

LabVIEW™ Upgrade Notes

These upgrade notes describe the process of upgrading LabVIEW for Windows, OS X, and Linux to LabVIEW 2014. 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 2014

Contents

Upgrading to LabVIEW 2014.....	2
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.....	6
Upgrade and Compatibility Issues.....	6
Upgrading from LabVIEW 2009 or Earlier.....	6
Upgrading from LabVIEW 2010.....	6
Upgrading from LabVIEW 2011.....	8
Upgrading from LabVIEW 2012.....	9
Upgrading from LabVIEW 2013.....	12
LabVIEW 2014 Features and Changes.....	13
LabVIEW Version- and Bitness-Specific System Tray Icons.....	13
Installing and Updating DataPlugins in LabVIEW.....	14
Block Diagram Enhancements.....	14
Front Panel Enhancements.....	15
Environment Enhancements.....	15
New and Changed VIs, Functions, and Nodes.....	16
Application Builder Enhancements.....	18
Additional Functionality through Add-On Consolidation.....	19
New LabVIEW Suites.....	20
LabVIEW Web Services Enhancements.....	20
Providing Custom Documentation for VIs and Applications.....	20
LabVIEW 2014 (64-bit) for Mac OS X.....	21
LabVIEW 2014 (64-bit) for Linux.....	22
Changes to Touch Panel Functionality.....	22
New and Changed Toolkits.....	23
Features and Changes in Previous Versions of LabVIEW.....	23

Upgrading to LabVIEW 2014

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 and enter the Info Code `lifecycle` for information about which versions of LabVIEW still receive mainstream support.

Overview of the Recommended Upgrade Process

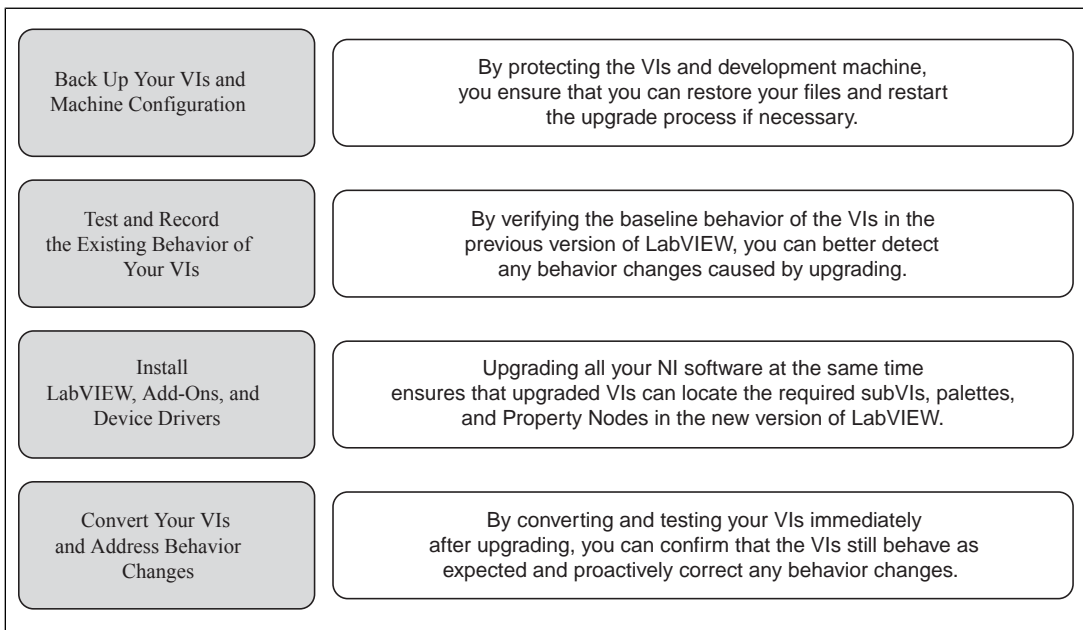


Figure 1.



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 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 2014, 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 2014 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 2014, copy files from the `labview\user.lib` directory in the previous version to the `labview\user.lib` directory in LabVIEW 2014.

