2009

Class	LabVIEW 8.6 Name	LabVIEW 2009 and Later Name	Type
Document	Do Namespaces	Process Namespaces	Property
Document	Do Schema	Process Schema	Property
Variable	Alarming:BadStatus:AckType	Alarming:BadStatus:Ack Type	Property
Variable	Alarming:BadStatus:AllowLog	Alarming:BadStatus:Allow Log	Property

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

Class	LabVIEW 8.6 Name	LabVIEW 2009 and Later Name	Type
Variable	Real-Time:DatapointsInWaveform	Real-Time:Datapoints In Waveform	Property
Variable	Real-Time:UseBuffering	Real-Time:Use Buffering	Property
Variable	Scaling:EngineeringMax	Scaling:Engineering Max	Property
Variable	Scaling:EngineeringMin	Scaling:Engineering Min	Property
Variable	Scaling:InvertMask	Scaling:Invert Mask	Property
Variable	Scaling:RawMax	Scaling:Raw Max	Property
Variable	Scaling:RawMin	Scaling:Raw Min	Property
Variable	Scaling:SelectMask	Scaling:Select Mask	Property

Application Builder Changes

LabVIEW 2009 and later includes the following Application Builder changes.

File Layout Changes

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

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

Custom Configuration File Changes

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

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

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

Case Structure Output Tunnel Changes

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

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

Custom Icon Editor VI Changes

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

Custom Probes Changes (Linux)

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

.NET Changes

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

LabVIEW MathScript Changes

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

Upgrading from LabVIEW 8.5

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

Platforms Supported

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

Disk Space Requirements

(Windows) LabVIEW 8.6 and LabVIEW 2010 both require at least 1.6 GB of disk space for the LabVIEW installation.

(Mac OS X) LabVIEW 8.6 requires at least 262 MB of disk space. LabVIEW 2010 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 2010 requires at least 680 MB of disk space for the minimum LabVIEW installation or 890 MB of disk space for the complete LabVIEW installation.

Refer to the *LabVIEW Release Notes* for additional system requirements for LabVIEW 2010.

VI and Function Behavior Changes

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

Report Generation VIs

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

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

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

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

External Code (DLLs and CINS)

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

Miscellaneous VI and Function Behavior Changes

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

- The STFT Spectrograms VI was rewritten with two new inputs in LabVIEW 8.6 and later. Replace versions of this VI from previous versions of LabVIEW with an STFT Spectrograms VI from the **Functions** palette to use the new functionality.
- Many of the Mathematics and Signal Processing VIs changed from non-reentrant VIs to reentrant VIs. Because of these changes, you should not call many of these VIs from a reentrant VI set to share clones between instances. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `exrehi` for more information about which VIs you cannot call from a VI set to share clones between instances.
- LabVIEW 8.6 and later forces single-process shared variables to be target-relative. You cannot configure single-process shared variables to be absolute.
- If you wire an empty path to the **path in** input of the Call Library Function Node, LabVIEW no longer returns an error.
- The **output element** output of the Get Report Type VI changed to **report type**. LabVIEW automatically renames and reconnects code you take from previous LabVIEW versions and insert into LabVIEW 8.6 and later. However, the VI breaks if you use the Call By Reference Node to call the Get Report Type VI.
- The **report type** input of the New Report VI is a required input. You must wire data to this input.

Deprecated VIs and Functions

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

- **Nonlinear System Single Solution**—Use the nD Nonlinear System Single Solution VI instead. The nD Nonlinear System Single Solution VI differs from the Nonlinear System Single Solution VI because it is reentrant.

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

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

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

Property, Method, and Event Behavior Changes

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

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

Deprecated Properties, Methods, and Events

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

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

Renamed Properties, Methods, and Events

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

Table 3. Renamed Properties, Methods, and Events in LabVIEW 8.6

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

Shared Variable Changes

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

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

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

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 receive an error because the shared component is missing functionality that LabVIEW 8.6 or later relies on. To correct this issue, reinstall LabVIEW 8.6 or later.

Saving Password-Protected VIs for Previous Versions

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

Upgrading from LabVIEW 8.2

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

Platforms Supported

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

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

Disk Space Requirements

(Windows) LabVIEW 8.5 requires at least 1.2 GB of disk space for the LabVIEW installation. LabVIEW 2010 requires at least 1.6 GB of disk space for the LabVIEW installation.

(Mac OS X) 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 2010 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 2010 requires at least 680 MB of disk space for the minimum LabVIEW installation or 890 MB of disk space for the complete LabVIEW installation.

Refer to the *LabVIEW Release Notes* for additional system requirements for LabVIEW 2010.

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 the Find All Zeros of f(x) VI.

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

Numeric Functions

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

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

Signal Processing VIs

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

Hyperbolic Functions

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

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

Libraries & Executables VIs and Functions

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

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

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

Miscellaneous VI 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.

Property, Method, and Event Behavior Changes

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

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

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

Deprecated Properties, Methods, and Events

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

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

Renamed Properties, Methods, and Events

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

Table 4. Renamed Properties, Methods, and Events in LabVIEW 8.5

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

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

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

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

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

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

LabVIEW Class Icons

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

Opening LLBs in LabVIEW

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

Timed Loop Priority Level Restriction

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

Waveform Data Type

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

Enum Coercion

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

Upgrading from LabVIEW 8.0 or Earlier

Refer to the National Instruments Web site at `ni.com/info` and enter the info code `exc6mf` to access upgrade and compatibility issues you might encounter when upgrading from LabVIEW 8.0 or earlier to the current version. Also, refer to the *Upgrading from LabVIEW 8.2*, *Upgrading from LabVIEW 8.5*, *Upgrading from LabVIEW 8.6*, and *Upgrading from LabVIEW 2009* sections of this document for information about other upgrade issues you might encounter.

