LabVIEW™ Upgrade Notes

These upgrade notes describe the process of upgrading LabVIEW for Windows, Mac OS, and Linux to version 8.6, issues you might encounter when you upgrade, and new features. To learn about any potential compatibility issues, read these upgrade notes prior to loading any VIs you saved in a previous version of LabVIEW in this new version of LabVIEW. Consider creating backup copies of all LabVIEW files you saved in a previous version of LabVIEW before you load the files in this new version of LabVIEW.

If you are upgrading from LabVIEW 7.1 or earlier to LabVIEW 8.6, National Instruments recommends that you review the following documents in addition to these upgrade notes for more information about enhancements, changes, and features added to LabVIEW between LabVIEW 7.1 and LabVIEW 8.6.

- **LabVIEW 8.0 Upgrade Notes**—The *Upgrade and Compatibility Issues* section and the *LabVIEW 8.0 Features and Changes* section provide important information for upgrade users. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `upnote8` to access the *LabVIEW 8.0 Upgrade Notes*.
- **LabVIEW 8.2 Upgrade Notes**—The *Upgrade and Compatibility Issues* section and the *LabVIEW 8.2 Features and Changes* section provide important information for upgrade users. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `upnote82` to access the *LabVIEW 8.2 Upgrade Notes*.
- **LabVIEW 8.5 Upgrade Notes**—The *Upgrade and Compatibility Issues* section and the *LabVIEW 8.5 Features and Changes* section provide important information for upgrade users. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `upnote85` to access the *LabVIEW 8.5 Upgrade Notes*.

Refer to the *LabVIEW Help* for more information about LabVIEW 8.6 features, as well as for information about LabVIEW programming concepts, step-by-step instructions for using LabVIEW, and reference information about LabVIEW VIs, functions, palettes, menus, tools, properties, methods, events, dialog boxes, and so on. The *LabVIEW Help* also lists the LabVIEW documentation resources available from National Instruments. Access the *LabVIEW Help* by selecting **Help»Search the LabVIEW Help**.

Contents

Upgrading to LabVIEW 8.6

If you are upgrading from a previous version of LabVIEW, read this section, *Upgrading to LabVIEW 8.6*, and the *Upgrading from LabVIEW x.x* sections in the *Upgrade and Compatibility Issues* section of this document first, where *x.x* is the version of LabVIEW from which you are upgrading.

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 8.6**—This section includes instructions for uninstalling toolkits and modules, copying environment settings and `user.lib` files from a previous version of LabVIEW, and converting VIs to LabVIEW 8.6.
   - **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 `exqmex` to download tests that can evaluate VIs for some compatibility issues.

- (Optional) **LabVIEW 8.6 Features and Changes**—This section includes brief descriptions of the new features in this version of LabVIEW. Refer to the *LabVIEW Help* for more complete information about using these features. Access the *LabVIEW Help* by selecting **Help»Search the LabVIEW Help**.

2. (Optional) Uninstall 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 8.6* and the appropriate subsection for the platform to which you are installing
   - *Installing LabVIEW Add-Ons* if you are installing LabVIEW toolkits or modules from a CD instead of the LabVIEW Platform DVD
   - **(Windows)** *Activating the LabVIEW License* and all subsections
   - (Optional) *Installing and Configuring Hardware* and the appropriate subsection for the platform to 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 8.6. Refer to the *Converting VIs* section of this document for more information converting VIs saved in a previous version of LabVIEW.

Converting VIs

You cannot open a VI saved in a version of LabVIEW prior to LabVIEW 6.0 without contacting an NI representative to obtain conversion software to upgrade your code to VI formats compatible with LabVIEW 8.6. When you open a VI last saved in LabVIEW 6.0 or later, LabVIEW 8.6 automatically converts and compiles the VI. You must save the VI in LabVIEW 8.6, 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 8.6 do not load in earlier versions of LabVIEW. Select **File»Save for Previous Version** to save VIs so they can run in LabVIEW 8.5, 8.2, or 8.0. Before saving VIs in LabVIEW 8.6 after you convert them, keep a backup copy of VIs you plan to use in LabVIEW 8.5, 8.2, or 8.0.

If your computer does not have enough memory to convert all the VIs at once, convert the VIs in stages. Examine the hierarchy of VIs you want to convert and begin by loading and saving subVIs in the lower levels of the hierarchy. Then progress gradually to the higher levels of the hierarchy. Open and convert the top-level VI last. You also can select **Tools»Advanced»Mass Compile** to convert a directory of VIs. However, mass compiling converts VIs in a directory or LLB in alphabetical order. If the conversion process encounters a high-level VI first, mass compiling requires approximately the same amount of memory as if you opened the high-level VI first.

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

Upgrading 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 8.6. For those modules and toolkits that are not on the LabVIEW Platform DVDs, refer to the National Instruments Web site at `ni.com/info` and enter the info code `compat` for more information about which LabVIEW modules and toolkits are compatible with the current version of LabVIEW.

NI Modules and Toolkits

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

Operating System	Media to Use	Important Notes
Windows	DVD	Use the LabVIEW Platform DVDs to install LabVIEW 8.6 and versions of modules and toolkits that are compatible with LabVIEW 8.6. 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 8.6 without uninstalling or modifying previous versions. Refer to the *LabVIEW Release Notes* for information about installing LabVIEW, modules, and toolkits.
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 8.6 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.
Mac/Linux		

Note Some versions of toolkits do not work with LabVIEW 8.6. 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 8.6 installation, first uninstall the toolkit and then repair the LabVIEW installation using **Add or Remove Programs**.

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 8.6 support using one of the following methods.

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

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

Third Party Add-Ons

Contact the vendor of third-party LabVIEW add-ons to determine if the add-on is compatible with LabVIEW 8.6 for your operating system. Make sure you mass compile any VIs that are related to the add-on.

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

Mass Compiling LabVIEW

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

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

Upgrading Additional National Instruments Software

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

NI TestStand 3.5 and NI TestStand 4.0 return an error when you attempt to configure the following LabVIEW 8.6 Express VIs:

- Close Data Storage
- Formula
- Get Properties

- Open Data Storage
- Read Data
- Set Properties
- Spectral Measurements
- Write Data

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

Note NI TestStand 4.1 or later resolves this issue.

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

You must use NI Spy 2.3 or later in LabVIEW 8.*x*. NI Spy 2.5 is available on the National Instruments Device Drivers CD.

LabVIEW 8.6 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 order Measurement Studio 8.0 or later.

Upgrading from Previous Versions of LabVIEW

You can install LabVIEW 8.6 without uninstalling previous versions of LabVIEW. While versions of LabVIEW might share components, upgrading to new versions of LabVIEW does not affect the performance of previous versions of LabVIEW on the computer because the new versions install in a different directory. LabVIEW 5.*x* and earlier install in the labview directory. LabVIEW 6.0 and later install in the labview *x.x* directory, where *x.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 8.6 installer, and set the installation directory to the same labview directory where you installed the previous version of LabVIEW.

(Windows) You also can replace the existing version of LabVIEW with LabVIEW 8.6 by using the Add/Remove Programs applet in the Control Panel to uninstall the existing version of LabVIEW. The uninstaller does not remove any files you created in the labview directory.

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

Copying Environment Settings from a Previous Version of LabVIEW

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

Caution If you replace the LabVIEW 8.6 preferences file with a preferences file from a previous version, you might override preference settings added to LabVIEW since the previous version.

After you install LabVIEW 8.6, copy the LabVIEW preferences file to the LabVIEW 8.6 directory.

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

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

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

Copying user.lib Files from a Previous Version of LabVIEW

To use files from the `user.lib` directory of a previous version of LabVIEW, copy the files from the `labview` directory in which the previous version is installed. After you install LabVIEW 8.6, copy the files to the `user.lib` directory in the LabVIEW 8.6 directory.

Upgrade and Compatibility Issues

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

Refer to the `readme.html` file in the `labview` directory for information about known issues in the new version of LabVIEW, additional compatibility issues, and information about late addition features in LabVIEW 8.6.

Upgrading from LabVIEW 8.5

You might encounter the following compatibility issues when you upgrade to LabVIEW 8.6 from LabVIEW 8.5.