4. Convert Your VIs and Address Behavior Changes

Mass compiling your VIs in LabVIEW 2014 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 2014. 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 2014 can occasionally cause some VIs to break if they use outdated features. To quickly identify and fix broken VIs in LabVIEW 2014, 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 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 2011 Upgrade VI Analyzer tests locate a potential behavior change, refer to the *Upgrading from LabVIEW 2010* 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 2014 Known Issues list—Lists bugs discovered before and throughout the release of LabVIEW 2014. Refer to the National Instruments website at ni.com/info and enter the Info Code `lv2014ki` 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 2014.

- **Your own tests**—Perform the same tests on the VIs in LabVIEW 2014 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 2014»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 2014 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 2014.

Upgrading from LabVIEW 2009 or Earlier

Refer to the National Instruments website at ni.com/info and enter the Info Code `oldUpgradeIssues` to access upgrade and compatibility issues you might encounter when you upgrade to LabVIEW 2014 from LabVIEW 2009 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 2010

You might encounter the following compatibility issues when you upgrade to LabVIEW 2014 from LabVIEW 2010. Refer to the *Upgrading from LabVIEW 2011*, *Upgrading from LabVIEW 2012*, and *Upgrading from LabVIEW 2013* 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.

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 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 2014 from LabVIEW 2011. Refer to the *Upgrading from LabVIEW 2012* and *Upgrading from LabVIEW 2013* sections of this document for information about other upgrade issues you might encounter.

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.

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 2014 from LabVIEW 2012. Refer to the *Upgrading from LabVIEW 2013* section of this document for information about other upgrade issues you might encounter.

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 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 Startup VIs 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.

Upgrading from LabVIEW 2013

Behavior Change in the String to Path Function

In LabVIEW 2014, the String To Path function is case insensitive when reading any variation of the string <Not A Path> and always returns a constant value of <Not A Path>. For example, you can specify <not a path> or <Not A Path> in the **string** input, and in both cases, the function returns a constant value of <Not A Path>. Refer to the following table for more information about how the String to Path function behaves in previous versions of LabVIEW.

LabVIEW 2012 and 2013	LabVIEW 2011 and Earlier
Regardless of case, the String To Path function does not return a constant value of <Not A Path>. You can specify any variation of the string <Not A Path>, and the function returns a path to a directory named <Not A Path> instead of returning a constant value of <Not A Path>.	Like LabVIEW 2014, the String To Path function is case insensitive and returns the constant value of <Not A Path> when you specify any variation of the string <Not A Path>. Whether you specify <not a path> or <Not a Path>, the function returns the constant value of <Not A Path>.

Reviewing and Updating Type Definitions

The **Review and Update from Type Def** shortcut menu item replaces the **Update from Type Def** shortcut menu item that appears in LabVIEW 2013 and earlier.

Deprecated VIs, Functions, and Nodes

LabVIEW 2014 does not support the following VIs, functions, and nodes.

Apple Event VIs

LabVIEW 2014 for Mac OS X no longer supports Apple Event VIs. Instead, use the Run AppleScript Code VI on the Libraries & Executables palette to communicate with OS X applications external to LabVIEW. If you attempt to load a VI that contains any of the following Apple Event VIs, LabVIEW may generate errors and be unable to run the VI:

- AESend Do Script
- AESend Finder Open
- AESend Open
- AESend Open Document
- AESend Print Document
- AESend Quit Application
- Get Target ID
- AESend Abort
- AESend Close
- AESend Open, Run, Close
- AESend Run
- AESend VI Active?
- AECreate Comp Descriptor
- AECreate Descriptor List
- AECreate Logical Descriptor
- AECreate Object Specifier
- AECreate Range Descriptor
- AECreate Record
- AESend

- Make Alias

Actor Framework VIs

LabVIEW 2014 does not support the Actor:Launch Actor VI. Use the Actor:Launch Root Actor VI or Actor:Launch Nested Actor VI instead.


In Port and Out Port VIs

LabVIEW 2014 does not support the In Port and Out Port VIs.

Deprecated Properties, Methods, and Events


LabVIEW 2014 does not support the Get VI:Old Help Info method of the Application class. Instead, use the Get VI:Help Info method to return help information from the **Documentation** page of the **VI Properties** dialog box for a specified VI.