LabVIEW 2010 Features and Changes

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

Refer to the *LabVIEW Help* for more information about LabVIEW 2010 features, including programming concepts, step-by-step instructions, and reference information. Access the *LabVIEW Help* by selecting **Help»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 2010.

Installing LabVIEW

(Windows) With LabVIEW 2010 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 2010* section of the *LabVIEW Release Notes* for more information.

New Example VIs

Refer to the **New Examples for LabVIEW 2010** folder on the **Browse** tab of the NI Example Finder to view descriptions for and launch example VIs added to LabVIEW 2010.

Block Diagram Enhancements

LabVIEW 2010 includes the following enhancements to the block diagram and related functionality.

Creating Owned Labels on Wires

You can use an owned label to label a long wire and identify its use. To create an owned label on a wire, right-click the wire and select **Visible Items»Label** from the shortcut menu. Labeling wires is useful for wires coming from shift registers and for long wires that span the entire block diagram. You can move an owned label anywhere along the wire. You cannot lock an owned label to a wire so that the label remains in the same position on the wire. [*Idea submitted by NI Discussion Forums member falkpl.*]

Note If you save a VI that contains owned labels on wires to a previous version of LabVIEW, LabVIEW replaces the owned labels with free labels.

Inlining SubVIs to Eliminate SubVI Overhead

You can inline subVIs into their calling VIs to eliminate subVI overhead and increase code optimization in calling VIs. When you inline a subVI, LabVIEW inserts the compiled code of the subVI into the compiled code of the calling VI. If you then make changes to the subVI, LabVIEW recompiles all calling VIs of that subVI to include those changes. Essentially, inlining a subVI removes the need to call the subVI at run time. Instead, LabVIEW executes the subVI code inside the compiled code of the calling VI.

Inlining subVIs is most useful for small subVIs, subVIs within a loop, VIs with unwired outputs, or subVIs you call only once. To inline a subVI, place a checkmark in the **Inline subVI into calling VIs** checkbox on the **Execution** page of the **VI Properties** dialog box.

Refer to the **Fundamentals»Managing Performance and Memory»Concepts»VI Execution Speed** topic on the **Contents** tab of the *LabVIEW Help* for more information about inlining subVIs.

Setting Iteration Schedules on Parallel For Loops

You can specify how LabVIEW assigns chunks of loop iterations to each loop instance when you enable parallel loop iterations on a For Loop. Right-click a For Loop and select **Configure Iteration Parallelism** from the shortcut menu to display the **For Loop Iteration Parallelism** dialog box. Select one of the following options from the **Iteration partitioning schedule** section of this dialog box to specify how LabVIEW assigns chunks:

- **Automatically partition iterations**—Divides loop iterations into chunks of loop iterations automatically.
- **Specify partitioning with chunk size (C) terminal**—Divides loop iterations into constant-sized chunks. The value you wire to the chunk size (**C**) terminal specifies the chunk size. You can wire an array to the chunk size terminal to specify a size for each chunk of iterations. If you wire too many chunk sizes, LabVIEW ignores the extra elements. If you wire too few chunk sizes, LabVIEW uses the last element in the array to determine the size of all remaining chunks of iterations.

Note If you open a VI that contains a For Loop with parallel loop iterations from a previous version of LabVIEW, LabVIEW updates the For Loop to divide loop iterations into chunks of loop iterations automatically.

Wiring a Constant of 0 to a For Loop

In LabVIEW 2009 and earlier, if you wire a numeric constant with a value of 0 to the count (**N**) terminal of a For Loop and place a subVI inside the For Loop, LabVIEW loads the subVI into memory when the calling VI loads. In LabVIEW 2010, if you wire a constant of 0 to the count terminal of a For Loop, LabVIEW does not load the subVI into memory when the calling VI loads unless you open the block diagram of the calling VI. In addition to subVIs, this behavior applies to other block diagram objects that a VI calls, including static references, shared libraries, cluster type definitions placed on the block diagram as constants, LabVIEW classes, and refnums.

Miscellaneous Block Diagram Enhancements

LabVIEW 2010 includes the following miscellaneous block diagram enhancements:

- You can wire a value of –1 to the parallel instances (**P**) terminal on a For Loop to use all the loop instances you specify in the **For Loop Iteration Parallelism** dialog box when running the parallel loop iterations. If you leave the parallel instances terminal unwired or wire a value of 0 to the terminal, LabVIEW determines the number of available logical processors at run time and uses that number of loop instances to run the parallel loop iterations.
- You can right-click a Shared Variable node on the block diagram and select **Replace with Programmatic Access** from the shortcut menu to switch to programmatic shared variable access. Right-click a Shared Variable node and select **Reference Mode»Target Relative** or **Reference Mode»Absolute** from the shortcut menu to change an absolute Shared Variable node to target-relative or change a target-relative Shared Variable node to absolute, respectively.
- You can right-click a cluster constant and select **View Cluster As Icon** from the shortcut menu to reduce the size of cluster constants on the block diagram. [*Idea submitted by NI Discussion Forums member chris.b.*]
- The appearance of Boolean constants on the block diagram changed so that only the currently selected Boolean value is visible. [*Idea submitted by NI Discussion Forums member altenbach.*]
- The appearance of global variable and local variable nodes changed so that their borders are thinner and they contain arrows to indicate whether the variable is set to read or write. An arrow appears on the right if the variable is set to read, and an arrow appears on the left if the variable is set to write. [*Idea submitted by NI Discussion Forums member altenbach.*]
- Right-click a string constant and select **Visible Items»Display Style** from the shortcut menu to display a glyph on the object that indicates the string display type. [*Idea submitted by NI Discussion Forums member altenbach.*]
- When you select specific objects to clean up on the block diagram, the icon of the **Clean Up Diagram** toolbar button changes to indicate LabVIEW cleans up only the selected objects.