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

Platforms Supported

LabVIEW 8.6 includes the following changes in platforms supported:

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

System Requirements

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

(Mac OS) LabVIEW 8.6 requires at least 262 MB of disk space for the minimum LabVIEW installation or 828 MB 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 disk space for the complete LabVIEW installation.

Printed Documentation

The following documents did not change for LabVIEW 8.6. Therefore, the content might not reflect changes made in LabVIEW 8.6.

- *Getting Started with LabVIEW* manual
- **(LabVIEW 8.2, 8.5, and 8.6)** *LabVIEW Fundamentals Manual*

Note Because the *LabVIEW Fundamentals Manual* is a subset of the **Fundamentals** book in the *LabVIEW Help*, refer to the **Fundamentals** book on the **Contents** tab of the *LabVIEW Help* for updated content.

VI and Function Behavior Changes

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

Report Generation VIs

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

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

External Code (DLLs and CINS)

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

Miscellaneous VI and Function Behavior Changes

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

- 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 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.

Deprecated VIs and Functions

LabVIEW 8.6 does not support the following VIs and functions:

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

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

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

Property, Method, and Event Behavior Changes

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

- 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.

Deprecated Properties, Methods, and Events

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

- Control Value:Set [Flattened] method of the VI class. Use the Control Value:Set method instead.
- Control Value:Get All [Flattened] method of the VI class. Use the Control Value:Get All method instead.
- Control Value:Get [Flattened] method of the VI class. Use the Control Value:Get method instead.
- VIModificationBitSet property of the VI Properties (ActiveX) class. Use the VIModificationBitSet2 property instead.
- Modifications:VI Modifications Bitset property of the VI Properties 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, the new Modifications:VI Modifications Bitset property returns a 64-bit value.

Renamed Properties, Methods, and Events

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

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

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, the shared component is replaced by an older version. If you then try to build a shared library in the most current version of LabVIEW, you will receive an error because the shared component is missing functionality that LabVIEW 8.6 relies on. To correct this issue, reinstall LabVIEW 8.6.

Upgrading from LabVIEW 8.2

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

Note You also can refer to the National Instruments Web site at `ni.com/info` and enter the info code `ex5d8c` for more information about additional issues you may encounter upgrading from LabVIEW 8.2.

Platforms Supported

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

- LabVIEW 8.5 and later supports Windows Vista x86 and x64.
- LabVIEW 8.5 and later supports Macintosh computers with both Intel and PowerPC processors.

System Requirements

(Windows) LabVIEW 8.5 requires at least 1.2 GB of disk space for the LabVIEW installation. LabVIEW 8.6 requires at least 1.6 GB of disk space for the LabVIEW installation.

(Mac OS) LabVIEW 8.5 requires at least 502 MB of disk space for the minimum LabVIEW installation or 734 MB disk space for the complete LabVIEW installation. LabVIEW 8.6 requires at least 262 MB of disk space for the minimum LabVIEW installation or 828 MB 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 disk space for the complete LabVIEW installation. LabVIEW 8.6 requires at least 365 MB of disk space for the minimum LabVIEW installation or 651 MB disk space for the complete LabVIEW installation.

Windows Vista Compatibility Issues

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

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

- **(Windows Vista x86)** VI components install properly but show up as unsigned in the Windows Defender log. The VIs do run properly.
- **(Windows Vista x64)** These VIs return error –4850.

VI and Function Behavior Changes

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

Enhancements to Analysis VIs and Functions

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

Mathematics VIs

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

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

Numeric Functions

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

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

Signal Processing VIs

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

Hyperbolic Functions

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

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

Libraries & Executables VIs and Functions

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

Open VI Reference Function

In LabVIEW 8.0 and 8.2, if the name of the VI from the **vi path** input matches the name of a VI in memory on that target, LabVIEW returns a reference to the VI in memory and LabVIEW does not load the VI specified in the **vi path** input. In LabVIEW 8.5 and later, if the name of the VI from the **vi path** input matches the name of a VI in memory on that target but the paths differ, the Open VI Reference function returns an error.

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.

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 of the GraphScale class instead.
- Grid Colors:Y Color property of the GraphChart class. Use the Grid Colors:Major Color and Grid Colors:Minor Color of the GraphScale class instead.
- Legend:Plots Shown property of the WaveformChart class. Use the Legend:Number of Rows property instead.
- Legend:Plots Shown property of the WaveformGraph class. Use the Legend:Number of Rows property instead.
- Pixel Width property of the ListBox class. Use the Bounds:Area Width property instead.
- Scrollbars Visible property of the Picture class. Use the Horizontal Scrollbar Visible and Vertical Scrollbar Visible properties instead.
- Set Geometry method of the SceneObject class. Use the Set Drawable method instead.
- Scene:Geometry:New Mesh method of the Application class. Use the Scene:Drawable:Geometry:New Mesh method instead.

- Drag Starting event of the Control class. Use the Drag Starting event of the appropriate control class instead.
- Drag Starting? event of the Control class. Use the Drag Starting? event of the appropriate control class instead.

Renamed Properties, Methods, and Events

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

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

Class	LabVIEW 8.2 Name	LabVIEW 8.5 Name	Type
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 place a class control or indicator on the block diagram, you must update the class icon to occupy a smaller space so that the class mask does not obscure any part of the class icon. Use an image no larger than 32 pixels wide by 19 pixels high.

Opening LLBs in LabVIEW

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

Timed Loop Priority Level Restriction

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

Waveform Data Type

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

Upgrading from LabVIEW 8.0

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

Refer to the *LabVIEW Upgrade Notes* for each version of LabVIEW between versions 8.0 and 8.5 at `ni.com/manuals` for more information about the new features and changes in each version.

Platforms Supported

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

- LabVIEW 8.2 and later does not support Windows XP x64.
- LabVIEW 8.2 and later does not support Mac OS X 10.3.8 or earlier.
- LabVIEW 8.2 provides some support for Macintosh computers with Intel processors. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `macintel` for more information about Macintosh support. LabVIEW 8.5 provides support for Macintosh computers with both Intel and PowerPC processors.

System Requirements

(Windows) LabVIEW 8.2 requires at least 1.2 GB for the LabVIEW installation. LabVIEW 8.6 requires at least 1.6 GB for the LabVIEW installation.

(Mac OS) LabVIEW 8.2 requires at least 500 MB of disk space for the minimum LabVIEW installation or 700 MB disk space for the complete LabVIEW installation. LabVIEW 8.6 requires at least 262 MB of disk space for the minimum LabVIEW installation or 828 MB disk space for the complete LabVIEW installation.

(Linux) LabVIEW 8.2 requires at least 430 MB of disk space for the minimum LabVIEW installation or 620 MB disk space for the complete LabVIEW installation. LabVIEW 8.6 requires at least 365 MB of disk space for the minimum LabVIEW installation or 651 MB disk space for the complete LabVIEW installation.

Printed Documentation

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

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

VI and Function Behavior Changes

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

Communicating between Application Instances

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

Back Transform Eigenvectors VI

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

DataSocket Write Function

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

File I/O VIs

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

GPIB Status Function

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

Histogram VI

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

Open VI Reference Function

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

Polynomial Roots VI

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

Ramp Pattern VI

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

Read Registry Value Simple VI

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

Resample Waveforms (single-shot) VI

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

Sound VIs

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

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

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

Waveform VIs

LabVIEW 8.2 and later includes changes to the following Waveform VIs:

- **Basic Level Trigger Detection VI**—In both instances of this VI, the **slope** input changed to **trigger slope**.
- **Get Waveform Subset VI**—Includes the following instances: WDT Get Waveform Subset DBL, WDT Get Waveform Subset CDB, WDT Get Waveform Subset EXT, WDT Get Waveform Subset I16, WDT Get Waveform Subset I32, WDT Get Waveform Subset I8, and WDT Get Waveform Subset SGL. The start/duration format input no longer includes an **Absolute Time** option. The start input changed to start samples/time, and the actual start output changed to actual start samples/time.
- **Get Waveform Time Array VI**—The X array output changed from a double-precision, floating-point numeric data type to a time stamp data type.
- **Get Y Value VI**—This VI and the corresponding polymorphic instances were renamed to Get XY Value. The Get XY Value VI now includes an X value output, and the **data value** output changed to Y value.
- **Number of Waveform Samples VI**—This VI is a polymorphic VI with the following instances: WDT Number of Waveform Samples DBL, WDT Number of Waveform Samples CDB, WDT Number of Waveform Samples EXT, WDT Number of Waveform Samples I16, WDT Number of Waveform Samples I32, WDT Number of Waveform Samples I8, and WDT Number of Waveform Samples SGL.
- **Read Waveform from File VI**—Returns an error status of TRUE in the **error out** output when the error is end-of-file.
- **Replace Subset VI**—The start input changed to start samples/time, and the actual start value output changed to actual start samples/time.
- **Search for Digital Pattern VI**—The start input changed to start index/time.
- **Search Waveform VI**—The time of best fit and time of fits outputs changed from a double-precision, floating-point numeric data type to a time stamp data type.
- **Waveform Min Max VI**—The min time and max time outputs changed from a double-precision, floating-point numeric data type to a time stamp data type.
- **Waveform to XY Pairs VI**—The **x** element of the XY pairs output changed from a double-precision, floating-point numeric data type to a time stamp data type.

