

## COMPREHENSIVE SERVICES

We offer competitive repair and calibration services, as well as easily accessible documentation and free downloadable resources.

## SELL YOUR SURPLUS

We buy new, used, decommissioned, and surplus parts from every NI series. We work out the best solution to suit your individual needs.

 Sell For Cash  Get Credit  Receive a Trade-In Deal

## OBSOLETE NI HARDWARE IN STOCK & READY TO SHIP

We stock **New**, **New Surplus**, **Refurbished**, and **Reconditioned** NI Hardware.



*Bridging the gap between the manufacturer and your legacy test system.*

 1-800-915-6216

 [www.apexwaves.com](http://www.apexwaves.com)

 [sales@apexwaves.com](mailto:sales@apexwaves.com)

*All trademarks, brands, and brand names are the property of their respective owners.*

**Request a Quote**

 **CLICK HERE**

**cRIO-9082**

# LabVIEW™ Upgrade Notes

These upgrade notes describe the process of upgrading LabVIEW for Windows, OS X, and Linux to LabVIEW 2013. Before you upgrade, read this document for information about the following topics:

- The recommended process for upgrading LabVIEW
- Potential compatibility issues you should know about prior to loading any VIs you saved in a previous version of LabVIEW
- New features and behavior changes in LabVIEW 2013

## Contents

---

Upgrading to LabVIEW 2013.....	1
1. Back Up Your VIs and Machine Configuration.....	2
2. Test and Record the Existing Behavior of Your VIs.....	3
3. Install LabVIEW, Add-Ons, and Device Drivers.....	4
4. Convert Your VIs and Address Behavior Changes.....	4
Troubleshooting Common Upgrade Issues.....	5
Upgrade and Compatibility Issues.....	6
Upgrading from LabVIEW 8.6 or Earlier.....	6
Upgrading from LabVIEW 2009.....	6
Upgrading from LabVIEW 2010.....	8
Upgrading from LabVIEW 2011.....	9
Upgrading from LabVIEW 2012.....	10
LabVIEW 2013 Features and Changes.....	13
LabVIEW Web Services Enhancements.....	13
Productivity Enhancements to Event-Driven Programming in LabVIEW.....	16
Block Diagram Enhancements.....	17
Front Panel Enhancements.....	18
Application Builder Enhancements.....	19
New and Changed VIs, Functions, and Nodes.....	21
New and Changed Classes, Properties, Methods, and Events.....	23
Updates to LabVIEW Examples.....	23
Improved Installation Experience for the VI Package Manager (Windows).....	23
Features and Changes in Previous Versions of LabVIEW.....	23

## Upgrading to LabVIEW 2013

---

Although you can upgrade small applications to a new version of LabVIEW by installing the new version and then loading your VIs, National Instruments recommends a more rigorous upgrade process to ensure that you can detect and correct upgrade difficulties as efficiently as possible.



**Tip** This process is especially beneficial for large LabVIEW applications that control or monitor critical operations; cannot afford extended down time; use multiple modules, toolkits, or drivers; or are saved in an unsupported version of LabVIEW. Refer to the National Instruments website

at [ni.com/info](http://ni.com/info) and enter the Info Code `lifecycle` for information about which versions of LabVIEW still receive mainstream support.

## Overview of the Recommended Upgrade Process

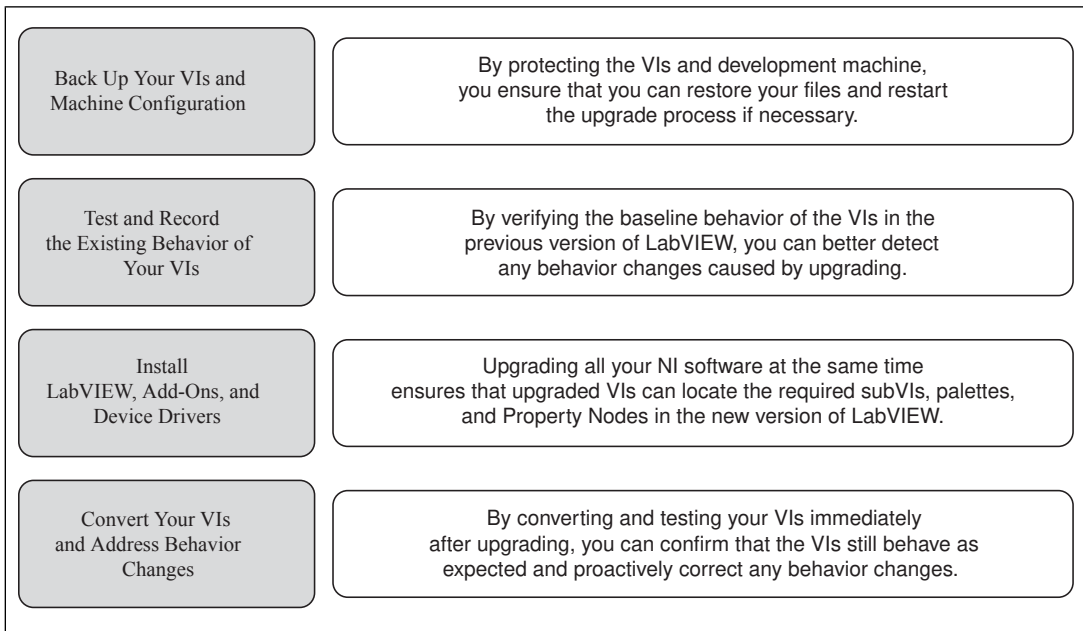


Figure 1.



**Note** To upgrade from LabVIEW 5.1 or earlier, you must first upgrade to an intermediate version of LabVIEW. Refer to the National Instruments website at [ni.com/info](http://ni.com/info) and enter the Info Code `upgradeOld` for more information about upgrading from your specific legacy version of LabVIEW.

### 1. Back Up Your VIs and Machine Configuration

By protecting a copy of your VIs and, if possible, the configuration of your development or production machine before upgrading to LabVIEW 2013, you ensure that you can restore your VIs to their previous functionality and restart the upgrade process if necessary.

#### a. Back Up Your VIs

If you back up your VIs before you upgrade LabVIEW, you can quickly revert to the back-up copy. Without the back-up copy, you can no longer open upgraded VIs in the previous version of LabVIEW without saving each VI for the previous version.

You can back up a set of VIs using either of the following methods:

- **Submit VIs to source code control**—This action allows you to revert to this version of the VIs if you cannot address behavior changes caused by upgrading the VIs. For more information about using source code control with LabVIEW, refer to the **Fundamentals»Working with Projects and Targets»Concepts»Using Source Control in LabVIEW** topic on the **Contents** tab of the *LabVIEW Help*.

- **Create a copy of the VIs**—Create a copy of the VIs according to how they are organized:
  - Saved as a project—Open the project and select **File»Save As** to duplicate the `.lvproj` file and all project contents. Ensure that you also maintain a copy of the files on which the project depends by selecting **Include all dependencies**.
  - Saved as an LLB or as VIs in a directory—From the file explorer of your operating system, create a copy of the LLB or directory and store it at a different location from the original. To prevent possible naming conflicts, avoid storing the copy on the same hard drive.