Front Panel Enhancements

LabVIEW 2010 includes the following enhancements to the front panel and related functionality.

Exporting Data from Graphs, Charts, Tables, and Arrays

You can export data from graphs, charts, tables, and arrays to Microsoft Excel or to the clipboard. **(Windows)** You also can export data from charts and graphs to DIAdem. However, you cannot export data from intensity graphs to DIAdem.

To export data, right-click a graph, chart, table, or array, select **Export**, and then select one of the available options from the shortcut menu. You can export only data that is visible in the front panel window.

Note You must have Microsoft Excel installed to use the **Export Data To Excel** option. You must have DIAdem installed to use the **Export Data To DIAdem** option. Refer to the

National Instruments Web site at ni.com/diadem to learn more about DIAdem and to download the latest version of DIAdem.

Refer to the **Fundamentals»Graphs and Charts»Concepts»Graphs and Charts»Customizing Graphs and Charts** topic on the **Contents** tab of the *LabVIEW Help* for caveats and recommendations to consider when you export data.

Miscellaneous Front Panel Enhancements

LabVIEW 2010 includes the following miscellaneous front panel enhancements:

- To switch the location of two connector pane terminals, press and hold the <Ctrl> key and use the Positioning tool to select the two terminals you want to switch. **(Mac OS X)** Press and hold the <Option> key. **(Linux)** Press and hold the <Ctrl> key. [*Idea submitted by NI Discussion Forums member tst.*]
- Right-click a string control or indicator and select **Visible Items»Display Style** from the shortcut menu to display a glyph on the object that indicates the string display type. [*Idea submitted by NI Discussion Forums member altenbach.*]
- The **Export Simplified Image** shortcut menu item, available when you right-click a graph, chart, table, picture control, digital data, or digital waveform control or indicator, is available at **Export»Export Simplified Image** in the shortcut menu.

Environment Enhancements

LabVIEW 2010 includes the following enhancements to the LabVIEW environment.

Searching Enhancements (Windows)

LabVIEW 2010 includes an application-wide search capability. The search returns results from the **Controls** and **Functions** palettes, the help system, and ni.com. The search text box appears in the upper-right corner of the **Getting Started** window and the front panel and block diagram windows of a VI in edit mode. Use the **Search** page of the **Options** dialog box to specify the categories you want LabVIEW to search.

Refer to the **Fundamentals»LabVIEW Environment»Concepts»Searching in LabVIEW** topic on the **Contents** tab of the *LabVIEW Help* for more information about performing application-wide searches.

Dialog Box Enhancements

LabVIEW 2010 includes the following dialog box enhancements.

3D Plot Properties Dialog Box Enhancements

LabVIEW 2010 includes the following changes to the **3D Plot Properties** dialog box:

- The **Plot Properties** dialog box is renamed **3D Plot Properties**.
- On the **Waterfall** page, the **Hidden Lines** option in the **Waterfall mode** pull-down menu no longer exists. Use the **Slices** option in the **Waterfall mode** pull-down menu to display the graph with chromatic slices on the X-Y plane.

Change Visible Palettes Dialog Box Enhancements

LabVIEW 2010 includes the following enhancements to the **Change Visible Palettes** dialog box:

- The **Change Visible Categories** dialog box is renamed **Change Visible Palettes**.

- This dialog box includes a **Restore Default** button that places checkmarks in the checkboxes for only the palettes that LabVIEW shows by default.

Project Properties Dialog Box Enhancements

LabVIEW 2010 includes the following enhancements to the **Project** page of the **Project Properties** dialog box:

- The **Separate Compiled Code From Source File** checkbox specifies whether to separate compiled code from VIs that you create in the project. Refer to the *Separating Compiled Code from VIs* section of this document for more information about separating compiled code from VIs.
- The **Mark Project VIs** button launches the **Mark Project VIs to Separate Compiled Code** dialog box. Use this dialog box to separate compiled code from existing VIs in a project.

Quick Drop Dialog Box Enhancements

LabVIEW 2010 includes the following enhancements to the **Quick Drop** dialog box:

- When the **Quick Drop** dialog box is active, you can access the following keyboard shortcuts:
  - **<Ctrl-P>**—Replaces the front panel or block diagram object(s) you select before you open the **Quick Drop** dialog box with the object you select in the dialog box. [*Idea submitted by NI Discussion Forums member Dany Allard.*]
  - **<Ctrl-I>**—Inserts the object you select in the **Quick Drop** dialog box on the wire(s) you select.
  - **<Ctrl-Shift-I>**—Inserts a single instance of the object you select in the **Quick Drop** dialog box on multiple wires you select.
  - **<Ctrl-B>**—Changes the VI Server class of the selected **Property Node(s)**, **Invoke Node(s)**, and/or class specifier constant(s) to the class you enter in the **Quick Drop** dialog box.
  - **<Ctrl-Shift-B>**—Changes the property or method of the selected Property Node(s) or Invoke Node(s), respectively, to the property or method name you type in the **Quick Drop** dialog box.
- The **Ctrl-Key Shortcuts** page of the **Quick Drop Shortcuts** dialog box allows you to configure which key corresponds to which keyboard shortcut. [*Idea submitted by NI Discussion Forums member Daklu.*]

VI Properties Dialog Box Enhancements

LabVIEW 2010 includes the following enhancements to the **General** page of the **VI Properties** dialog box:

- The **Source version** text box displays the version number of LabVIEW in which the VI was last saved.
- The **Separate compiled code from source file** checkbox separates compiled code from the VI. Refer to the *Separating Compiled Code from VIs* section of this document for more information about separating compiled code from VIs.