Property, Method, and Event Behavior Changes

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

- The default behavior for the **options** input of the ActiveX GetVIReference method is to prompt users to find missing subVIs of the referenced VI. A new value, `0x20`, specifies not to display the **Find** dialog box or prompt users to find missing subVIs of the referenced VI.
- The Add Item method of the ProjectItem class return an error when you try to add a shared variable to a library that is not opened in a project.
- If the **Auto Dispose Ref** input of the Run VI method is TRUE and the method returns an error, LabVIEW does not dispose of the reference.
- Valid values for the Application:Language property include `zh-cn` to indicate that Simplified Chinese is the language of the LabVIEW environment.

- In LabVIEW 8.0, .NET methods that pass array data types by reference pass all data as the refnum data type. In LabVIEW 8.2 and later, .NET methods that pass array data types by reference pass the data as the actual data type.
- The Edit Position property of the DigitalTable, MulticolumnListbox, Table, and TreeControl classes returns values of (–2, –2) to indicate that the user is not making edits to the text of the control. The Edit Row property of the ListBox class returns a value of –2 to indicate that the user is not making edits to the text of the control.
- In LabVIEW 8.0, the Defer Panel Updates property did not defer the update of front panels in a subpanel. In LabVIEW 8.2 and later, the Defer Panel Updates property works with subpanels.
- The Application Instance Close and Application Instance Close? events replace the Application Exit and Application Exit? events. When you use the Application Instance Close event in a VI running outside a LabVIEW project, LabVIEW generates the event when you quit LabVIEW through the user interface or programmatically. LabVIEW generates the Application Instance Close? event when you quit LabVIEW through the user interface. When you register the Application Instance Close and Application Instance Close? events for a VI running within a LabVIEW project, LabVIEW generates the events when the application instance closes or when you quit LabVIEW.

Deprecated Properties, Methods, and Events

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

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

Renamed Properties, Methods, and Events

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

Class	LabVIEW 8.0 Name	LabVIEW 8.2 and Later Name	Type
Application	Disconnect From Slave	LVRT:Disconnect From Slave	Method
Application	Application Exit	Application Instance Close	Event
Application	Application Exit?	Application Instance Close?	Event
IntensityGraph, MixedSignalGraph, and WaveformGraph	Cursor Palette Visible	Cursor Legend Visible	Property
Library	Delete Library Tag	Library Tag:Delete	Method
Library	Get Icon	Icon:Get	Method
Library	Get Library Tag	Library Tag:Get	Method
Library	Get Library Tag Names	Library Tag:Get Names	Method
Library	Get Lock State	Lock State:Get	Method
Library	Get Source Scope	Source Scope:Get	Method
Library	Save	Save:Library	Method
Library	Save a Copy	Save:Copy	Method

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

Application Builder Changes

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

Upgrading from LabVIEW 7.*x*

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

Refer to the *LabVIEW Upgrade Notes* for each version of LabVIEW between versions 7.*x* and 8.5 at `ni.com/manuals` for more information about the new features and changes in each version.

Note The *LabVIEW Fundamentals Manual* did not change for LabVIEW 8.6. You can access the PDF versions of these documents in the `labview\manuals` directory. Because the *LabVIEW Fundamentals Manual* is a subset of the **Fundamentals** book in the *LabVIEW Help*, refer to the **Fundamentals** book on the **Contents** tab of the *LabVIEW Help* for updated content.

Platforms Supported

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

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

System Requirements

LabVIEW 7.*x* requires a minimum of 128 MB of RAM, but National Instruments recommends 256 MB of RAM. LabVIEW 8.5 requires a minimum of 256 MB of RAM, but National Instruments recommends 1 GB of RAM.

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

Windows

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

LabVIEW 7.*x* requires at least 130 MB of disk space for the minimum LabVIEW installation or 550 MB disk space for the complete LabVIEW installation. LabVIEW 8.6 requires 1.6 GB disk space for the complete LabVIEW installation.

Mac OS

LabVIEW 7.*x* requires at least 280 MB of disk space for the minimum LabVIEW installation or 350 MB disk space for the complete LabVIEW installation. LabVIEW 8.6 requires at least 262 MB of disk space for the minimum LabVIEW installation or 828 MB disk space for the complete LabVIEW installation.

Linux

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

LabVIEW 7.*x* requires at least 200 MB of disk space for the minimum LabVIEW installation or 300 MB disk space for the complete LabVIEW installation. LabVIEW 8.6 requires at least 365 MB of disk space for the minimum LabVIEW installation or 651 MB disk space for the complete LabVIEW installation.

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

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

Custom Palette Views

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

VI and Function Behavior Changes

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

.NET VIs and Applications

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

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

Analyze VI Algorithms

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

Append Signals Express VI

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

Comparison Functions

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

Dot Product VI

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

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

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

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

where y_i^* is the complex conjugate of y_i.

Easy Text Report VI (Mac OS and Linux)

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

Format Into String Function

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

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

Join Numbers Function

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

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

Mathematics VIs and Matrices

In LabVIEW 8.*x*, Mathematics VIs support the matrix data type. If you load a VI from LabVIEW 7.*x* in LabVIEW 8.*x* and the VI contains a Mathematics VI wired to a function that can use the matrix data type and is instead using a 2D array, a red 7.*x* glyph appears on the function. The red glyph indicates that LabVIEW replaced the 2D array with the matrix data type.

Number to String Conversion Functions

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

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

Open VI Reference Function

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

In LabVIEW 8.*x*, if the vi path input of the Open VI Reference function is a string, LabVIEW opens the VI only if vi path matches the qualified VI name of a VI in memory on that target. If vi path is a path, LabVIEW searches for a VI in memory with the same path on the same target. If LabVIEW does not find a VI with a matching path, LabVIEW tries to load the VI from disk at the specified path. In LabVIEW 8.5 and later, an error occurs if LabVIEW cannot find the file or if the VI name of the file is the same as the qualified VI name as another VI in memory on that target.