## b. Back Up Your Machine Configuration

Installing a new version of LabVIEW updates shared files in ways that sometimes affect the behavior of VIs even in previous versions. However, after you update those shared files, it is very difficult to restore the previous versions of the files. Therefore, consider one of the following methods for backing up the configuration of NI software on your development machine, especially if you are upgrading from an unsupported version of LabVIEW or if down time for your applications would be costly:

- **Create a back-up image of the machine configuration**—Use *disk imaging software* to preserve the disk state of the machine before you upgrade, including installed software, user settings, and files. To return the machine to its original configuration after you upgrade, deploy the back-up disk image.
- **Test the upgrade process on a test machine**—Although upgrading on a test machine requires more time than creating a back-up image, National Instruments strongly recommends this approach if you need to prevent or minimize down time for machines that control or monitor production. After resolving any issues that result from upgrading on the test machine, you can either replace the production machine with the test machine or replicate the upgrade process on the production machine.



**Tip** To minimize the possibility that upgraded VIs on the test machine behave differently than on the development machine, use a test machine that matches the features of the development machine as closely as possible, including CPU, RAM, operating system, and versions of software.

## 2. Test and Record the Existing Behavior of Your VIs

When you upgrade VIs, improvements between the previous version of LabVIEW and LabVIEW 2013 can occasionally change the behavior of the VIs. If you test the VIs in both versions, you can compare the results to detect behavior changes specifically caused by upgrading. Therefore, verify that you have current results for any of the following tests that you have available:

- **Mass compile logs**—Mass compiling your VIs in the previous version of LabVIEW produces a thorough log of broken VIs. This information is particularly useful if multiple people contribute to the development of the VIs or if you suspect that some of the VIs have not been compiled recently. To generate this mass compile log, place a checkmark in the **Log Results** checkbox of the **Mass Compile** dialog box. For more information about mass compiling VIs, refer to the **Fundamentals»Creating VIs and SubVIs»How-To»Saving VIs»Mass Compiling VIs** topic on the **Contents** tab of the *LabVIEW Help*.
- **Unit tests** that verify whether individual VIs perform their intended functions correctly
- **Integration tests** that verify whether a project or group of subVIs work together as expected
- **Deployment tests** that verify whether VIs behave as expected when deployed to a target, such as a desktop or FPGA target
- **Performance tests** that benchmark CPU usage, memory usage, and code execution speed. You can use the **Profile Performance and Memory** window to obtain estimates of the average execution speeds of your VIs.

- Stress tests that verify whether the VIs handle unexpected data correctly

For more information about testing VIs, refer to the **Fundamentals»Application Development and Design Guidelines»Concepts»Developing Large Applications»Phases of the Development Models»Testing Applications** topic on the **Contents** tab of the *LabVIEW Help*.



**Note** If you changed any VIs as the result of testing, back up the new versions of the VIs before proceeding.

### 3. Install LabVIEW, Add-Ons, and Device Drivers

#### a. Install LabVIEW, Including Modules, Toolkits, and Drivers

When you upgrade to a new version of LabVIEW, you must install not only the new development system but also modules, toolkits, and drivers that are compatible with the new version. For instructions about installing this software in the appropriate order, refer to the *LabVIEW Installation Guide*.

#### b. Copy user.lib Files

To ensure that custom controls and VIs you created in the previous version of LabVIEW are available to VIs in LabVIEW 2013, copy files from the `labview\user.lib` directory in the previous version to the `labview\user.lib` directory in LabVIEW 2013.

### 4. Convert Your VIs and Address Behavior Changes

Mass compiling your VIs in LabVIEW 2013 converts the VIs to the new version of LabVIEW and creates an error log to help you identify VIs that are broken. You can use this information in conjunction with the *Upgrade and Compatibility Issues* section of this document to identify and correct behavior changes associated with the new version of LabVIEW.

#### a. Mass Compile Your VIs in the New Version of LabVIEW

Mass compiling VIs simultaneously converts and saves the VIs in LabVIEW 2013. However, after mass compiling the VIs, you no longer can open the VIs in a previous version of LabVIEW without selecting **File»Save for Previous Version** for each VI or project. Therefore, mass compile only the VIs that you want to convert to the new version of LabVIEW. To help identify any problems that arose from upgrading, create a mass compile log by placing a checkmark in the **Log Results** checkbox of the **Mass Compile** dialog box.

For more information about mass compiling VIs, refer to the following topics on the **Contents** tab of the *LabVIEW Help*:

- **Fundamentals»Creating VIs and SubVIs»How-To»Saving VIs»Mass Compiling VIs**
- **Fundamentals»Creating VIs and SubVIs»How-To»Saving VIs»Common Mass Compile Status Messages**



#### b. Fix Any Broken VIs

Improvements between your previous version of LabVIEW and LabVIEW 2013 can occasionally cause some VIs to break if they use outdated features. To quickly identify and fix broken VIs in LabVIEW 2013, complete the following steps:

1. To identify VIs that broke during upgrading, compare the mass compile error log you created in the previous step to the log you created when testing the existing behavior of the VIs.
2. To determine whether updates to LabVIEW caused each VI to break, refer to the *Upgrade and Compatibility Issues* section of this document.

### c. Identify and Correct Behavior Changes

Although National Instruments invests significant effort to avoid changing the behavior of VIs between different versions of LabVIEW, improvements and bug fixes occasionally do alter the behavior of VIs. To quickly identify whether the new version of LabVIEW changes the behavior of your VIs, use one or more of the following tools:

- **Upgrade VI Analyzer Tests**—For large sets of VIs, these tests provide an efficient way to identify many behavior changes caused by upgrading. Complete the following steps to obtain and use these tests:
  1. Download the Upgrade VI Analyzer Tests for all versions of LabVIEW later than your previous version. Refer to the National Instruments website at [ni.com/info](http://ni.com/info) and enter the Info Code `analyzevi` to download these tests.
  2. Open and run the tests by selecting **Tools»VI Analyzer»Analyze VIs** and starting a new VI Analyzer task. To analyze an entire project at once, select this menu option from the **Project Explorer** window rather than from a single VI.
  3. Resolve test failures by referring to the *Upgrade and Compatibility Issues* section for the version of LabVIEW that corresponds to the tests. For example, if the LabVIEW 2010 Upgrade VI Analyzer tests locate a potential behavior change, refer to the *Upgrading from LabVIEW 2009* section of that topic.
- **Upgrade documentation**
  - *Upgrade and Compatibility Issues* section of this document—Lists changes that may break or affect the behavior of your VIs. Refer to the subsections for each version of LabVIEW beginning with your previous version.
    -  **Tip** To quickly locate deprecated objects and other objects mentioned in the *Upgrade and Compatibility Issues* section, open your upgraded VIs and select **Edit»Find and Replace**.
  - LabVIEW 2013 Known Issues list—Lists bugs discovered before and throughout the release of LabVIEW 2013. Refer to the National Instruments website at [ni.com/info](http://ni.com/info) and enter the Info Code `lv2013ki` to access this list. Refer to the *Upgrade - Behavior Change* and *Upgrade - Migration* sections to identify workarounds for any bugs that may affect the behavior of upgraded VIs.
  - Module and toolkit documentation—For some modules and toolkits, such as LabVIEW FPGA and the LabVIEW Real-Time Module, lists upgrade issues specific to that add-on
  - Driver readme files—Lists upgrade issues specific to each driver. To locate each readme, refer to the installation media for the driver.
    -  **Tip** To determine whether a behavior change resulted from a driver update rather than an update to LabVIEW, test your VIs in the previous version of LabVIEW after installing LabVIEW 2013.
- **Your own tests**—Perform the same tests on the VIs in LabVIEW 2013 that you performed in the previous version and compare the results. If you identify new behaviors, refer to the upgrade documentation to diagnose the source of the change.