LabVIEW 2014 Features and Changes

The Idea Exchange icon  denotes a new feature idea that originates from a product feedback suggestion on the NI Idea Exchange discussion forums. Refer to the National Instruments website at 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 2014.

LabVIEW Version- and Bitness-Specific System Tray Icons

 LabVIEW 2014 introduces the following new system tray icons that indicate the version and bitness of the installed LabVIEW:

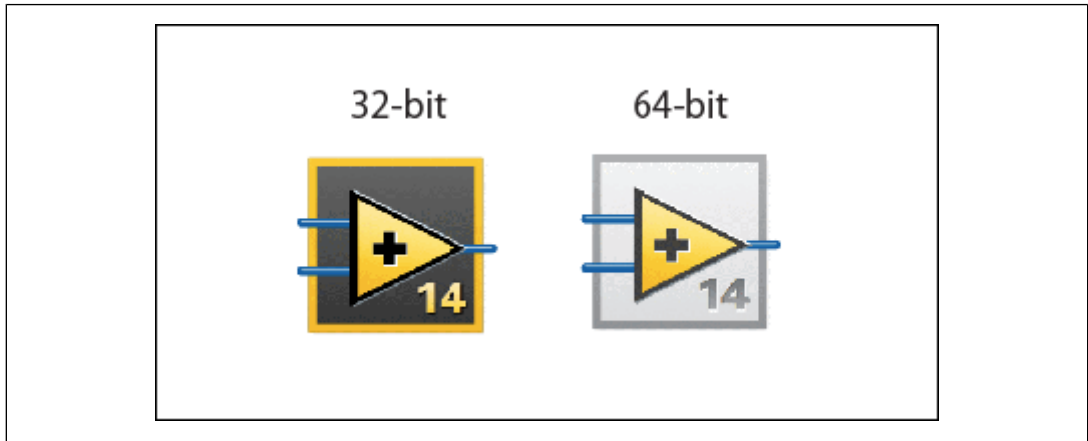


Figure 2.

You can use these new icons to help identify different versions of LabVIEW installed on a computer. For example, if you have LabVIEW 8.6 and LabVIEW 2014 installed, you can differentiate between the system tray icons to avoid opening a VI in the incorrect version of LabVIEW. Because the new system tray icons also display bitness, you can easily differentiate between LabVIEW 2014 (32-bit) and LabVIEW 2014 (64-bit) if both are installed.



Note Version- and bitness-specific system tray icons are available for only Windows and OS X operating systems.

[Idea submitted by NI Discussion Forums member Darren]

Installing and Updating DataPlugins in LabVIEW

In previous versions of LabVIEW, you must download and manually install DataPlugins from ni.com/dataplugins. In LabVIEW 2014, you can directly search, install, and update DataPlugins by using the Storage/DataPlugin VIs.

Block Diagram Enhancements

LabVIEW 2014 includes the following enhancements to the block diagram and related functionality.

Replacing a Tunnel with a Case Selector

In LabVIEW 2014, you can replace an input tunnel on a Case structure with the case selector for that structure. Right-click a tunnel and select **Replace with Case Selector** from the shortcut menu, and LabVIEW converts the tunnel to the case selector. Changing the input data to the case selector can change the allowed cases in the case selector label. If you replace a tunnel with the case selector, LabVIEW 2014 also converts the original case selector to a tunnel.

Replace with Case Selector

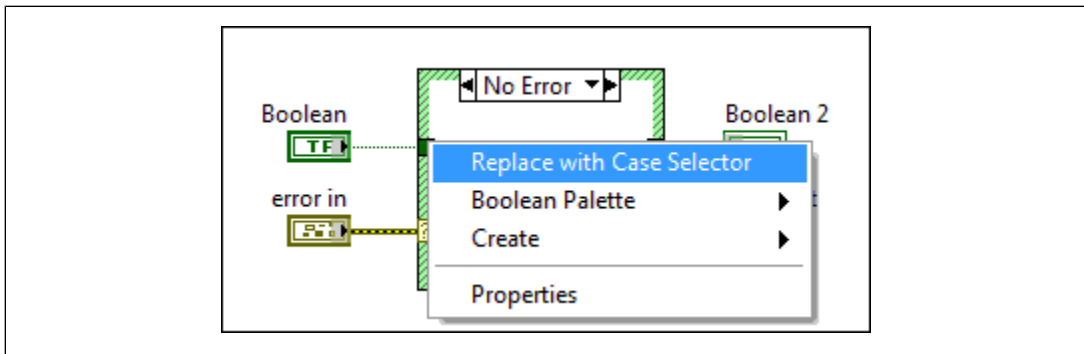


Figure 3.