Quick Scale VI

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

Read Key VI

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

Scale VI

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

Semaphore VIs

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

SMTP Email VIs

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

Sort Complex Numbers VI

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

Unit Vector VI

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

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

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

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

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

User VIs

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

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

Deprecated VIs and Functions

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

- LabVIEW 7.1 and later do not install the Polynomial Real Zero Counter VI. Use the Polynomial Real Zeros Counter VI instead.
- **(Mac OS)** LabVIEW 7.1 and later do not install the PPC VIs. Use the TCP VIs instead.
- LabVIEW 8.*x* does not support the QR Factorization VI. Use the QR Decomposition VI instead.
- LabVIEW 8.*x* does not support the Levenberg Marquardt or the Nonlinear Lev-Mar Fit VIs. Use the Nonlinear Curve Fit VI instead.
- In LabVIEW 8.*x*, the VISA Status Description function is not on the **Functions** palette. Use the Simple Error Handler or General Error Handler VIs instead.
- LabVIEW 8.*x* does not support the Chi Square Distribution, F Distribution, Normal Distribution, and T Distribution VIs. Use the Chi-Squared, F, Normal, and Student t instances, respectively, of the Continuous CDF VI instead.
- LabVIEW 8.*x* does not support the Inv Chi Square Distribution, Inv F Distribution, Inv Normal Distribution, and Inv T Distribution VIs. Use the Chi-Squared, F, Normal, and Student t instances, respectively, of the Continuous Inverse CDF VI instead.
- In LabVIEW 8.*x*, the 1D Linear Evaluation VI and the 2D Linear Evaluation VI are not on the **Functions** palette. Use the Linear Evaluation VI instead.
- In LabVIEW 8.*x*, the 1D Polynomial Evaluation VI and the 2D Polynomial Evaluation VI are not on the **Functions** palette. Use the Polynomial Evaluation VI instead.
- In LabVIEW 8.*x*, the 1D Rectangular to Polar VI and the 1D Polar to Rectangular VI are not on the **Functions** palette. Use the Re/Im To Polar function and the Polar To Re/Im function instead.
- In LabVIEW 8.*x*, the Harmonic Analyzer VI is not on the **Functions** palette. Use the Harmonic Distortion Analyzer VI instead to measure the **THD** or **component levels** outputs, or use the SINAD Analyzer VI to measure the **SINAD** or **THD Plus Noise** outputs.
- In LabVIEW 8.*x*, the Network Functions (avg) VI is not on the **Functions** palette. Use the Frequency Response Function (Mag-Phase), Frequency Response Function (Real-Im), Cross Spectrum (Mag-Phase), or Cross Spectrum (Real-Im) VIs instead.
- In LabVIEW 8.*x*, the Pulse Parameters VI is not on the **Functions** palette. Use the Transition Measurements VI instead to measure the **slew rate**, **duration**, **overshoot** (the Transition Measurements VI equivalent is the **post-transition** output), or **preshoot** (the Transition Measurements VI equivalent is the **pre-transition** output) outputs, the Pulse Measurements VI to measure the **period**, **pulse duration**, or **duty cycle** outputs, or the Amplitude and Levels VI to measure the **amplitude**, **high state level**, or **low state level** outputs.
- In LabVIEW 8.*x*, the Transfer Function VI is not on the **Functions** palette. Use the Frequency Response Function (Mag-Phase) or Frequency Response Function (Real-Im) VIs instead.
- In LabVIEW 8.*x*, the NI DIAdem Report Wizard Express VI is not on the **Functions** palette. Use the DIAdem Report Express VI instead.
- In LabVIEW 8.*x*, the VISA resource name constant and the IVI logical name constant are not on the **Functions** palette. To specify a VISA resource name, use the VISA resource name input of the VISA VIs. To specify an IVI logical name, use the appropriate input of the appropriate driver VI that initializes the instrument.
- In LabVIEW 8.*x*, the error ring constant is not on the **Functions** palette. Use a 32-bit signed integer constant instead to enter the error code that you want.

- **(Windows and Linux)** In LabVIEW 8.*x*, the Sound VIs available on the **Sound** palette in LabVIEW 7.*x* are not on the **Functions** palette. Use the Sound VIs in LabVIEW 8.*x* instead. The examples shipped with LabVIEW 7.*x* do not ship with LabVIEW 8.*x*.

File I/O VIs and Functions

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Property, Method, and Event Behavior Changes

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

Application Properties and Methods

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

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

Front Panel:Open Method

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

Run VI Method

In LabVIEW 7.1, if you set the **Auto Dispose Ref** input of the Run VI method to TRUE, LabVIEW automatically disposes the reference after the VI stops running. If the Run VI Method generates an error, the reference is not automatically closed. In LabVIEW 8.0, LabVIEW automatically disposes the reference if the method returns an error. This behavior might break a VI at run time if part of the block diagram depends on the reference. In LabVIEW 8.2 and later, the behavior has been changed back to the 7.1 behavior.

Key Down and Key Repeat Events

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

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

ListBox Properties

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

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

Owning VI Property

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

Text Property

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

TreeControl Properties

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

VI Strings Methods

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

Deprecated Properties, Methods, and Events

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

Cursor Properties

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

ListBox, MulitcolumnListbox, Table, DigitalTable, and TreeControl Properties and Events

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

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

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

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

NamedNumeric Properties

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

Panel Properties

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

Subpanel Properties

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

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

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

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

VI Properties, Methods, and Events

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

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

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

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

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

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

Changes from Previous Versions of the Application Builder

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

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

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

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

Application Item Tags

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

- APP_SAVE_WITH_OPTIONS
- APP_UPDATE_VXI
- APP_DSC_TOOLBAR
- APP_DSC_TAGEDITOR
- APP_DSC_TAGMONITOR
- APP_DSC_HTV
- APP_DSC_ENGINE
- APP_DSC_SECURITY
- APP_DSC_LOGOUT
- APP_DSC_CPWD
- APP_DSC_USERINFO
- APP_DSC_USEREDITOR
- APP_DSC_ADVANCED
- APP_DSC_STARTUP
- APP_DSC_SRVBRW
- APP_DSC_IST
- APP_DSC_IMAGENAV
- APP_DSC_OPTIONS
- APP_SRC_CODE_CTRL
- APP_BUILD_STANDALONE_APP
- APP_EDIT_VI_LIBRARY
- APP_DN_ASSEMBLY_REFS
- APP_SHOW_CLIPBOARD
- APP_VIEW_PRINTED_MANUALS
- APP_RT_ENGINE_INFO
- APP_SWITCH_EXEC_TARGET
- APP_REALTIME

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

HiQ Support

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

Error List Window

In LabVIEW 7.*x* and earlier, the **VI List** section of the **Error list** window shows errors for all VIs in memory. In LabVIEW 8.*x*, the **Items with errors** section of the **Error list** window shows errors for all items in memory, such as VIs and libraries. If two or more items have the same name, this section shows

the specific application instance for each ambiguous item. Refer to the *LabVIEW Help* for more information about application instances.

VI String File Syntax

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

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

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

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

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

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

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

Converting Type Descriptor Data to and from LabVIEW 7.*x*

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

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

In LabVIEW 8.*x*, when you load a VI last saved in LabVIEW 7.*x* or earlier, LabVIEW 8.*x* automatically sets the **Convert 7.x Data** attribute on the Flatten To String function. The function continues to operate as in LabVIEW 7.*x* and earlier. If you want a VI to use the LabVIEW 8.*x* type descriptor format, right-click the Flatten To String function and select **Convert 7.x Data** from the shortcut menu to remove the checkmark. Use the LabVIEW 8.*x* type descriptor format if VIs do not need to manipulate files that contain data written in LabVIEW 7.*x* or earlier and do not send or receive data to or from VIs running in LabVIEW 7.*x* or earlier. Support for the previous type descriptor format might be discontinued in future versions of LabVIEW.

Migrating from the LabVIEW Built-In Source Control Provider

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

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

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

1. In the previous version of LabVIEW, make sure that the files included in the LabVIEW built-in source control provider are checked in by all users.
2. On the computer where you want to add the files to the new source control provider, use the built-in provider to get the latest versions of all the files.
3. Use the built-in provider to check out the files from source control.
4. In the third-party source control provider, configure the settings you want for the new source control project.
5. Configure LabVIEW to work with the third-party source control provider.

   Refer to the **Fundamentals»Organizing and Managing a Project»How-To»Using Source Control in LabVIEW** book on the **Contents** tab of the *LabVIEW Help* for information about configuring LabVIEW to work with a third-party source control provider.
6. Create a LabVIEW project. Add the files included in the built-in provider to the project. When LabVIEW prompts you, add the files to source control. You also can add the files directly in the third-party provider.

   Refer to the **Fundamentals»Organizing and Managing a Project»How-To»Creating a LabVIEW Project** book on the **Contents** tab of the *LabVIEW Help* for information about creating a LabVIEW project.

Converting NaN Strings to Integer Types (Windows)

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

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

Constants Wired to Case Structures

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

Delaying Operating System Messages

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

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

In LabVIEW 7.*x*, LabVIEW processes operating system messages while running DLL or shared library functions. In LabVIEW 8.*x*, LabVIEW delays the processing of operating system messages until the end of calls to DLL functions or until you load a modal dialog box from the DLL. This delay allows DLL functions to execute without interruption and prevents LabVIEW from calling the same DLL while a DLL function is running, which can result in a deadlock state. Delaying the operating system messages, such as keyboard messages from the users, is useful in order to avoid calling the same shared library file while a shared library function runs. For example, if the shared library function is called in response to the user pressing a button, the user should not be able to press the button again until the shared library function has completed.