### Troubleshooting Common Upgrade Issues

Refer to the **Upgrading to LabVIEW 2013»Troubleshooting Common Upgrade Issues** topic on the **Contents** tab of the *LabVIEW Help* for more information about solving the following upgrade issues:

- Locating missing module or toolkit functionality

- Locating missing subVIs, palettes, and Property Nodes
- Determining why LabVIEW 2013 cannot open VIs from a previous version of LabVIEW
- Determining which versions of NI software are installed
- Restoring VIs to a previous version of LabVIEW

## Upgrade and Compatibility Issues

---

Refer to the following sections for changes specific to different versions of LabVIEW that may break or alter the behavior of your VIs.

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 2013.

### Upgrading from LabVIEW 8.6 or Earlier

Refer to the National Instruments website at [ni.com/info](http://ni.com/info) and enter the Info Code `oldUpgradeIssues` to access upgrade and compatibility issues you might encounter when you upgrade to LabVIEW 2013 from LabVIEW 8.6 or earlier. Also, refer to the other *Upgrading from LabVIEW x* sections in this document for information about other upgrade issues you might encounter.

### Upgrading from LabVIEW 2009

You might encounter the following compatibility issues when you upgrade to LabVIEW 2013 from LabVIEW 2009. Refer to the *Upgrading from LabVIEW 2010*, *Upgrading from LabVIEW 2011*, and *Upgrading from LabVIEW 2012* sections of this document for information about other upgrade issues you might encounter.

#### VI and Function Behavior Changes

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

- Elliptic Coefficients
- Elliptic Filter
- Elliptic Filter PtByPt

#### VISA Find Resource Function

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

#### Deprecated VIs, Functions, and Nodes

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

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

## Floating-Point Math Operations

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



**Note** Refer to the National Instruments website at [ni.com/info](http://ni.com/info) and enter the Info Code `exdj8b` for more information about mathematical operations using floating-point numbers.

## Creating LabVIEW Classes

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

## Building an Installer (Windows)

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

## Using the Correct Calling Convention in a Call Library Function Node

In LabVIEW 8.5, LabVIEW 8.6, and LabVIEW 2009, when you specify an incorrect calling convention for a Call Library Function Node, LabVIEW recovers from the error and uses the correct calling convention. LabVIEW 2010 and LabVIEW 2011 do not perform this check, which requires you to select the correct calling convention yourself. Therefore, if you convert VIs that contain Call Library Function Nodes from LabVIEW 8.5, LabVIEW 8.6, or LabVIEW 2009 to LabVIEW 2010 or later, the VIs will crash if they have the incorrect calling convention selected.

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

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

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

## Compatibility Issues with LabVIEW and NI TestStand

LabVIEW 2010 and later has known compatibility issues with TestStand 4.2.1 and earlier versions. Refer to the National Instruments website at [ni.com/info](http://ni.com/info) and enter the Info Code `exvaku` to access the KnowledgeBase article that details these issues.



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

## Upgrading from LabVIEW 2010

You might encounter the following compatibility issues when you upgrade to LabVIEW 2013 from LabVIEW 2010. Refer to the *Upgrading from LabVIEW 2011* and *Upgrading from LabVIEW 2012* sections of this document for information about other upgrade issues you might encounter.

### VI and Function Behavior Changes

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

### Deprecated VIs, Functions, and Nodes

In LabVIEW 2011 and later, the Zero Phase Filter VI no longer has the **init/cont** input in any of its polymorphic instances. To use the new version of the VI, replace instances of the Zero Phase Filter VI from previous versions of LabVIEW with the VI of the same name from the Filters palette.

### Property, Method, and Event Behavior Changes

The behavior of the following properties, methods, and events changed in LabVIEW 2011 and later:

- In LabVIEW 2010, the Clear Compiled Object Cache method clears the object cache associated with a specific target. In LabVIEW 2011 and later, the Clear Compiled Object Cache method clears the entire user cache for the running version of LabVIEW. Therefore, although VIs created in LabVIEW 2010 that contain the Clear Compiled Object Cache method do not break in LabVIEW 2011 and later, they delete more VI object files than they did previously, which causes the associated VIs to recompile when loaded.
- In LabVIEW 2010 and earlier, the **NewRange** event data field for the Scale Range Change event ignores custom offset and multiplier values you set for a graph or chart. In LabVIEW 2011 and later, the **NewRange** event data field factors in custom offset and multiplier values in the results it returns. If you use code to work around this issue in LabVIEW 2010 or earlier, you must update the upgraded version of the code.

### Deprecated Properties, Methods, and Events

LabVIEW 2011 and later do not support the Subsystem From Selection method of the SimDiagram class.

### Migrating Build Specifications for Targets That Do Not Support SSE2 Instructions

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

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

### Transferring Flattened Data between Different Versions of LabVIEW

In LabVIEW 2011 and earlier, you transfer data between versions of LabVIEW using the Flatten To String and Unflatten From String functions. In LabVIEW 2012, use the VariantFlattenExp VI in the `labview\vi.lib\Utility` directory to transfer this data. The VariantFlattenExp VI accepts a hex integer of the target version of LabVIEW to which you want to transfer data.

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

In LabVIEW 2011 and later, if you wire extended-precision numeric data to the terminal of a polymorphic VI that supports both the double-precision numeric and 64-bit integer data types, LabVIEW coerces the extended-precision numeric data to double-precision data.

This behavior matches the behavior in LabVIEW 8.5 and 8.6. However, in LabVIEW 8.0, 8.2, 2009, and 2010, LabVIEW selects the 64-bit integer data type instead of the double-precision data type.

## Improved Error Reporting for Certain LabVIEW Shared Libraries

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

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

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

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

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

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

## Compatibility Issues with LabVIEW 2011 and Additional National Instruments Software

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

LabVIEW 2011 supports Measurement Studio 8.0 and later. Refer to the National Instruments website at [ni.com/info](http://ni.com/info) and enter the Info Code `exd8yy` to access the Upgrade Advisor and purchase Measurement Studio 8.0 or later.