After Replace with Case Selector

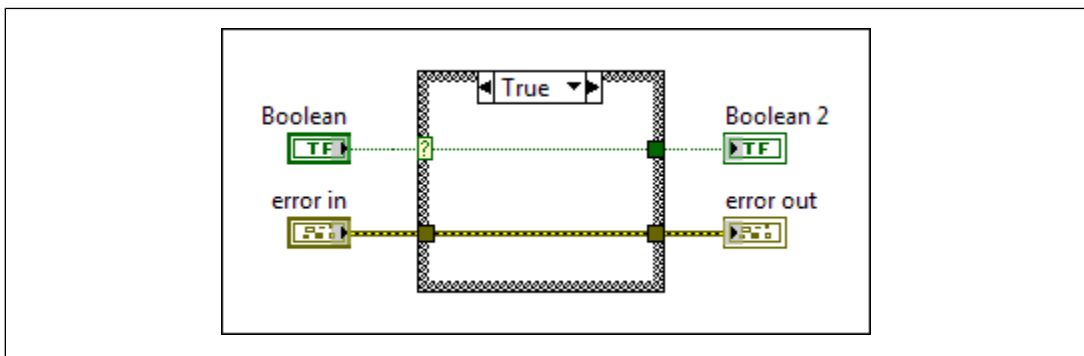


Figure 4.

You can also use the Replace With Case Selector method to convert a tunnel to the case selector programmatically.

[Idea submitted by NI Discussion Forums member NelsonUWP]

Automatically Wiring Objects in a VI

In LabVIEW 2014, you can use a **Quick Drop** keyboard shortcut to quickly wire multiple block diagram objects together. Highlight one or more parallel rows of objects and press <Ctrl-Space> to open the **Quick Drop** dialog box. Once the **Quick Drop** dialog box appears, press <Ctrl-W>. LabVIEW automatically wires the highlighted objects together. You also can press <Ctrl-Shift-W> to wire one or more parallel rows of block diagram objects together and clean up the highlighted code.

For more information about quick drop shortcuts, refer to **Fundamentals»LabVIEW Environment»How-To»Keyboard Shortcuts»Quick Drop Keyboard Shortcuts** on the **Contents** tab of the *LabVIEW Help*.

Front Panel Enhancements

LabVIEW 2014 includes the following enhancements to the front panel and related functionality.

Using Silver Style Controls

In LabVIEW 2014, use the new Decorations palette on the Silver palette to easily access silver style decorations.

Environment Enhancements

LabVIEW 2014 includes the following enhancements to the LabVIEW environment.

Troubleshoot Errors Caused by Missing SubVIs

In LabVIEW 2014, you can use the **Error List** window or the **Context Help** window to determine the location of missing subVIs. In previous versions of LabVIEW, the **Error List** window indicates when a subVI is missing but does not provide details about the location of the missing subVI. In LabVIEW 2014, the **Error List** window lists the driver, toolkit, or module that contains the missing subVI and provides information for resolving the error caused by the missing subVI. The **Context Help** window also lists the path to the missing subVI.

Reviewing and Updating Type Definition Instances

In LabVIEW 2013 and earlier, when you apply changes from a type definition to instances of that type definition, LabVIEW may lose or incorrectly preserve the default values for the instances you want to update.

In most cases, LabVIEW 2014 automatically preserves the default values of each instance you update from a type definition. When an instance cannot be automatically updated, LabVIEW places the instance in an unresolved state until you manually update using the **Review and Update from Type Def.** dialog box.

You can view the **Review and Update from Type Def.** dialog box by right-clicking an unresolved instance and selecting **Review and Update from Type Def.** The **Review and Update from Type Def** shortcut menu item replaces the **Update from Type Def** shortcut menu item that appears in LabVIEW 2013 and earlier.