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

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

Resource Manager (Mac OS)

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

One- and Two-Button Dialog Boxes

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

Property and Invoke Nodes

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

Updating Shared Libraries

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

Margin Values for Printing

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

Upgrading from LabVIEW 6.*x*

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

Refer to the *LabVIEW Upgrade Notes* for each version of LabVIEW between versions 6.*x* and 8.6 at `ni.com/manuals` for more information about the new features and changes in each version.

Changes to the Waveform Data Type

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

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

Serial Compatibility VIs

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

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

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

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

Default Data in Loops

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

Remote Front Panel License

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

Development System includes a remote front panel license that allows five clients to view and control a front panel remotely.

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

Multiple Thread Allocation

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

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

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

Instrument Drivers

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

Units and Conversion Factors

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

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

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

Unit	6.1 Definition	7.*x* and Later Definition
slug	32.174 lb	14.59390 kg
unified atomic mass (u)	1.66057e–27 kg	1.66053873e–27 kg

Defer Panel Updates Property

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

Data Ranges for Numeric Controls

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

Coercion Dots and Type Definitions

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

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

File Dialog Box Button Label

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

Control Online Help Function

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

Displaying the Front Panel When Loaded

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

Open VI Reference Function

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

In LabVIEW 8.0 and later, if you use the Open VI Reference function to create a reference to a template, LabVIEW instantiates a VI from the template even if that template is already in memory.

Exponential Representation

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

IVI Configuration Store File

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

Upgrading from LabVIEW 5.*x* or Earlier Versions

Refer to the National Instruments Web site at `ni.com/info` and enter the info code `ext8h9` for information about upgrading to LabVIEW 8.6 from LabVIEW 5.*x* or earlier. This site provides information about older versions of *LabVIEW Upgrade Notes* which you should read as well as more information about converting your VIs so they work with LabVIEW 8.6.

LabVIEW 8.6 Features and Changes

Refer to the *LabVIEW Help* for more information about LabVIEW 8.6 features, including programming concepts, step-by-step instructions, and reference information. Access the *LabVIEW Help* by selecting **Help»Search the LabVIEW Help**.

Refer to the `readme.html` in the `labview` directory for known issues, a partial list of bugs fixed, additional compatibility issues, and information about late addition features in LabVIEW 8.6.

Installing LabVIEW

(Windows) With LabVIEW 8.6 you can install LabVIEW and select 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 8.6* section of the *LabVIEW Release Notes* for more information.

LabVIEW Documentation

The *LabVIEW Quick Reference Card* is reorganized to include keyboard shortcuts, online resources, and a documentation usage guide.

New Example VIs

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

Block Diagram Enhancements

LabVIEW 8.6 includes the following enhancements to the block diagram and related functionality.

Creating and Editing Express VIs

Select **Tools»Advanced»Create or Edit Express VI** to create or edit an Express VI using the **Create or Edit Express VI** dialog box. You can create an Express VI from an existing VI, from another Express VI, or from a blank VI. In previous versions of LabVIEW, this functionality is available only if you have the LabVIEW Express VI Development Toolkit installed.

Refer to the **Fundamentals»Building the Block Diagram»Concepts»Express VIs** book on the **Contents** tab in the *LabVIEW Help* for more information about creating Express VIs in LabVIEW.

Managing Breakpoints Using the Breakpoint Manager Window

You can right-click an object on the block diagram and select **Breakpoint»Set Breakpoint** from the shortcut menu to create a breakpoint. To disable a breakpoint and prevent the VI from pausing at the breakpoint without physically removing the breakpoint, right-click an object with a breakpoint and select **Breakpoint»Disable Breakpoint** from the shortcut menu. You can enable a disabled breakpoint by right-clicking an object with a disabled breakpoint and selecting **Breakpoint»Enable Breakpoint** from the shortcut menu. To remove a breakpoint, right-click an object with a breakpoint and select **Breakpoint»Clear Breakpoint** from the shortcut menu.

Use the **Breakpoint Manager** window to manage breakpoints in a VI. The **Breakpoint Manager** window allows you to disable, enable, and clear breakpoints in the VI hierarchy individually or all at once. Double-click a breakpoint in the **Breakpoint Manager** window to find the location of a specific breakpoint on the block diagram.

Select **View»Breakpoint Manager** to display the **Breakpoint Manager** window, or right-click an object and select **Breakpoint»Breakpoint Manager** from the shortcut menu. Select individual breakpoints or click the **Select All** button to select all breakpoints. Click the **Enable** or **Disable** buttons to enable or disable all selected breakpoints simultaneously.

Allowing Preallocation of Shared-Clone Reentrant VIs in Timed Structures

You can configure LabVIEW to either preallocate or share clones of a shared-clone reentrant VI within a Timed Loop or Timed Sequence structure. For example, if you place a shared-clone reentrant VI inside of a Timed Loop or Timed Sequence structure, you can set the structure to preallocate clones for instances of the reentrant VI you call within the structure. Instances of the reentrant VI that you call outside the structure continue to share clones. By default, LabVIEW preallocates clone VIs for timed structures. To conserve memory usage among shared-clone reentrant VIs, configure the timed structure to allocate clone VIs as needed. To set the allocation of a VI within a Timed Loop or a Timed Sequence structure, right-click the structure, select **Shared Clone Allocation** from the shortcut menu, and select one of the following options:

- **Automatic**—LabVIEW decides whether to set the structure as **Preallocate** or **Allocate As Needed** based on the context of the structure. LabVIEW will append the shortcut menu item with text that indicates the setting LabVIEW chooses. For example, if LabVIEW chooses **Preallocate**, the menu item becomes **Automatic (Preallocate)**.
- **Preallocate**—LabVIEW creates a clone VI for each call to the reentrant VI within the structure. The Timed Loop or Timed Sequence structure displays a particular icon to indicate that LabVIEW is preallocating clones for any reentrant VI in the structure.
- **Allocate As Needed**—LabVIEW does not create a clone VI until you make a call to the reentrant VI within the structure. The Timed Loop or Timed Sequence structure displays a particular icon to indicate that LabVIEW is sharing clones for any reentrant VI in the structure.

Wiring Tunnels Automatically in Structures with Multiple Cases

LabVIEW can wire input and output tunnels in multiple cases of a structure automatically. Configure an input and output tunnel for automatic wiring by right-clicking the existing output tunnel and selecting one of the following options from the **Linked Input Tunnel** shortcut menu:

- **Create & Wire Unwired Cases**—LabVIEW automatically wires the tunnels in any existing cases that you have not wired previously. LabVIEW also wires the tunnels in any new cases you create.
- **Create**—LabVIEW automatically wires the tunnels in any new cases you create. Existing cases remain unwired.

If you select one of the previous options, LabVIEW associates, or creates a link between, the output and input tunnels. A white triangle glyph appears on both tunnels indicating LabVIEW linked the tunnels. Right-click a linked output tunnel and select **Linked Input Tunnel»Find** or **Linked Input Tunnel» Clear** from the shortcut menu to locate linked input tunnels or to remove the link between the tunnels, respectively.

Note You cannot configure tunnels for automatic wiring on the Stacked Sequence structure.

Miscellaneous Block Diagram Enhancements

LabVIEW 8.6 includes the following miscellaneous block diagram enhancements:

- In LabVIEW 8.6, all comparison functions work with LabVIEW classes.
- LabVIEW 8.6 generates prototype information for Boolean controls and indicators on the connector pane of a VI that you build into a shared library. If you call the shared library in LabVIEW, LabVIEW handles the Boolean as an unsigned 8-bit integer. LabVIEW returns an error if the label of the control is `Boolean`. `Boolean` is not case-sensitive.

Front Panel Enhancements

LabVIEW 8.6 includes the following enhancements to the front panel and related functionality.

3D Graph Enhancements

(Windows) The 3D Surface Graph, 3D Parametric Graph, and 3D Curve Graph that shipped with LabVIEW 8.5 and earlier are renamed the ActiveX 3D Surface Graph, ActiveX 3D Parametric Graph, and ActiveX 3D Curve Graph, respectively. The ActiveX 3D graphs use ActiveX technology and VIs that handle 3D representation.

LabVIEW 8.6 includes an XControl version of the 3D graph that is available on all platforms. Use the 3D Surface, 3D Parametric, and 3D Line graphs to display 3D data on a 3D plot. Customize the appearance of a 3D graph XControl using the **3D Graph Properties** dialog box. You can configure the general graph appearance; format the x, y, and z plots; format the graph axes; add value pairs; and add cursors.