Miscellaneous Dialog Box Enhancements

LabVIEW 2010 includes the following miscellaneous dialog box enhancements:

- Use the **Clear Compiled Object Cache** dialog box to clear the object cache, a collection of the compiled code for all VIs with separate compiled code. Refer to the *Separating Compiled Code from VIs* section of this document for more information about separating compiled code from VIs.
- The **Configuration** page of the **Properties** dialog box for Shared Variable nodes includes options for configuring Shared Variable nodes.
- On the **Environment** page of the **Options** dialog box, the default value for the **Maximum undo steps per VI** text box changed to 99 steps. [*Idea submitted by NI Discussion Forums member PJM_Labview.*]

- In the **Insert Subpalette** dialog box, the **Link to a palette file in a project library (.lvlib)** option is renamed **Link to a palette file in a project library** because you can select this option to link palettes to `.lvlib` and `.lvlibp` files.
- The **Parameters** page of the **Call Library Function** dialog box includes an **Allow Resize** checkbox when you select **Adapt to Type** from the **Type** pull-down menu, then select **Interface to Data** from the **Data format** pull-down menu. Use this checkbox to specify whether the library is allowed to resize the parameter.
- In the **Icon Editor** dialog box, select **Edit»Import Glyph from File** to display a file dialog box where you can locate and import a graphic file to use as an icon or template.
- On the **Browse Options** page of the **Properties** dialog box for path controls, the **Treat LLBs as folders** checkbox is renamed **Allow selection of files in LLBs and packed project libraries** because you can select files from LLBs and packed libraries. Refer to the *LabVIEW Packed Project Libraries* section of this document for more information about using packed libraries.
- The **VI Server** page of the **Options** dialog box includes options for enabling VI Scripting and displaying additional information related to VI Scripting in the **Context Help** window. Refer to the *Programmatically Scripting VIs in LabVIEW* section of this document for more information about using VI Scripting.

Miscellaneous Environment Enhancements

LabVIEW 2010 includes the following miscellaneous environment enhancements:

- The **Help»Search the LabVIEW Help** menu item changed to **Help»LabVIEW Help**.
- The **View** button on the **Controls** and **Functions** palettes changed to the **Customize** button.
- The **Tools»Instrumentation»NI Spy** menu item launches NI Spy if it is installed. You can use NI Spy to capture instrument I/O calls and their results while LabVIEW VIs run.
- The **Window** menu includes a new section that lists open LabVIEW projects separately from other open windows, such as front panels or block diagrams. [*Idea submitted by NI Discussion Forums member gsussman.*]

Application Builder Dialog Box Enhancements

LabVIEW 2010 includes the following enhancements to the LabVIEW Application Builder and build specifications:

- The **Packed Library Properties** dialog box allows you to configure the build settings for a packed project library. Refer to the *LabVIEW Packed Project Libraries* section of this document for more information about using packed libraries.
- The **Pre/Post Build Actions** page of the **Application Properties**, **.NET Interop Assembly Properties**, **Shared Library Properties**, **Source Distribution Properties**, and **Web Service Properties** dialog boxes allows you to specify VIs you want LabVIEW to execute before or after the build.
- On the **Version Information** page of the **Application Properties** and **Shared Library Properties** dialog boxes, the **Description** text box is available only on Windows.
- On the **Source File Settings** page of the **Application Properties**, **.NET Interop Assembly Properties**, **Shared Library Properties**, and **Web Service Properties** dialog boxes, LabVIEW enables the **Remove front panel** option if you remove the checkmark from the **Use default save settings** checkbox. LabVIEW enables the **Remove block diagram** option if you remove the checkmark from the **Remove front panel** checkbox.
- On the **Advanced** page of the **Application Properties**, **.NET Interop Assembly Properties**, and **Shared Library Properties** dialog boxes, the **Copy error code files** checkbox contains a checkmark by default.

- On the **Advanced** page of the **Application Properties**, **.NET Interop Assembly Properties**, and **Shared Library Properties** dialog boxes, the **Use the default project aliases file** checkbox is renamed **Use custom aliases file**. The **Use the default LabVIEW Configuration file** checkbox is renamed **Use custom configuration file**. When you place a checkmark in the **Use custom aliases file** checkbox, you can select an aliases file in the project. When you place a checkmark in the **Use custom configuration file** checkbox, you can select a configuration file in the project.
- On the **Windows Security** page of the **Application Properties**, **.NET Interop Assembly Properties**, and **Shared Library Properties** dialog boxes, the **Certificate** pull-down menu is renamed **Personal store certificates**.
- On the **Additional Exclusions** page of the **Source Distribution Properties** dialog box, place a checkmark in the **Remove compiled code** checkbox to separate compiled code from the VIs you include in the source distribution during the build.
- **(Windows, Linux)** The **Build Status** dialog box includes an **Explore** button, which you can use to open the location of the build specification on disk. **(Mac OS X)** The corresponding button in the **Build Status** dialog box is named **Open Enclosing Folder**.
- When you build a .NET interop assembly in LabVIEW 2009, LabVIEW assigns generic names, such as `TD1`, to .NET objects that correspond to clusters and enumerated type controls assigned to VI connector panes. When you build a .NET interop assembly in LabVIEW 2010, if a LabVIEW cluster or enumerated type control is a type definition or strict type definition, LabVIEW names the corresponding .NET object according to the name of the type definition or strict type definition. For LabVIEW objects that are not type definitions or strict type definitions, LabVIEW assigns names to the corresponding .NET object that according to the label of the LabVIEW object.

Note If you place a checkmark in the **Remove compiled code** checkbox, LabVIEW dims the **Use default save settings** option on the **Source File Settings** page because you cannot remove the front panel or block diagram from a VI with separate compiled code.