## Upgrading from LabVIEW 2011

You might encounter the following compatibility issues when you upgrade to LabVIEW 2013 from LabVIEW 2011. Refer to the *Upgrading from LabVIEW 2012* section of this document for information about other upgrade issues you might encounter.

### Deprecated VIs, Functions, and Nodes

LabVIEW 2012 and later do not support the following VIs, functions, and nodes:

- **Polar Plot**—Use the Polar Plot with Point Options VI instead. The Polar Plot with Point Options VI provides two new inputs, **Lines/Points** and **Size**.
- **Draw Rect**—Use the Draw Rectangle VI instead.

## Property, Method, and Event Behavior Changes

In the Set Cell Value method of the Table class, the **X Index** and **Y Index** inputs changed from 32-bit unsigned integers to 32-bit signed integers.

## Deprecated Properties, Methods, and Events

LabVIEW 2012 and later do not support the following properties, methods, and events:

- Create from Data Type method of the Diagram class. If you upgrade a VI that contains this method, the VI now calls the Create from Data Type (Deprecated) method. Replace the deprecated method with the new Create from Data Type method, which no longer includes the **style** input.
- Frames[] property of the TimeFlatSequence class. Use the Frames[] property of the FlatSequence class instead.
- Front Panel Window:Open property of the VI class. Use the Front Panel:Open method, the Front Panel:Close method, or the Front Panel Window:State property instead.
- FPWinOpen property of the VI (ActiveX) class. Use the OpenFrontPanel method, the CloseFrontPanel method, or the FPState property instead.
- Static Member VIs property of the LVClassLibrary class. Use the new version of the Static Member VIs[] property instead.
- Dynamic Member VIs property of the LVClassLibrary class. Use the new version of the Dynamic Member VIs[] property instead.

## Renamed Properties, Methods, and Events

The following properties, methods, and events are renamed in LabVIEW 2012 and later.

Class	LabVIEW 2011 Name	LabVIEW 2012 and Later Name	Type
ProjectItem	Children[]	Owned Items[]	Property
ProjectItem	Parent	Owner	Property
LVClassLibrary	AncestorControlRefs	Ancestor Restricts Reference Creation	Property

## Upgrading from LabVIEW 2012

You might encounter the following compatibility issues when you upgrade to LabVIEW 2013 from LabVIEW 2012.

### VI and Function Behavior Changes

The behavior of the following VIs and functions changed in LabVIEW 2013.

#### Web Services VIs

The following VIs on the Web Services palette are rewritten in LabVIEW 2013. The VIs include a **LabVIEW Web Service Request** input, which replaces the **httpRequestID** input. To use the new functionality, replace the deprecated VIs with VIs of the same name from the Web Services palette.

- Web Services palette:
  - Read All Form Data
  - Read All Request Variables
  - Read Form Data
  - Read Postdata
  - Read Request Variable
  - Read Uploaded Files Info

- Output subpalette:
  - Flush Output
  - Render ESP Template
  - Set ESP Variable
  - Set HTTP Header
  - Set HTTP Redirect
  - Set HTTP Response Code
  - Set HTTP Response MIME Type
  - Write Response
- Security subpalette:
  - Decrypt
  - Encrypt
  - Get Auth Details
- Sessions subpalette:
  - Check if Session Exists
  - Create Session
  - Delete Session Variable
  - Destroy Session
  - Get Session ID Cookie
  - Read All Session Variables
  - Read Session Variables
  - Write Session Variables

### **Changes to the Behavior of the Event Structure Timeout Terminal for Non-Handled, Dynamically Registered Events**

In LabVIEW 2012 and earlier, when you dynamically register for events, any event you do not configure the Event structure to handle can reset the timeout terminal when that event occurs. For example, if you use the Register For Events function to register for the Mouse Up, Mouse Down, and Mouse Move events but configure an Event structure to handle only the Mouse Up and Mouse Down events, the timeout terminal resets when the Mouse Move event occurs.



**Note** The timeout terminal resets only if you wire a value to that terminal.

In LabVIEW 2013, non-handled, dynamically registered events do *not* reset the Event structure timeout terminal.

### **Changes to the Default .NET Framework**

In LabVIEW 2013, creating and communicating with .NET objects requires the .NET Framework 4.0. The .NET Framework 4.0 allows you to load pure managed assemblies built in any version of the .NET Framework and mixed-mode assemblies built in .NET 4.0. The LabVIEW 2013 installer includes the .NET Framework 4.0. However, if you uninstall the .NET Framework 4.0 or attempt to load assemblies that target a different version of the .NET Framework, LabVIEW may return an error when you attempt to create or communicate with .NET objects.

LabVIEW 2013 loads the Common Language Runtime (CLR) 4.0 by default. However, you can force LabVIEW to load .NET mixed-mode assemblies that target the CLR 2.0.

Refer to the **Fundamentals»Windows Connectivity»How-To».NET»Loading .NET 2.0, 3.0, and 3.5 Assemblies in LabVIEW** topic on the **Contents** tab of the *LabVIEW Help* for more information about loading assemblies in LabVIEW.

### Changes to the System Button

In LabVIEW 2012 and earlier, when you add the system button to the front panel from the **System** palette, the return key toggles the value by default. In LabVIEW 2013, LabVIEW does not provide default key binding for the system button.

### Changes to the Value and Value (Signaling) Properties

In LabVIEW 2012 and earlier, if you set the value of a latched Boolean control with the Value or Value (Signaling) property, LabVIEW returns an error. However, if you change the latched Boolean control to a type definition, LabVIEW no longer returns an error. In LabVIEW 2013, to avoid race conditions, the Value and Value (Signaling) properties always return an error if you attempt to set the value of a latched Boolean control.

### Improvements to the Performance of Conditional Tunnels

In LabVIEW 2012, you can use the **Conditional** tunnel option to include only the values you specify in each output tunnel of a loop, but National Instruments recommends you consider alternatives to the conditional tunnel in parts of the application where performance is critical. In LabVIEW 2013, performance improvements to conditional tunnels reduce memory allocations for the **Conditional** tunnel option.

### Wiring Custom Controls to a Subpanel

LabVIEW returns an error if you wire a custom control to the Insert VI method in the SubPanel class. To wire a custom control to a subpanel, add the control to the front panel of a VI and wire that VI to the subpanel.

### Using NI Web-Based Configuration & Monitoring with SSL

In LabVIEW 2012 and earlier, you view and edit Secure Sockets Layer (SSL) certificates and Signing Requests (CSRs) from the NI Distributed System Manager. The Distributed System Manager no longer supports this functionality.

You now can create, edit, view, and remove SSL certificates and CSRs from NI Web-based Configuration & Monitoring. From the NI Web-based Configuration & Monitoring utility, navigate to the Web Server Configuration page and display the SSL Certificate Management tab to manage your SSL certificates and CSRs.