Miscellaneous Front Panel Enhancements

A private data control cannot contain an XControl or a strictly typed XControl refnum.

Environment Enhancements

LabVIEW 8.6 introduces the following enhancements to the LabVIEW environment.

Error List Window Enhancements

LabVIEW 8.6 includes enhancements to LabVIEW class error reporting that might result in fewer unbroken class items listed in the **Error List** window.

Performance Enhancements

VI file size has decreased.

VI Hierarchy Window Enhancements

LabVIEW 8.6 includes the following enhancements to the **VI Hierarchy** window:

- The **VI Hierarchy** window displays LabVIEW classes and dynamic member VIs, XControls, shared libraries, `.m` files you reference from MathScript Nodes, Express VIs, and static VI references.

- To highlight the call chain of the VI, in the **VI Hierarchy** window right-click the VI icon of a paused VI and select **Show Call Chain** from the shortcut menu. The call chain is the chain of callers from the current VI to the top-level VI. When you highlight a hierarchy connection, the connection appears as a thick, red line.

Refer to the **Fundamentals»Creating VIs and SubVIs»How-To»Creating SubVIs»VI Hierarchy** book on the **Contents** tab in the *LabVIEW Help* for more information about the **VI Hierarchy** window.

Dialog Box Enhancements

LabVIEW 8.6 includes the following dialog box enhancements.

Warnings Dialog Box Enhancements

The **Warnings** dialog box is now the **Load Warning Summary** and **Save for Previous Warning Summary** dialog boxes. Click the **Show Details** button on either summary dialog box to open the **Load and Save Warning List** dialog box and view details for each warning category. You also can select **Load and Save Warning List** from the **View** menu to open the **Load and Save Warning List** dialog box.

The **Load Warning Summary** dialog box reports a list of warning categories for the top-level item you attempt to load.

The **Save for Previous Warning Summary** dialog box reports a list of warning categories for the top-level item you attempt to save.

Application Builder Dialog Box Changes

LabVIEW 8.6 includes the following changes to the LabVIEW Application Builder:

- On the **Additional Installers** page of the **Installer Properties** dialog box, the **Installer source location** text box only displays the source location. You can no longer change the installer source location in this dialog box. If LabVIEW cannot find the installer source automatically, LabVIEW prompts you to locate the source manually later in the build process.
- When you build an installer that includes additional installers or components, you might be prompted with the **Distributions Needed** dialog box if LabVIEW cannot locate the installer you selected or any of the dependencies of the installer.
- **(Windows)** The Web service build specification allows you to build and deploy VIs in a LabVIEW project as Web services. Right-click **Build Specifications** in the **Project Explorer** window and select **New»Web Service (RESTful)** from the shortcut menu to display the **Web Service Properties** dialog box. LabVIEW Web services are available only in the LabVIEW Full Development System and the LabVIEW Professional Development System.

Shared Variable Properties Dialog Box Enhancements

LabVIEW 8.6 includes the following enhancements to the **Shared Variable Properties** dialog box:

- On the **Variable** page of the **Shared Variable Properties** dialog box, **Bind to Source** was renamed to **Enable Aliasing**. When you place a checkmark in the **Enable Aliasing** checkbox, you can specify the access type of the shared variable.
- The **Use Buffering** and **Single Writer** options on the **Variable** page were moved to the **Network** page of the **Shared Variable Properties** dialog box.
- The **Description** page of the **Shared Variable Properties** dialog box appears without the DSC module installed.

Options Dialog Box Enhancements

LabVIEW 8.6 includes the following enhancements to the **Web Server: Configuration** page of the **Options** dialog box:

- You can configure more than one interface for the LabVIEW Web Server. The **Server Label** option allows you to customize the name of a server interface and the **Server Labels (Ports)** listbox displays all interfaces currently configured for the Web Server.
- The **IP Address of Listener** specifies the IP address of the selected Web Server interface. The pull-down menu contains all addresses available on the local computer. The field does not automatically populate when using a LabVIEW Real-Time Module target.
- The **Allow Access** options configure available features for the selected Web Server interface. In the **Allow Access** section, you can activate remote front panels, scripting, Web services, static content, and snapshots.

LabVIEW 8.6 also includes the new **Web Services: Security** page of the **Options** dialog box, which allows you to configure security for LabVIEW Web services.

Miscellaneous Dialog Box Enhancements

LabVIEW 8.6 includes the following miscellaneous dialog box enhancements:

- The **Conflict Resolution** dialog box includes options to resolve conflicts LabVIEW detects during deployment. For example, this dialog appears if you deploy a VI to a target that is turned off. If you can resolve the conflict, a pull-down menu appears in the **Conflict Resolution** column. Select an option from the list to resolve the conflict. This dialog box also includes the **Previous** and **Next** buttons to highlight the previous or next item with a conflict in the **Conflicts** list.
- The **Data Type** page of the **Numeric Properties** dialog box and the **Numeric Constant Properties** dialog box and the **Output Configuration** page of the **Numeric Node Properties** dialog box include an **Include overflow status** checkbox that sets whether LabVIEW includes an overflow status with the fixed-point number.
- The **Find Missing Items** dialog box finds all items in a LabVIEW project that reference an item on disk that LabVIEW cannot find. Right-click the project root in the **Project Explorer** window and select **Find Missing Items** from the shortcut menu to display this dialog box.

Palette Enhancements

LabVIEW 8.6 includes the following palette enhancements:

Editing the Controls and Functions Palette Set Programmatically

You can use the Palette Editing VIs to create and edit palette sets programmatically. Use the Palette Editing VIs if you want to edit a large number of palettes, create new palettes, or validate the appearance of a palette after you edit it.

Refer to the `labview\examples\Palette API` directory for examples of editing the **Controls** and **Functions** palette set programmatically.

Refer to the **VI and Function Reference»Programming VIs and Functions»Application Control VIs and Functions»Palette Editing VIs** book on the **Contents** tab in the *LabVIEW Help* for more information about editing palettes programmatically.

Miscellaneous Palette Enhancements

LabVIEW 8.6 includes the following miscellaneous palette enhancements:

- The VIs and functions that appeared on the **XML** palette are now available on the **LabVIEW Schema** palette. The **LabVIEW Schema** palette is a subpalette on the **XML** palette.
- When you upgrade, LabVIEW attempts to restore the palette format from the previously installed version of LabVIEW.

Miscellaneous Environment Enhancements

LabVIEW 8.6 includes the following miscellaneous environment enhancements:

- You can silently uninstall a LabVIEW-built installer and the files it installs by entering the following command in the command line window:

  `uninst.exe /qb /x` *`product name`* where *`product name`* is the name of the product you want to uninstall.

  If the product name you enter does not exactly match the name of the product, the uninstall fails without a prompt that notifies you of the failure. The `uninst.exe` file is in the `C:\Program Files\National Instruments\Shared\NIUninstaller` directory. If the `C:\Program Files\National Instruments\Shared\NIUninstaller` directory is not the working directory, enter the complete path on disk to the `uninst.exe` file.
- Use the `fileVersionInfo.llb` located in the `labview\vi.lib\Platform` directory to retrieve the version information of a stand-alone application or shared library programmatically. The VI is not available on the **Functions** palette.
- The **Breakpoints** option in the **Operate** menu no longer exists. Select **View»Breakpoint Manager** instead to manage breakpoints in a VI hierarchy.
- If you have a project library within a folder, the source control operations you perform now extend to items within the project library.
- `_goopsup.llb` was moved to the `labview\vi.lib\Utility\` directory.

New and Changed VI, Function, and Node Enhancements

LabVIEW 8.6 includes the following new and changed VIs and functions. Refer to the **VI and Function Reference** book on the **Contents** tab of the *LabVIEW Help* for more information about VIs, functions, and nodes.

New VIs and Functions

LabVIEW 8.6 includes the following new VIs and functions.

Advanced Notifier Waiting VIs and Functions

The **Advanced Notifier Waiting** palette includes the following new functions:

- Wait on Notification from Multiple with Notifier History
- Wait on Notification with Notifier History

Application Control VIs and Functions

The **Application Control** palette includes the new **Palette Editing** palette with the following new VIs:

- Read Palette
- Refresh Palettes
- Write Palette

Comparison Functions

The **Comparison** palette includes the following new function:

- Fixed-Point Overflow?