The **Installer Properties** dialog box includes the following new pages and components:

- The new **Destinations** page allows you to configure the destination directory structure of the installer.
- The new **Version Information** page allows you to enter version information for the installer and view, edit, or regenerate an upgrade code.
- The new **Windows Security** page allows you to configure the digital signature for `setup.exe` when the installer runs.
- On the **Additional Installers** page, the LabVIEW Run-Time Engine tree in the **National Instruments Installers to Include** list expands so that you can remove checkmarks from components of the LabVIEW Run-Time Engine to reduce the size of the installer.
- On the **Additional Installers** page, the new **Copy all future installers to this computer when you run them** checkbox allows you to specify whether you want to copy all National Instruments installers from a distribution into a permanent location on the computer when you run the installers. Also, the **Copy selected distributions to this computer during the build** checkbox is renamed **Copy selected installers to this computer before building**.
- On the **Dialog Information** page, the **Include custom Welcome graphic** option allows you to specify a graphic that appears in the welcome dialog box of the installer. The **Include custom Banner graphic** option on this page allows you to specify a banner graphic that appears at the top of all installer dialog boxes.
- On the **Source File Settings** page, the **Unlock** checkbox allows you to remove the administrator access requirements from deployed files and folders.

The **Installer Properties** dialog box also includes the following renamed, moved, or deprecated components:

- The **Advanced** page includes the **Windows Server 2003 or later**, **Windows Server 2008 or later**, and **Windows 7 or later** options, which you can use to specify the version of Windows that users must have to run the installer. The **Windows 2000 or later** option no longer exists because you can no longer build an installer for Windows 2000. The **Windows XP or later** option is selected by default.
- The **Product version**, **Auto increment product version**, **Company name**, **Company URL**, **Company contact**, and **Company phone** options on the **Product Information** page moved to the **Version Information** page.
- On the **Registry** page, the **Rename Key** option no longer exists. Instead, you can double-click a key in the **Destination Registry** tree to rename the key. On the **Registry** page, you also can right-click a folder and select **Unlock** from the shortcut menu to remove administrator access requirements from a registry key.
- On the **Source File Settings** page, the **File Attributes** section is renamed **File and Folder Attributes**.
- On the **Source Files** page, the **Add Folder**, **Rename Folder**, **Set as default installation directory**, and **Hide unused folders?** options no longer exist. To add folders, rename folders, and set folders as default directories, use the **Destinations** page.
- On the **Source Files** page, the **Remove** option changed from a button at the bottom of the page to an arrow between the **Project Files View** and the **Destination View**.
- On the **Destinations**, **Source Files**, and **Source File Settings** pages, the labels of the `labview` and Microsoft Installer (MSI) folders in the **Destination View** changed. On the **Shortcuts** page, the names of the MSI folders in the **Directory** pull-down menu changed. The following table lists the label name changes between LabVIEW 2009 and 2010.

Table 5. Renamed Destination View Labels in LabVIEW 2010

LabVIEW 2009 or earlier Destination View labels	LabVIEW 2010 Destination View labels
`[DesktopFolder]`	`[All Users Desktop]`
No label existed.	`[LabVIEW` *x* `Examples]`
No label existed.	`[LabVIEW` *x* `Help]`
No label existed.	`[LabVIEW` *x* `Instrument Drivers]`
No label existed.	`[LabVIEW` *x* `Palettes]`
`[LV`*x*`RTEDIR]`	`[LabVIEW` *x* `Run-Time]`
No label existed.	`[LabVIEW` *x* `User Libraries]`
`[LVDIR]`	`[LabVIEW` *x*`]`
`[PersonalFolder]`	`[Personal]`
`[CommonFilesFolder]` or `[CommonFiles64File]`	`[Program Files Common]`
`[ProgramFilesFolder]` or `[ProgramFiles64Folder]`	`[Program Files]`
`[CommonAppDataFolder]`	`[Public App Data]`
`[SystemFolder]` or `[System64Folder]`	`[System]`
`[TempFolder]`	`[Temp]`

LabVIEW 2009 or earlier Destination View labels	LabVIEW 2010 Destination View labels
`[WindowsVolume]`	`[Windows Volume]`
`[WindowsFolder]`	`[Windows]`
`[ProgramMenuFolder]`	`[Program Menu]`
`[SendToFolder]`	`[Send To]`
`[StartMenuFolder]`	`[Start Menu]`
`[StartupFolder]`	`[Startup]`

LabVIEW Project Enhancements

LabVIEW 2010 includes the following enhancements to the LabVIEW project and related functionality.

LabVIEW Packed Project Libraries

LabVIEW packed project libraries are project libraries that package multiple LabVIEW files into a single file with a `.lvlibp` file extension. The top-level file of a packed library is a project library. By default, the packed library has the same name as the top-level project library. Packed libraries contain one or more VI hierarchies that compile for a specific operating system. Use packed libraries to deploy fewer files than when deploying project libraries. When you deploy VIs in a packed library, fewer files deploy because the packed library is a single file.

To build a packed library, right-click **Build Specifications** in the **Project Explorer** window and select **New»Packed Library** from the shortcut menu to display the **Packed Library Properties** dialog box. Use this dialog box to configure build settings for the packed library.

Convert a project library to a packed library in order to shorten build time when stand-alone applications call VIs in a packed library. To replace an existing project library with a packed library in a LabVIEW project, right-click the project library and select **Replace with a packed library** from the shortcut menu. LabVIEW updates all calling VIs of the project library with the qualified name of the packed library.

Refer to the **Fundamentals»Working with Project and Targets** book on the **Contents** tab of the *LabVIEW Help* for more information about using packed libraries.