### Creating and Publishing LabVIEW Web Services


In LabVIEW 2013, you no longer use RESTful Web Service build specifications to create Web services or to configure properties, such as URL mappings, for Web services. You can continue to use build specifications you created in LabVIEW 2012 or earlier, or you can convert them to Web service project items. To download a tool that performs the conversion, visit [ni.com/info](http://ni.com/info) and enter the Info Code `ConvertWS`.

If you convert a Web service to the LabVIEW 2013 format, you can access most of the options from LabVIEW 2012 and earlier for configuring a Web service build specification by right-clicking the Web service project item in a project and selecting **Properties**. However, the following table describes Web service behaviors and options available in LabVIEW 2012 and earlier that are changed or removed in LabVIEW 2013.

LabVIEW 2012 and Earlier	LabVIEW 2013
The term <i>Web method VI</i> refers to the VIs that receive HTTP requests from clients and return data to clients.	The concept of Web method VIs is renamed <i>HTTP method VIs</i> .
You can define service aliases for the Web service name, which customize the URL clients use to access the service.	Use the exact service name to access the Web service.
You can map multiple URLs to a Web method VI.	You can map only a single URL to an HTTP method VI. To allow multiple URLs to invoke the same VI, use it as a subVI in multiple HTTP method VIs, each of which has its own URL mapping.
You can specify values that override the default values of connector pane terminals of the VI.	This option is removed because you cannot map multiple URLs to an HTTP method VI. Thus you cannot create alternate URL mappings that rely on the override behavior.
You can mark VIs in the project as auxiliary VIs, meaning they exchange data with Web method VIs but are not exposed to clients.	The concept of auxiliary VIs is renamed <i>startup VIs</i> . LabVIEW considers all VIs you place under the <b>Startup VIs</b> project item in the project to be startup VIs.
You can disable "stand-alone" deployment for a Web service, which means the Web service is only deployed when the LabVIEW development system is open.	This option is removed.
You can set VIs to run as pre- and post-build steps that run when you build the Web service.	This feature is not available because you do not build Web services from build specifications.

Refer to the *Creating and Publishing LabVIEW Web Services* section for information about how to create, debug, and publish Web services in LabVIEW 2013.

## LabVIEW 2013 Features and Changes

The Idea Exchange icon  denotes a new feature idea that originates from a product feedback suggestion on the LabVIEW Idea Exchange discussion forums. Refer to the National Instruments website at [ni.com/info](http://ni.com/info) and enter the Info Code `ex3gus` to access the NI Idea Exchange discussion forums.

Refer to the `readme.html` file in the `labview` directory for known issues, a partial list of bugs fixed, additional compatibility issues, and information about late-addition features in LabVIEW 2013.

## LabVIEW Web Services Enhancements

LabVIEW 2013 includes the following enhancements to LabVIEW Web services and related functionality.

### Creating and Publishing LabVIEW Web Services

LabVIEW 2013 allows you to more easily create, debug, and publish LabVIEW Web services. Refer to the following table for information about how LabVIEW 2013 improves several tasks related to Web services.

Task	LabVIEW 2012 and Earlier	LabVIEW 2013	LabVIEW Help topic with More Information
Creating a Web Service	<ol style="list-style-type: none"> <li>1. Add VIs and other files that make up the Web service to a LabVIEW project.</li> <li>2. Create a RESTful Web service build specification and configure build settings, such as which files to include and their URL mappings.</li> <li>3. Build the Web service build specification each time you update a Web service source file or a build setting.</li> </ol>	<ol style="list-style-type: none"> <li>1. Add a Web service project item to a project under <b>My Computer</b> or a remote target.</li> <li>2. Add files under the item and configure settings from the project. LabVIEW automatically includes all files under the project item in the Web service when you publish it.</li> </ol>	Tutorial: Creating and Accessing a LabVIEW Web Service
Publishing Stand-Alone Web Services	Deploy the built Web service to the host computer or a target.	Publish the Web service files directly from the project by right-clicking the Web service project item and selecting <b>Application Web Server»Publish</b> .	Publishing a Web Service
Publishing Web Services that Depend on a LabVIEW Stand-Alone Application	<ul style="list-style-type: none"> <li>• Run the built application, and then deploy the built Web service.</li> <li>• The Web service runs on the Application Web Server.</li> </ul>	<ul style="list-style-type: none"> <li>• Include the Web service in the build specification for the application. The built application automatically publishes the Web service when it runs.</li> <li>• The Web service runs on a web server that is specific to the application.</li> </ul>	Including a Web Service in a Stand-alone Application or Installer
Communicating with Owing Applications	Web services run in a separate context from the main LabVIEW application instance, so communication between the Web service and stand-alone applications is limited to communication features that support cross-context communication, such as network streams.	When you include a Web service in a stand-alone application, the Web service runs in the main application instance. This allows you to implement communication via many protocols and APIs that LabVIEW provides.	-

Task	LabVIEW 2012 and Earlier	LabVIEW 2013	LabVIEW Help topic with More Information
Debugging a Web Service	<ol style="list-style-type: none"> <li>1. Add breakpoints to Web method VIs.</li> <li>2. Enable debugging in the Web service build specification.</li> <li>3. Build and deploy the Web service.</li> <li>4. Use the <b>Debug Application or Shared Library</b> dialog box to connect to the Web service and open the Web method VI.</li> <li>5. Use typical debugging techniques to debug the issue in the source files.</li> </ol>	<ol style="list-style-type: none"> <li>1. Start a debugging session from the project by right-clicking a Web service project item and selecting <b>Start</b>.</li> <li>2. Use typical debugging techniques to debug the issue in the source files.</li> </ol>	Testing and Debugging a Web Service
Integrating Static Content	<ul style="list-style-type: none"> <li>• Add folders of static content to the project.</li> <li>• Include the static content files in the build specification and define URL mappings and destinations for each file and folder.</li> </ul>	Add public and private static-content folders under the Web service project item. LabVIEW automatically assigns a URL mapping to each public static file.	Integrating Static Content into a Web Service
Reusing a Web Service on Multiple Targets	You cannot copy a Web service build specification from one target to another.	To copy a Web service and its properties to another target in a project, click and drag the parent Web service project item to the other target.	-

## Enabling Secure Communication with Web Services

If SSL Support for LabVIEW RT is installed, you can configure secure communication with Web services using port 3581. You configure Web services communication using the NI Web-based Configuration & Monitoring utility, which you can access by entering the URL `http://localhost:3582` in a Web browser on the host system.

Complete the following steps to enable secure communication with Web services:

1. Select **SSL (HTTPS) Enabled** on the **Web Server Configuration** page in NI Web-based Configuration & Monitoring.
2. Specify the **SSL (HTTPS) Port** as 3581.
3. Disable **HTTP Enabled**.
4. Click the **Apply** button.

Refer to the **Fundamentals»Working with Projects and Targets»How-To»Monitoring and Configuring a Remote Device from a Web Browser** topic on the **Contents** tab of the *LabVIEW Help* for more information about NI Web-based Monitoring & Configuration.