Connectivity VIs and Functions (Windows)

The **Connectivity** palette includes the new **Web Services** palette with the following new VIs:

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

- Check If Session Exists
- Create Session
- Delete Session Variable
- Destroy Session
- Flush Output
- Get Session ID Cookie
- Read All Form Data
- Read All Request Variables
- Read All Session Variables
- Read Form Data
- Read Postdata
- Read Request Variable
- Read Session Variable
- Render ESP Template
- Set ESP Variable
- Set HTTP Header
- Set HTTP Redirect
- Set HTTP Response Code
- Set HTTP Response MIME Type
- Write Response
- Write Session Variable

The **Web Services** palette includes the new **Utilities** palette with the following new VIs:

- Escape HTTP URL
- Unescape HTTP URL

Fixed-Point Functions

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

- Clear Fixed-Point Overflow Status
- Include Fixed-Point Overflow Status
- Remove Fixed-Point Overflow Status

Helpers VIs

The **Helpers** palette includes the Sensor Mapping Express VI in the LabVIEW Full and Professional Development Systems. Use the Sensor Mapping Express VI to map real-world data onto a 3D model.

Queue Operations Function

The **Queue Operations** palette includes the following new function:

- Lossy Enqueue Element

Semaphore VIs

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

- Obtain Semaphore Reference
- Release Semaphore Reference

XML VIs and Functions

The **XML** palette includes the new **XML Parser** palette with the following new VIs and functions:

- Close
- Get First Non-Text Child
- Get Next Non-Text Sibling
- Get Node Text Content
- Invoke Node (XML)
- Load
- New
- Property Node (XML)
- Save

VISA Advanced VIs and Functions

The **VISA Advanced** palette includes the following new node:

- VISA Property Node

Changed VIs, Functions, and Nodes

The following VIs, functions, and nodes changed in LabVIEW 8.6.

Advanced File VIs and Functions

The **Advanced File** palette includes the following changed function.

The Get Volume Info function includes the **sector size (bytes)** output that indicates the size of the smallest physical storage unit on disk. If you open a file without buffering, you must make the amount of data in that file a multiple of the given sector size. You can disable buffering if you set the **disable buffering** input of the Open/Create/Replace File function to TRUE.

Comparison Functions

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

- **In Range and Coerce**—Supports the fixed-point data type.
- **Max & Min**—Supports the fixed-point data type.

File I/O VIs and Functions

The **File I/O** palette includes the following changed function.

The Open/Create/Replace File function includes the **disable buffering** input that allows you to specify if LabVIEW opens, creates, or replaces the file without buffering.

Disable buffering to speed up data transfers in certain situations. For example, if you stream large amounts of data or store data in a Redundant Array of Independent Disks (RAID), disable buffering to force LabVIEW to write data directly to disk.

LabVIEW Schema

The **LabVIEW Schema** palette includes the following changed functions:

- **Flatten To XML**—Supports LabVIEW classes and the fixed-point data type.
- **Unflatten From XML**—Supports LabVIEW classes and the fixed-point data type.

Numeric Functions

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

- **Divide**—Supports the fixed-point data type.
- **Reciprocal**—Supports the fixed-point data type.
- **Square Root**—Supports the fixed-point data type.

Report Generation VIs

The **Report Generation** palette includes VIs that have inputs and outputs that are specific to the LabVIEW Report Generation Toolkit. The VIs ignore these inputs and outputs unless you install the Report Generation Toolkit. The **Report Generation** palette also includes the following additionally changed VIs:

- **Append Horizontal Line to Report**—The default value of the **width** input is changed from pixels to percentage.
- **Append Report Text**—Includes a **format string** input that specifies the number formatting to use when LabVIEW converts the numbers to characters.
- **Append VI Block Diagram to Report**—All instances include an **alignment** input that sets the alignment of the image in a report.
- **Append VI Hierarchy to Report**—All instances include an **alignment** input that sets the alignment of the image in a report.
- **Append VI Icon to Report**—All instances include an **alignment** input that sets the alignment of the image in a report.
- **Append VI List of Controls to Report**—All instances include an **alignment** input that sets the alignment of the image in a report.
- **Append VI List of SubVIs to Report**—All instances include an **alignment** input that sets the alignment of the image in a report.
- **New Report Line**—Includes a **number of lines** input to specify how many lines the VI adds to the report.
- **Save Report to File**—Includes a **Prompt to Replace?** input that determines whether the VI displays a dialog box if you specify an existing filename.

Spectral Analysis VIs (Not in Base Package)

The **Spectral Analysis** palette includes the following changed VI.

The STFT Spectrograms VI includes a **time-freq config** input that specifies the configuration of the **frequency bins** and determines the number of columns in **STFT Spectrogram {X}**. This VI also includes an **energy conservation?** input that specifies whether to scale **STFT Spectrogram {X}** so that the energy in the joint time-frequency domain equals the energy in the time domain.

Call Library Function Node

The **Call Library Function** dialog box includes the following changes:

- The **Data type** pull-down menu of the **Parameters** tab includes the **Signed Pointer-sized Integer** and **Unsigned Pointer-sized Integer** numeric data types. If you use these two data types, the Call Library Function Node adapts to the specific operating system it is being executed on and returns data of the appropriate size to the library function. On 64-bit platforms LabVIEW translates these numeric data types to 64-bit integer types. On 32-bit platforms LabVIEW translates the numeric data types to 32-bit integer types.
- The **Call Library Function Node** supports the use of the `*` wildcard to make the **Library name or path** reference platform independent between 32- and 64-bit Windows. A single instance of the `*` to the left of the extension will translate to `32` on a 32-bit Windows platform and `64` on a 64-bit Windows platform. For example, `myLibrary*.dll` translates to either `myLibrary32.dll` or `myLibrary64.dll`. A double instance of `*` addresses situations where the library name did not previously reference the platform but the new library name does. For example, `myLibrary**.dll` translates to `myLibrary.dll` on a 32-bit platform and `myLibrary_64.dll` on a 64-bit platform.

Shared Variable Node

The Shared Variable node includes the following changes:

- Includes the option to enable the **ms timeout** control and **timed out?** indicator for network-published Shared Variable nodes configured to read data.
- Shared variables exhibit performance enhancements in LabVIEW 8.6 when working with a large amount of shared variables on a block diagram.
- The time-triggered variable, available on supported targets when you install the LabVIEW Real-Time Module, appears separately from standard shared variables. Right-click a supported target and select **New»Time-Triggered Variable** from the shortcut menu to create a time-triggered variable.

Miscellaneous VI, Function, and Node Changes

LabVIEW 8.6 includes the following miscellaneous VI change.

The **method** input of the Sort Complex Numbers VI contains a renamed value. **Magnitude** is renamed **Magnitude, Real, Imaginary**. The **method** input also contains a new value **Magnitude, Phase Angle**. **Magnitude, Real, Imaginary** sorts the elements with respect to their magnitude first. For the elements with the same magnitude, the VI sorts them by their real parts and then by their imaginary parts. **Magnitude, Phase Angle** sorts the elements with respect to their magnitude first. For the elements with same magnitude, the VI sorts them by their phase angle. The phase angle is in the ranges of –pi to pi.

New Classes, Properties, Methods, and Events

LabVIEW 8.6 includes new VI Server classes, properties, methods, and events. Refer to the **LabVIEW 8.6 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.

XML Parser Properties and Methods

Refer to the **Property and Method Reference»XML Parser** book on the **Contents** tab of the *LabVIEW Help* for a list of new properties and methods you can use with the XML Parser VIs and functions.

3DPC_SurfacePlot Properties

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

Cleaning Up the Block Diagram Automatically

Select **Edit»Clean Up Diagram** to automatically reroute all existing wires and rearrange objects on the block diagram to generate a cleaner layout. To configure the clean up options, select **Tools»Options** to display the **Options** dialog box and select **Block Diagram: Cleanup** from the **Category** list. You can configure LabVIEW to automatically move controls to the left side of the containing diagram and indicators to the right side of the containing diagram, place a specified number of pixels between block diagram objects and wires, and compact the block diagram layout.