Sorting Items in a Project

You can use sort options to organize files within a LabVIEW project. Using a sort option does not alter the organization of the project on disk, which allows you to share projects more easily with other developers, as well as merge and compare fewer changes when you submit a project to source control.

Note Source control is available only with the Professional Development System.

To sort items in the **Project Explorer** window, right-click a target or a folder, select **Arrange By**, and select a sort option from the shortcut menu. You can sort projects by **Name**, **Type**, **Path**, **Custom**, or **Same As Parent**. By default, all new projects sort by **Name**. All projects you create in previous versions of LabVIEW sort by **Custom**. The sort option you select applies automatically to any new items you add to the project.

Note Sort options also apply to project libraries and auto-populating folders in the **Project Explorer** window.

Retrieving Missing or Out-of-Date Files in Source Control

To retrieve missing or out-of-date files in a LabVIEW project, right-click the project root and select **Get Latest Version of All Files** from the shortcut menu. LabVIEW retrieves the latest version of any missing or out-of-date files from source control and copies the files to the local directory. If LabVIEW identifies missing files that it cannot retrieve, such as dependencies or files not on the server, the **Missing File Information** dialog box displays information about the missing files. Click the **Continue** button to retrieve the latest version of the files that are available on the server.

Miscellaneous Project Enhancements

LabVIEW 2010 includes the following miscellaneous project enhancements:

- The **File** menu includes the new menu item **Close All (this Project)**, which closes all open files in the current LabVIEW project. [*Idea submitted by NI Discussion Forums member Simon Holman.*]
- When you select a type definition as the data type of a shared variable, LabVIEW links the shared variable to the type definition. If you update the type definition, LabVIEW updates all associated shared variables. To select a type definition as the data type of a shared variable, on the **Variable** page of the **Shared Variable Properties** dialog box, select **From Custom Control** from the **Data Type** pull-down menu, then navigate to the type definition you want to use.
- Shared variables support the Vision Image data type.
- The **General Settings** page of the **Project Library Properties** dialog box includes an **Apply Icon To VIs** button that allows you to apply an existing project library icon to VIs in the library. [*Idea submitted by NI Discussion Forums member hfettig.*]
- **(Mac OS X, Linux)** LabVIEW 2010 no longer supports the use of symbolic links for files in auto-populating folders.

New and Changed VI, Function, and Node Enhancements

LabVIEW 2010 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 2010 includes the following new VIs and functions.

Advanced File VIs and Functions

The **Advanced File** palette includes the new **Packed Library** palette with the following new VIs:

- Get Exported File List
- Packed Library Path

Advanced IIR Filtering VIs

The **Advanced IIR Filtering** palette includes the following new VIs:

- Butterworth Order Estimation
- Chebyshev Order Estimation
- Elliptic Order Estimation
- Inverse Chebyshev Order Estimation

Advanced TDMS VIs and Functions (ETS, VxWorks, Windows)

The **Advanced TDMS** palette includes the following new functions that work with both Windows and real-time operating systems, such as ETS and VxWorks:

- TDMS Advanced Close
- TDMS Advanced Open

- TDMS Advanced Synchronous Read
- TDMS Advanced Synchronous Write
- TDMS Set Channel Information
- TDMS Set Next Read Position
- TDMS Set Next Write Position

The **Advanced TDMS** palette includes the following new functions that work only with Windows operating systems:

- TDMS Advanced Asynchronous Read
- TDMS Advanced Asynchronous Write
- TDMS Configure Asynchronous Reads
- TDMS Configure Asynchronous Writes
- TDMS Get Asynchronous Read Status
- TDMS Get Asynchronous Write Status
- TDMS Reserve File Size
- TDMS Start Asynchronous Reads
- TDMS Stop Asynchronous Reads

Application Control VIs and Functions

The **Application Control** palette includes the new **VI Scripting** palette with the following new VI and functions:

- New VI
- New VI Object
- New VI Object Offset From Referenced Object
- Open VI Object Reference
- Traverse for GObjects

HTTP Client VIs

The **HTTP Client** palette includes the following new VIs:

- AddHeader
- CloseHandle
- DELETE
- GET
- HEAD
- OpenHandle
- POST
- POSTMultipart
- PUT
- SetAPIKey

Network Streams Functions

The **Network Streams** palette includes the following new functions:

- Create Network Stream Reader Endpoint
- Create Network Stream Writer Endpoint
- Destroy Stream Endpoint
- Flush Stream
- Property Node (Network Stream Endpoint)

- Read Multiple Elements from Stream
- Read Single Element from Stream
- Write Multiple Elements to Stream
- Write Single Element to Stream

Shared Variable Node, VI, and Functions

The **Shared Variable** palette includes the following new constant and function:

- Local Variable Object Reference
- Search Variable Container

The **PSP Variable** palette includes the new Read Variable with Timeout function.

Storage/DataPlugin VIs (Windows)

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

- Convert to TDM or TDMS
- Data File Viewer

The **Storage/DataPlugin** palette includes the new **Manage DataPlugins** palette with the following new VIs:

- Export DataPlugin
- List DataPlugins
- Register DataPlugin
- Unregister DataPlugin

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

- Get Property Info
- Merge Storage Refnums

Miscellaneous New VIs and Functions

LabVIEW 2010 includes the following miscellaneous new VIs and functions:

- The **Dialog & User Interface** palette includes the Merge Errors function. You can resize this node to merge the number of error clusters you desire. [*Idea submitted by NI Discussion Forums member Dany Allard.*]
- The **Filters** palette includes the Mathematical Morphological Filter VI.
- The **Numeric** palette includes the DBL numeric constant.
- The **Ordinary Differential Equations** palette includes the DAE Radau 5th Order VI.
- The **Signal Operation** palette includes the AutoCorrelation Matrix VI.
- The **Waveform Measurements** palette includes the FFT Power Spectrum and PSD VI.