In previous versions of LabVIEW, LabVIEW Web Services uses port 3580, and you cannot adjust the HTTPS settings.



## New and Changed Web-Service Related VIs

LabVIEW 2013 includes the following new and changed VIs.

### Web Services VIs

The Web Services palette includes a new Service subpalette. Unlike other Web Services VIs, you can call the Service VIs from startup VIs as well as HTTP method VIs. This palette includes the following VIs:

- **Get Web Service Status**—Returns whether a Web service is stopping.
- **Read Service Attribute**—Returns information about a Web service, including the paths to published private and public content folders.

### SMTP Email VIs

LabVIEW 2013 includes all new SMTP Email VIs on the SMTP Email palette. Use the SMTP Email VIs to send email, including attached data and files. LabVIEW sends the email using the Simple Mail Transfer Protocol (SMTP). For added security, you can use Transport Layer Security (TLS) to communicate with the SMTP server. You also can specify authentication credentials for the SMTP client.

The SMTP Email palette includes the Send Email Express VI. You can use the Send Email Express VI to send emails quickly from LabVIEW to a list of recipients you specify. However, if you want to configure headers, set TLS settings, or include an attachment, you must use the standard SMTP Email VIs to configure and send an email with more advanced settings.

### WebDAV VIs

LabVIEW 2013 includes a new WebDAV palette and new VIs that you can use to transfer files securely to and from a target. The WebDAV palette also includes the more advanced WebDAV Synchronous and WebDAV Asynchronous subpalettes.

Use the WebDAV Synchronous VIs to securely manage files on and transfer files to and from a WebDAV target using a synchronous interface. You can send requests to the WebDAV server only one at a time using the WebDAV Synchronous VIs.

Use the WebDAV Asynchronous VIs to securely manage files on and transfer files to and from a WebDAV target using an asynchronous interface. You can send multiple requests to the WebDAV server simultaneously using the WebDAV Asynchronous VIs.


## Productivity Enhancements to Event-Driven Programming in LabVIEW

LabVIEW 2013 includes the following enhancements to event-driven programming and related functionality.

### Viewing Enqueued Events at Run Time

In LabVIEW 2013, you can inspect the events waiting in an event queue associated with an Event structure. You also can view a list of VIs that contain Event structures with registered events and a log of the events the Event structure handled. Right-click an Event structure and select **Event Inspector Window** to view the event information.

### Customizing When a Control Responds to Movements of the Mouse Wheel

 In LabVIEW 2013, you can specify when a control responds to mouse wheel movements. To do this, navigate to the **Key Navigation** page of the **Properties** dialog box for that control. Then select from the available options for the **Built-In Mouse Wheel Support** component. For example, you can select whether a control responds to mouse wheel movements **On Hover** or **On Key Focus**.

*[Idea submitted by NI Discussion Forums members tst and jacemdom1]*

## Enhancements to the Edit Events Dialog Box

The **Edit Events** dialog box includes a checkbox that allows you to specify a maximum number of event instances to keep in an event queue. If you enable **Limit maximum instances of this event in event queues**, LabVIEW maintains the queue limit by automatically discarding older events from the event queue as newer events of the same event type arrive.

## New and Changed Events Functions

LabVIEW 2013 includes the following new and changed functions:

- The Events palette includes a new Flush Event Queue function. You can use this function to discard the least recent notify events from one or more event queues.
- The Generate User Event function includes a **priority** input that allows you to specify `normal priority` or `high priority` for the user event. When you specify `high priority`, LabVIEW enqueues the user event and associated event data into the event queue in front of any previously generated normal priority events.


## Programmatically Controlling Event Structures

LabVIEW 2013 includes new VI Scripting properties and methods to control Event structures. Refer to the **LabVIEW 2013 Features and Changes»New VI Scripting Objects** topic on the **Contents** tab of the *LabVIEW Help* for lists of these properties and methods.

## Block Diagram Enhancements

LabVIEW 2013 includes the following enhancements to the block diagram and related functionality.


### Attaching Comments to Block Diagram Objects

 In LabVIEW 2013, you can attach free labels to block diagram objects to keep comments associated with specific objects. LabVIEW keeps the comments attached to the objects while you arrange the block diagram or use the **Clean Up Diagram** button. You can attach multiple, different comments to each object, but you cannot attach the same comment to more than one object.

To attach a comment, hover over a comment to reveal a glyph in the bottom right corner. Then, click the glyph and then click a block diagram object.

*[Idea submitted by NI Discussion Forums member Chris\_H.]*

### Managing Unfinished Tasks in LabVIEW

 You can manage a list of tasks, or to-do items, using hashtags (#) in labels and block diagram comments. A hashtag followed by text is called a bookmark. When you add a hashtag to a block diagram comment, LabVIEW detects a bookmark and bolds the bookmark tag to distinguish it from the rest of the text, such as **#rewrite** algorithm. Use bookmarks to identify incomplete code that needs further development. To find all bookmarks in your application, select **View»Bookmark Manager** to display the **Bookmark Manager** window.



**Note** You can place bookmarks only in labels and block diagram comments. You cannot use these bookmarks for control or indicator labels.

*[Idea submitted by NI Discussion Forums member Edupo]*

## Front Panel Enhancements

### Enhancements to Graphs

LabVIEW 2013 includes the following enhancements to the mixed signal and digital waveform graphs.

#### Grouping Digital Lines into Buses on a Mixed Signal Graph

In previous versions of LabVIEW, when you wire digital data to a mixed signal graph, the mixed signal graph displays individual digital lines for each digital data set, and LabVIEW groups them under a parent node.

In LabVIEW 2013, when you wire digital data to a mixed signal graph, the mixed signal graph groups digital lines into buses, which display the combined value of all of the lines within them, similar to the digital waveform graph. If you wire an array of digital data where each array element represents a bus, the mixed signal graph plots each column of digital data per element as a different line.

To compare data more closely, you can move buses to another plot area. To move a digital plot from one plot area to another, you must move the entire bus. To move a bus, select the icon in the plot legend next to the name of the bus you want to move, and drag the icon to another plot area in the plot legend.



**Note** You cannot separate digital lines from a bus on a mixed signal graph because separating a line from a bus breaks the data representation.

#### Formatting Bus Labels with Fixed-Point Representation

In LabVIEW 2013, you can format a bus label on a mixed signal or digital waveform graph with fixed-point representation by right-clicking a bus in the plot legend and selecting **Label Format»Fixed-Point** from the shortcut menu. To configure the encoding and range of fixed-point labels, right-click a digital bus and select **Label Format»Configure Fixed Point** to display the **Configure Fixed Point** dialog box.

Refer to the **Configure Fixed Point Dialog Box (Mixed Signal and Digital Waveform Graphs)** topic in the *LabVIEW Help* for more information about configuring the fixed-point digital bus label.

You also can use the Fixed Point Style property, Fixed Point Settings property, and Fixed Point Digits of Precision property to configure the **Fixed Point** label format programmatically.