Placing Objects Using Quick Drop

Use the **Quick Drop** dialog box to search for a block diagram or front panel object by name and place it on the block diagram or front panel without navigating through the **Controls** or **Functions** palettes. **(Windows and Linux)** Press the <Ctrl-Space> keys or select **View»Quick Drop** to display the **Quick Drop** dialog box. On Chinese keyboards, press the <Ctrl-Shift-Space> keys. **(Mac OS)** Press the <Command-Shift-Space> keys to display the dialog box.

Refer to the **Fundamentals»LabVIEW Environment»How-To»Finding Objects** book on the **Contents** tab in the *LabVIEW Help* for more information about using the **Quick Drop** dialog box.

Editing the Properties of Multiple Objects

You can select multiple objects on the front panel or the block diagram and edit any properties the objects share. To select multiple objects, use the Positioning tool to drag a selection rectangle around all of the objects you want to edit or hold down the <Shift> key while clicking each object. Right-click an object from the selection and select **Properties** from the shortcut menu to display the **Properties** dialog box. The **Properties** dialog box only displays tabs and properties that the objects you select share. Select similar objects to display more tabs and properties. If you select objects that do not share any common properties, the **Properties** dialog box does not display any tabs or properties.

LabVIEW Web Services (Windows, Not in Base Package)

LabVIEW 8.6 allows you to publish a VI as a Web service. Web services provide a standardized method for servers to deploy applications that any HTTP client can access. The LabVIEW Web services functionality supports clients across most major platforms and programming languages and allows you to easily implement and deploy Web applications over a network using LabVIEW.

You can activate Web services functionality through the **Web Server: Configuration** page in the **Options** dialog box. To build and deploy a Web service, right-click **Build Specifications** in the **Project Explorer** window and select **New»Web Service (RESTful)** from the shortcut menu to display the **Web Service Properties** dialog box. LabVIEW Web services are available only in the LabVIEW Full Development System and the LabVIEW Professional Development System.

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

NI Distributed System Manager

(Windows) LabVIEW 8.6 includes the NI Distributed System Manager, which replaces the Variable Manager as the unified tool to manage shared variables. Use the System Manager to create and monitor variables, processes, I/O servers, and Web services. Also, use the System Manager to interact with the Shared Variable Engine and to manage security and aliases.

Refer to the **Fundamentals»Networking in LabVIEW»Concepts»Sharing Live Data Using Shared Variables** topic on the **Contents** tab in the *LabVIEW Help* for more information about the Distributed System Manager.

LabVIEW MathScript Enhancements (Not in Base Package)

LabVIEW 8.6 introduces the following enhancements and changes to MathScript.

Note Select **Tools»MathScript Window** to display the **LabVIEW MathScript Window**.

New MathScript Functions

LabVIEW 8.6 includes the following new MathScript functions. You can use these functions in the **LabVIEW MathScript Window** or the MathScript Node.

MathScript Class	Function
advanced	`sphbesselh`, `sphbesselj`, `sphbessely`
approximation	`csaps`, `lsqcurvefit`
audio	`auread`
commands	`keyboard`
filter design	`gaussfir`, `yulewalk`
filter implementation	`cconv`
integration	`quad`
ode	`odeset`, `radau5`
plots	`imagesc`, `xlim`, `ylim`, `zlim`
spectral analysis	`buffer`, `cpsd`, `mscohere`, `pburg`, `pcov`, `peig`, `periodogram`, `pmcov`, `pmusic`, `pwelch`, `pyulear`, `rooteig`, `rootmusic`, `specgram`, `spectrogram`, `tfestimate`
pde	`pdeelliptic`
support	`warning`, `xlsread`, `xlswrite`
waveform generation	`stepfun`
windows	`taylorwin`

LabVIEW MathScript on Mac OS and Linux

The **LabVIEW MathScript Window** and the MathScript Node are available on Mac OS and Linux. However, some functions are not available on these operating systems. The following table displays the MathScript functions that are not available on Mac OS or Linux. The **x** indicates that the function is not available on that operating system.

Note The `getfileproperty` function, the `load` function, the `save` function, and the `setfileproperty` function are available on Mac OS and Linux. However, you cannot use these functions with binary measurement files (`.tdm` or `.tdms`).

MathScript Function	Mac OS	Linux
`aich`	x	
`aiwf`	x	
`aoch`	x	
`aowf`	x	
`calllib`	x	x
`dioread`	x	
`diowrite`	x	
`dos`	x	x
`ginput`	x	x
`gtext`	x	x
`libfunctionsview`	x	x
`libisloaded`	x	x
`loadlibrary`	x	x
`pause`	x	x
`system`	x	x
`unloadlibrary`	x	x
`waitforbuttonpress`	x	x

Debugging Enhancements for MathScript Nodes

LabVIEW 8.6 includes the following enhancements for debugging MathScript Nodes:

- You can use execution highlighting, single-stepping, and breakpoints to debug a script in a MathScript Node.
- The gray region on the left side of the MathScript Node displays the following:
  - Red error glyphs next to lines of script that contain an error
  - Warning glyphs
  - Breakpoints
- When you select a MathScript Node error in the **Error list** window and click the **Show Error** button, LabVIEW highlights the line of script that contains the error.

Script Highlighting in MathScript Nodes

Script highlighting uses colors to distinguish between different parts of a script in a MathScript Node. These colors improve the readability of the script and help you debug a script that contains errors or returns unexpected data. For example, script highlighting allows you to see when a user-defined function or a variable overrides a built-in MathScript function. Script highlighting is enabled by default, except for MathScript Nodes that were last saved in a previous version of LabVIEW and that use custom colors.

To enable or disable script highlighting for a particular MathScript Node, right-click inside the MathScript Node and select **Enable Script Highlighting** or **Disable Script Highlighting** from the shortcut menu. To enable or disable script highlighting for all MathScript Nodes and to customize the script highlighting colors for MathScript Nodes that have script highlighting enabled, use the **MathScript: Script Highlighting** page of the **Options** dialog box.

Miscellaneous MathScript Enhancements and Changes

LabVIEW 8.6 include the following miscellaneous MathScript enhancements and changes:

- The **VI Hierarchy** window displays `.m` files you reference from MathScript Nodes.
- The `delete` function includes an **obj** input that allows you to delete a plot object and remove the object from the **Variable List** in the **LabVIEW MathScript Window**.
- The `filter` function includes a **'direct'** input that directs LabVIEW to use a direct convolution instead of a Fourier transform.
- LabVIEW MathScripts compile faster in both the **LabVIEW MathScript Window** and the MathScript Node due to performance improvements.
- If a VI contains a MathScript Node, LabVIEW deletes all undo information when you save or run the VI.

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

Managing Overflow for Fixed-Point Numbers

Overflow conditions can occur when you perform an arithmetic operation on fixed-point numbers or when you use the To Fixed-Point function to convert numeric data to fixed-point data. To determine whether overflow occurs, you can configure a fixed-point number to include an overflow status. When you include an overflow status with a fixed-point number, LabVIEW allocates additional storage space to track whether the fixed-point number is the result of an operation that overflowed.

After you configure a fixed-point number to include an overflow status, you can display an overflow status LED on fixed-point controls, constants, and indicators. This LED lights up when the overflow status of the fixed-point number is TRUE. You also can use the Fixed-Point Overflow? function to determine the overflow status of a fixed-point number. Use the Fixed-Point functions to manipulate the overflow status of a fixed-point number.

Refer to the **Fundamentals»Building the Block Diagram»Concepts»Numeric Data** topic on the **Contents** tab in the *LabVIEW Help* for more information about fixed-point numbers.

Merging LLBs (Not in Base Package)

In the LabVIEW Professional Development System, use the **Select LLBs to Merge** dialog box to merge and resolve the differences between an LLB and two revisions of that original LLB.

Select **Tools»Merge»Merge LLBs** to display the **Select LLBs to Merge** dialog box. Specify the original LLB in the **Base LLB** field. Specify the two revised LLBs in the **Their LLB** and **Your LLB** fields. Click the **Merge** button to merge the selected LLBs and open the **Merge LLBs** dialog box. Resolve any differences between the revisions of the LLB and click **Close** and **Save** to save the merged LLB.

Refer to the **Fundamentals»Development Guidelines»How-To»Merging VIs and LLBs»Merging LLBs** topic on the **Contents** tab in the *LabVIEW Help* for more information about merging LLBs.

371780D-01 Jun08