Changed VIs, Functions, and Nodes

The following VIs, functions, and nodes changed in LabVIEW 2010.

Shared Variable Node, VI, and Functions

You can wire a reference to an I/O variable container to the **shared variable refnum in** input of the following functions on the **Shared Variable** palette. You can use this new functionality to optimize performance by reading and writing I/O variable containers as arrays.

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

- Scanned Variable Write
- Write Variable

Signal Operation PtByPt VIs

The **Signal Operation PtByPt** palette includes the following changed VIs:

- **Threshold Peak Detector PtByPt**—Renamed Threshold Detector PtByPt. This VI also includes an **index** output that displays the beginning index of the most recent peak.
- **Unwrap Phase PtByPt**—Includes a **wrap base** input that controls the discontinuities in **unwrapped phase**.

Storage/DataPlugin VIs

The **Storage** palette is renamed the **Storage/DataPlugin** palette. This palette includes the following changed VIs:

- **Get Object Info**—The **storage format** output is renamed **DataPlugin name**.
- **Get Properties**—The **Description**, **Minimum**, and **Maximum** block diagram outputs appear only if you select the **Terminal** option next to these properties in the **Source** column of the configuration dialog box.
- **Open Data Storage**—Includes a **DataPlugin name** input, an **additional parameters** input, and a **DataPlugin name out** output. Use these inputs and output to specify a DataPlugin and use the DataPlugin to open the supported storage format. The configuration dialog box includes a **Get More DataPlugins from ni.com/dataplugins** button. In the configuration dialog box, the **Data storage format** pull-down menu is renamed **DataPlugin** and includes an **<auto-detect the storage format>** item. The **Data storage parameters** section is renamed **DataPlugin parameters**.
- **Read Data**—Includes **index** and **count** block diagram inputs. Use these inputs to specify a subset of data to read.
- **Set Property**—The data type of the **Value** input of the Set Property (Enum) instance changed from a 32-bit signed integer to a 16-bit unsigned integer.

TDM Streaming VIs and Functions

The following VIs moved from the **TDM Streaming** palette to the **Advanced TDMS** palette:

- TDMS Convert Format
- TDMS Create Scaling Information

Waveform Generation VIs

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

- **Basic Multitone**—The **phase relationship** input contains a new `linear phase` value, and the `linear` value is renamed `linear difference`. The **phase relationship** input also changed from an enum to a 16-bit unsigned integer.
- **Basic Multitone with Amplitudes**—The **phase relationship** input contains a new `linear phase` value, and the `linear` value is renamed `linear difference`. The **phase relationship** input also changed from an enum to a 16-bit unsigned integer.

Waveform Measurements VIs

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

The Pulse Measurements VI includes an **export mode** input that specifies whether this VI returns the **period** or **duty cycle** outputs.

In all instances of the following VIs, the **export signals** input is renamed **export mode**:

- Extract Multiple Tone Information
- Extract Single Tone Information

- Harmonic Distortion Analyzer
- SINAD Analyzer

All instances of the following VIs include a **window parameter** input that specifies the beta parameter for a Kaiser window, the standard deviation for a Gaussian window, and the ratio of the mainlobe to the sidelobe for a Dolph-Chebyshev window:

- Cross Spectrum (Mag-Phase)
- Cross Spectrum (Real-Im)
- FFT Power Spectrum and PSD
- FFT Spectrum (Mag-Phase)
- FFT Spectrum (Real-Im)
- Frequency Response Function (Mag-Phase)
- Frequency Response Function (Real-Im)

Miscellaneous VI, Function, and Node Changes

LabVIEW 2010 includes the following miscellaneous VI, function, and node changes:

- The **Analysis of Variance** VIs can accept and automatically convert input levels that do not begin with zero or input levels that have nonconsecutive values.
- The **path** output of the Close File function changed from optional to recommended.
- The Digital IIR Filter VI includes a **filter structure option** input that specifies the order of the IIR cascade filter.
- The **Treat LLBs as folders** checkbox in the configuration dialog box of the File Dialog Express VI is renamed **Allow selection of files in LLBs and packed project libraries** because you can select a file from LLBs and packed libraries.
- The In Place Element structure moved from the **Memory Control** palette to the **Structures** palette. [*Idea submitted by NI Discussion Forums member Jim Kring.*]
- The data type of the **lcm(x,y)** output of the Lcm VI changed from a 32-bit signed integer to a 64-bit signed integer.
- The Limit Testing Time instance of the Limit Testing VI includes an **output t0** input that determines the source of start time, t0, of the waveforms for **output values**.
- The LU Factorization VI supports real or complex matrices. In LabVIEW 2009 and earlier, the LU Factorization VI supports only square real or square complex matrices.
- The **options** input of the Open VI Reference function accepts a new value, `0x40`. This option allows calls to reentrant VIs to use shared clone VIs.
- The Ramp Pattern VI changed to a polymorphic VI and contains a new instance called Ramp Pattern by Delta.
- The STFT Spectrograms VI includes a **window parameter** input that specifies the beta parameter for a Kaiser window, the ratio of the mainlobe to the sidelobe for a Dolph-Chebyshev window, and the standard deviation for a Gaussian window. Also, the **type** element of the **window info** input accepts new types of windows to use to compute the STFT.
- The Threshold Peak Detector VI is renamed Threshold Detector.

New and Changed Classes, Properties, Methods, and Events

LabVIEW 2010 includes new VI Server classes, properties, methods, and events. Refer to the **LabVIEW 2010 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.

Math Plots Properties and Methods

The Plot Specific:Waterfall:Waterfall Mode property no longer sets or returns the `Hidden Lines` display mode. This property does set or return the `Slices` display mode.