Refer to the **Property and Method Reference»VI Server»Generic»Bus»Properties** book on the **Contents** tab of the *LabVIEW Help* for more information about fixed point properties.

LabVIEW 2013 includes the following enhancements to the LabVIEW environment.

### Dialog Box Enhancements

LabVIEW 2013 includes the following dialog box enhancements.

#### Creating Reports to Compare VIs and VI Hierarchies

In the LabVIEW Professional Development System, LabVIEW 2013 allows you to create a comparison report after comparing VIs and VI hierarchies. Select **Tools»Compare** to compare VIs or VI hierarchies. To save the comparison in a report, click **Create Report** in the **Compare VI Hierarchies** dialog box or in the **Differences** dialog box. You can save the report as a Web page (.xml) or as a text file (.txt). (**Windows**) You also can save the report as a Microsoft Word document (.doc).



**Note** You must have Microsoft Word installed to create a Microsoft Word report.

The comparison report summarizes all the differences between the VIs or VI hierarchies that you compare. The Web page and Microsoft Word reports also contain screenshots of the differences on the front panel and block diagram. You can view a list of the differences and details in the **Differences** dialog box and **Compare VI Hierarchies** dialog box.



**Note** The comparison report is only in English.

Refer to the **Fundamentals»Application Development and Design Guidelines»How-To»Comparing VIs and VI Hierarchies** book on the **Contents** tab of the *LabVIEW Help* for more information about creating comparison reports.

## Application Builder Enhancements

### Automatically Selecting NI Software for Installers

When you build an installer in LabVIEW 2013, LabVIEW automatically selects installers for the drivers and other software components required by the built application. Use this feature to reduce the possibility of building an installer without the right components. To disable this feature, remove the checkmark from the **Automatically select recommended installers** checkbox on the **Additional Installers** page of the **Installer Properties** dialog box for the installer.

*[Idea submitted by NI Discussion Forums member jlokanis]*

### Creating Directory Versions in Build Specifications

In LabVIEW 2012 and earlier, if you create a build specification, LabVIEW does not include the build version number in the directory path on disk. In LabVIEW 2013, you can use tags in the build destination path so LabVIEW automatically includes the build version in the directory path. You can include the `[VersionNumber]` tag in the **Destination path** field on the **Destinations** page or the **Destination directory** field on the **Information** page of the build specification properties dialog box.

The following table lists the tags to use for each build specification and the resulting directory name on disk.

Build Specification	Tag	User-Defined Path	Directory Created on Disk
Application (EXE)	<code>[VersionNumber]</code>	C:\temp\builds\[VersionNumber]	C:\temp\builds\1.0.0.0
Installer	<code>[ProductVersion]</code>	C:\temp\builds\[ProductVersion]	C:\temp\builds\1.0.0
.NET Interop Assembly	<code>[VersionNumber]</code>	C:\temp\builds\[VersionNumber]	C:\temp\builds\1.0.0.0
Packed Library	<code>[VersionNumber]</code>	C:\temp\builds\[VersionNumber]	C:\temp\builds\1.0.0.0
Shared Library (DLL)	<code>[VersionNumber]</code>	C:\temp\builds\[VersionNumber]	C:\temp\builds\1.0.0.0
Source Distribution	<code>[VersionNumber]</code>	C:\temp\builds\[VersionNumber]	C:\temp\builds\1.0.0.0

For example, if you build a stand-alone application and include the `[VersionNumber]` tag in the **Destination directory** field on the **Information** page of the **Application Properties** dialog box, LabVIEW builds the application to a path that includes a folder with the name of the current version.



**Note** If you enable the **Auto increment** checkbox, or **Auto increment product version** checkbox for installers, on the **Version Information** page, and you use the `[VersionNumber]` or `[ProductVersion]` tags, LabVIEW creates a new directory that includes the version number instead of overwriting the previous build on disk.

## Building Applications for a Target That Runs Windows Embedded Standard

You can use the Application Builder to build certain types of applications on targets that run the Windows Embedded Standard operating system. However, you cannot use the **Project Explorer** window to deploy any type of application to a target that runs this operating system. To deploy any application to a target that runs Windows Embedded Standard, you must copy the application files from the development computer to the target.

### Developing Applications on Windows Embedded Standard Targets

You can develop LabVIEW applications on National Instruments hardware that runs the Windows Embedded Standard operating system. By adding a Windows Embedded Standard target to a LabVIEW project on the host computer, you can debug and run VIs on the target from the host computer.


LabVIEW 2013 supports the following National Instruments hardware for developing applications on Windows Embedded Standard targets:

- NI cRIO-9081
- NI cRIO-9082

For the host computer to connect to the Windows Embedded Standard target, the target must enable NI LabVIEW Remote Development Target Support. Refer to the *NI LabVIEW Remote Development Target Support Readme*, available on the target or on the LabVIEW Remote Development Target Support software installer, for more information about LabVIEW Remote Development Target Support and how to configure the Windows Embedded Standard target for remote development of LabVIEW applications.

Refer to the **Fundamentals»Working with Projects and Targets»Concepts»Working with Windows Embedded Standard Targets** topic on the **Contents** tab of the *LabVIEW Help* for more information about developing applications on Windows Embedded Standard targets.

### Enhancements to Troubleshooting Errors Encountered While Building an Application

 LabVIEW 2013 includes improvements to the most common error messages you can encounter during the build process for the following LabVIEW builds:

- Stand-alone applications
- .NET interop assemblies
- Packed libraries
- Shared libraries

The error dialog returns more explicit error messages and includes a link to the LabVIEW Application Builder support Web page at [ni.com](http://ni.com). Use this support page to troubleshoot an error message or search through the related resources for a specific type of application.

Refer to the National Instruments website at [ni.com/info](http://ni.com/info) and enter the Info Code `appbuilder_support` to access the LabVIEW Application Builder support Web page.

*[Idea submitted by NI Discussion Forums member RavensFan]*

### Including Web Services in a Stand-Alone Application or Installer

When you build a stand-alone application or installer, LabVIEW can include a Web service in the built application or installer so the Web service runs when the resulting build runs. To include a Web service in an application or installer, right-click an application or installer build specification, select **Properties**, and browse to the new **Web Services** page of the properties dialog box for the build specification. The Web service must exist in the same project as the build specification.

Refer to the *Creating and Publishing LabVIEW Web Services* section for more information about improvements to the process of creating Web services in LabVIEW 2013.

## Enhancements to the Caching Behaviors for Installers

If the distribution media is unavailable when you use the Application Builder to build an installer that contains additional installers or components, the Application Builder prompts you to locate the original distribution for the components. In previous versions of LabVIEW, on the **Additional Installers** page of the **Installer Properties** dialog box, you can enable options to cache the distribution components from their original media location or cache the components when you run the installers or both.

LabVIEW 2013 combines the previous caching options into one checkbox. Enable the **Cache for Future Distributions** checkbox to copy the selected installers from the original media location *and* all future installers to the computer.