XML Parser Properties and Methods

In the following Document methods, the **Array of Nodes** parameter is renamed **Array of Elements**:

- Get Elements By Tag Name Array
- Get Elements By Tag Name NS Array

In the following Element methods, the **Array of Nodes** parameter is renamed **Array of Elements**:

- Get Element By Tag Name Array
- Get Element By Tag Name NS Array

Separating Compiled Code from VIs

Every VI contains compiled code by default. When you edit a VI, LabVIEW recompiles the code of the VI. LabVIEW also recompiles any VIs that call the VI, causing you to save VIs that you do not edit because recompiling causes an unsaved change.

In LabVIEW 2010, you can separate compiled code from a VI. When you open a VI with separate compiled code, LabVIEW generates and saves a VI object file (`.viobj`) with the compiled code. You do not open VI object files directly, only the VI.

When you separate compiled code from a VI, recompiling the code of the VI does not create an unsaved change. As a result, if you use source control, you do not have to check out all the files in a hierarchy when you edit a VI.

Note The LabVIEW Run-Time Engine cannot compile code. If you intend to load or run a VI using the LabVIEW Run-Time Engine, do not separate compiled code from the VI.

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

Streaming Data Continuously between Applications

A network stream is a lossless, unidirectional, one-to-one communication channel that consists of a writer and a reader endpoint. Network streams can stream any data type continuously between two LabVIEW applications. Use the **Network Streams** functions to stream data with network streams. Use the Network Stream Endpoint properties to view information about endpoints.

Refer to the **Fundamentals»Networking in LabVIEW»Concepts»Streaming Data Continuously** book on the **Contents** tab of the *LabVIEW Help* for more information about network streams.

NI Instrument Driver Finder Enhancements

After you install a driver or double-click an installed driver in the **Installed Instrument Drivers** list of the NI Instrument Driver Finder, the **Start Using Instrument Driver** page displays, where you can perform the following actions:

- If the driver is project-based, click the **Open Project** button to open the LabVIEW project that contains the driver VIs.
- Click the **Open Palette** button to open the palette that contains the driver VIs.
- Double-click a VI in the **Examples** list to open the VI. The **Examples** list displays example VIs included in the driver project.
- Click the **Explore** button to open the location of the driver on disk.

You also can launch the Instrument Driver Finder and click the **Scan for Instruments** button to detect instruments connected to the computer that use NI-VISA.

Refer to the **Controlling Instruments»Using Instrument Drivers** book on the **Contents** tab of the *LabVIEW Help* for more information about using instrument drivers.

Using a Property Node to Access Private Data of a LabVIEW Class

In LabVIEW 2009 and earlier, you had to add multiple read or write accessor VIs to the block diagram to access the private data of a LabVIEW class. In LabVIEW 2010, you can wire a LabVIEW class to a Property Node to access the private data. To make the private data accessible through a Property Node, you must create an accessor VI for each data member. Right-click the LabVIEW class and select **New»Property Definition Folder** from the shortcut menu to create a property definition folder, in which you can create new accessor VIs or add existing accessor VIs. You also can use the **Create Accessor** dialog box to create an accessor VI automatically. If you use the **Create Accessor** dialog box to create an accessor VI, you must place a checkmark in the **Make available through Property Nodes** checkbox.

Programmatically Scripting VIs in LabVIEW

Use the VI and functions on the **VI Scripting** palette with the associated properties and methods to create, edit, and run VIs programmatically. You can complete simple and complex tasks ranging from showing or hiding control labels to creating entire VIs programmatically.

Note You must enable VI Scripting to use the VI Scripting VI and functions. To enable VI Scripting, select **Tools»Options** to display the **Options** dialog box, select **VI Server** from the **Category** list, and then place a checkmark in the **Show VI Scripting Functions, Properties and Methods** checkbox.

With VI Scripting, you can reduce the amount of time you spend on repetitive VI editing tasks, such as:

- Creating several similar VIs
- Aligning and distributing controls and indicators
- Showing or hiding control and indicator labels
- Wiring block diagram objects

Refer to the **Fundamentals»Programmatically Controlling VIs** book on the **Contents** tab of the *LabVIEW Help* for more information about using VI Scripting.

Activating Third-Party LabVIEW Add-ons

The **Third Party Add-on Activation** wizard allows you to activate licensed third-party add-ons. If a third-party add-on is already installed on the computer but not activated, this wizard appears automatically after you launch LabVIEW. You also can select **Help»Activate Add-ons** to display this wizard.

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

LabVIEW 2010 includes the following enhancements to Web services:

- To use Web services, deploy LabVIEW applications to the Application Web Server, which is an independent system service that runs on a remote target or local system. You can enable and run the Application Web Server and any applications deployed to the server without running LabVIEW or the LabVIEW Run-Time Engine on the host system.
- On the **Service Settings** page of the **Web Service Properties** dialog box, the **Deploy as standalone Web Service** checkbox configures a Web service to run independently of LabVIEW on the Application Web Server system service.

- The **Web Server** page of the **Options** dialog box no longer enables Web services. You must use the Application Web Server to enable Web services and encrypt Web services communication with Secure Socket Layer (SSL).
- The **Auxiliary VI** option in the **Configure RESTful VI** dialog box configures a VI that runs as part of a Web service but does not exchange data with a Web client.
- To debug a Web service application on a desktop system, select **Operate»Debug Application or Shared Library** from the **Project Explorer** window of the application.
- Use the **HTTP Client** VIs to build a Web client in LabVIEW that can exchange data with Web services.
- Using Web-based target configuration, you can establish access permissions for users and groups to configure the Application Web Server on remote targets and local host machines.

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

371780F-01 Jun10