## New and Changed VIs, Functions, and Nodes

LabVIEW 2013 includes the following new and changed VIs, functions, and nodes.

### Application Control VIs and Functions

The Application Control palette includes the following new functions:

- Get Control Values by Index
- Set Control Values by Index

Use these functions to get or set control values faster than is possible with other VI Server objects, such as the Value property in the Control class. You can use these functions to get or set control values almost as fast as when you wire directly from or to a control terminal. However, these functions require more advanced application design than other methods for getting and setting control values.

### Cluster, Class, & Variant VIs and Functions

The Cluster, Class, & Variant palette includes the following new VIs:

- Get LV Class Name
- Get LV Class Default Value By Name

### Flatten/Unflatten String Functions

The new Flatten/Unflatten String palette includes the following new functions:

- Flatten To JSON
- Unflatten From JSON

The Flatten/Unflatten String palette also includes the following functions:

- Flatten To String
- Flatten To XML
- Flattened String To Variant
- Unflatten From String
- Unflatten From XML
- Variant To Flattened String

### Memory Manager Functions

LabVIEW 2013 includes the following new memory manager functions:

- DSNewAlignedHandle—Creates a new handle to a relocatable block of memory of the specified size, alignment, and alignment offset.
- DSNewAlignedHClr—Creates a new handle to a relocatable block of memory of the specified size, alignment, and alignment offset, and initializes the memory to zero.
- DSSetAlignedHandleSize—Changes the size, alignment, and alignment offset of the block of memory referenced by the specified handle.

- **DSSetAlignedHSzClr**—Changes the size, alignment, and alignment offset of the block of memory referenced by the specified handle, and sets any new memory to zero.

You can call these memory manager functions from external C/C++ code to allocate and resize memory blocks with specific alignment characteristics. These functions are most useful when you allocate arrays of LabVIEW data that you plan to access with specific operations that perform better when you align the data at certain address boundaries. For example, you can use these functions to allocate arrays of LabVIEW data that you plan to access with SSE or AVX vector instructions or DMA data transfers between LabVIEW-allocated memory and hardware.

## Changed VIs and Functions

The following VIs and functions changed in LabVIEW 2013.

### TDM Streaming VIs and Functions

In LabVIEW 2013, the TDM Streaming VIs and functions support additional data types, such as single-precision and double-precision complex floating-point numbers. This palette also includes the following changes:

- **TDMS Generate Random Data**—This VI includes the new CSG and CDB instances. You can use these instances to generate random complex floating-point numbers with single precision or double precision, respectively.
- **TDMS Open**—This function includes the new **create index file?** input, which specifies whether LabVIEW automatically generates a `.tdms_index` file for the corresponding `.tdms` file. This index file enables LabVIEW to speed up random access to the `.tdms` file. If you have limited disk space, wire a FALSE value to this input to prevent LabVIEW from generating the `.tdms_index` file. The default is TRUE.
- **TDMS File Viewer**—This VI includes the new **Go To** button in the **Values (table)** page of the **TDMS File Viewer** dialog box. Use this button to specify the index number of a data value you want to view.
- **TDMS Flush**—This function includes internal improvements and is more reliable than the same function in previous versions of LabVIEW. You can use this function to force the operating system to write any buffer data to a `.tdms` file. However, using this function might negatively impact the performance of your TDMS application.

### Waveform File I/O VIs

- **Export Waveforms to Spreadsheet File**—This VI includes a new Export Waveforms To Spreadsheet File (Digital) instance.
- **Read Waveforms from File**—This VI includes a new Read Waveform from File (Digital) instance.
- **Write Waveforms to File**—This VI includes a new Write Waveform to File (Digital) instance.

### Miscellaneous VI and Function Changes

- **Amplitude and Level Measurements**—The **Amplitude Measurements** parameter in the **Configure Amplitude and Level Measurements** dialog box of this Express VI contains the following renamed options:
  - **DC** is renamed **Mean (DC)**.
  - **Maximum peak** is renamed **Positive peak**.
  - **Minimum peak** is renamed **Negative peak**.
- **Write To Measurement File**—This Express VI includes the new **Microsoft Excel (.xlsx)** option in the configuration dialog box. Select this option to write data to a Microsoft Excel file. This Express VI first stores data in a temporary file and then flushes the data to the Excel file. Use the new **Flush?** block diagram input to specify the frequency at which you want to flush the data. (**OS**

**X and Linux)** In LabVIEW 2012 and earlier, this Express VI can create and save only .lvn files. In LabVIEW 2013, this Express VI can create and save additional file types, such as .tdms and .xlsx files.

## New and Changed Classes, Properties, Methods, and Events

LabVIEW 2013 includes new VI Server classes, properties, methods, and events. Refer to the **LabVIEW 2013 Features and Changes»New VI Server Objects** topic on the **Contents** tab of the *LabVIEW Help* for a list of new classes, properties, methods, and events.

LabVIEW 2013 includes changes to the Get VI Dependencies (Names and Paths) method. Use the **Include Alternate Call Setups VIs** parameter to return subVI calls that use the **Reload for each call** or **Load and retain on first call** options from the **VI Call Configuration** dialog box. Use the **Load Block Diagram?** parameter to load the block diagram of a VI before LabVIEW evaluates the application for any dependencies.

## Updates to LabVIEW Examples

LabVIEW 2013 includes redesigned and reorganized examples to better demonstrate LabVIEW programming. Refer to the examples folder in the `labview` directory to view the new locations and updates for these examples.

## Improved Installation Experience for the VI Package Manager (Windows)

You can install the VI Package Manager (VIPM) software from the LabVIEW Platform DVDs.

## Features and Changes in Previous Versions of LabVIEW

---

To identify new features in each version of LabVIEW that released since your previous version, review the upgrade notes for those versions. To access these documents, refer to the National Instruments website at [ni.com/info](http://ni.com/info) and enter the Info Code for the appropriate LabVIEW version from the following list:

- *LabVIEW 2009 Upgrade Notes*—upnote9
- *LabVIEW 2010 Upgrade Notes*—upnote10
- *LabVIEW 2011 Upgrade Notes*—upnote11
- *LabVIEW 2012 Upgrade Notes*—upnote12



Refer to the *NI Trademarks and Logo Guidelines* at [ni.com/trademarks](http://ni.com/trademarks) for more information on National Instruments trademarks. Other product and company names mentioned herein are trademarks or trade names of their respective companies. For patents covering National Instruments products/technology, refer to the appropriate location: **Help»Patents** in your software, the `patents.txt` file on your media, or the *National Instruments Patent Notice* at [ni.com/patents](http://ni.com/patents). For end-user license agreements (EULAs) and copyright notices, conditions, and disclaimers, including information regarding certain third-party components used in LabVIEW, refer to the *Copyright* topic in the *LabVIEW Help*. Refer to the *Export Compliance Information* at [ni.com/legal/export-compliance](http://ni.com/legal/export-compliance) for the National Instruments global trade compliance policy and how to obtain relevant HTS codes, ECCNs, and other import/export data.