Refer to the **Fundamentals»Building the Front Panel»How-To»Creating Custom Controls, Indicators, and Type Definitions** book on the **Contents** tab in the *LabVIEW Help* for more information about manually updating type definition instances.

Dialog Box Enhancements

LabVIEW 2014 includes the following dialog box enhancements.

Troubleshoot Error Codes from Error Dialog Boxes

In LabVIEW 2014, you can easily locate troubleshooting information for error codes from the **Explain Error** dialog box and the **Simple Error Handler VI** dialog box. Click the new **Search ni.com for error** hyperlink to display search results for the error on `ni.com` in the default web browser. By default, the hyperlink also appears in the **General Error Handler VI** dialog box. The hyperlink appears only in the development environment.



Figure 5.

New and Changed VIs, Functions, and Nodes

New VIs, Functions, and Nodes

LabVIEW 2014 includes the following new VIs, functions, and nodes.

Programmatically Get or Set Version Information for a Build Specification

The Application Builder palette includes the following new VIs:

- Get Build Specification Version
- Set Build Specification Version

Use these VIs to programmatically get or set the version information for any build specification with a version number.

Communicating with External Applications from LabVIEW for Mac OS X

LabVIEW 2014 for Mac OS X includes the following palettes with new VIs to help you communicate with OS X applications external to LabVIEW:

- The Libraries & Executables palette includes the Run AppleScript Code VI. This VI executes AppleScript code to communicate with external OS X applications from LabVIEW. In LabVIEW 2013 and earlier, you communicate with OS X applications external to LabVIEW with the Apple Event VIs. In LabVIEW 2014, you use only the Run AppleScript Code VI to communicate with OS X applications external to LabVIEW.
- The String palette includes the Normalize End Of Line VI. This VI converts the line endings of the string you specify to the line ending format you specify. If you do not specify a line ending format,

this VI converts the line endings of the string to the line endings that the command line of the current platform expects. Use this VI to make your strings readable on different platforms or the command line of the current platform.

- The Path/Array/String Conversion palette includes the following VIs:
 - **Path to Command Line String**—Converts the path into a string that describes that path. This VI formats the path string with the standard command line path format of the current platform. Use this VI to format paths before sending commands to the command line of the current platform.
 - **Command Line String to Path**—Converts the string you specify into a path. The input string must describe a path in the standard command line path format of the current platform. Use this VI to format paths you receive from the command line of the current platform for use with LabVIEW.

Actor Framework VIs

LabVIEW 2014 includes the following new VIs on the Actor Framework palette:

- **Actor:Launch Root Actor**—Launches an asynchronously running VI that performs tasks and handles messages for the Actor. This VI returns a reference to an enqueueer that you can use to send messages to the newly launched actor.
- **Actor:Launch Nested Actor**—Launches an asynchronously running VI that performs tasks and handles messages for the Nested Actor. Use this VI to launch nested actors. This VI returns a reference to the enqueueer that you can use to send messages to the newly launched actor.
- **Actor:Send Launch Nested Actor Msg**—Sends a message containing an actor to another actor. The actor receiving the message will launch the payload actor as a nested actor. Use this VI only to send a message from an actor to itself.

Changed VIs and Functions

The following VIs and functions changed in LabVIEW 2014

Storage/DataPlugin VIs

- **Open Data Storage**—The **Get More DataPlugins from ni.com/dataplugins** button in the configuration dialog box of this Express VI is replaced by the **Install/Update DataPlugins** button. Click **Install/Update DataPlugins** to launch the **Install/Update DataPlugins** dialog box, which allows you to search, install, and update DataPlugins from the National Instruments website at ni.com/dataplugins.
- **List DataPlugins**—This VI includes the new **source** input, which specifies whether LabVIEW lists DataPlugins from the local computer or the National Instruments website.
- **Register DataPlugins**—This VI includes the new **Install DataPlugin by Name** instance. You can use this instance to install DataPlugins to the local computer from ni.com/dataplugins.

Miscellaneous VI and Function Changes

LabVIEW 2014 includes the following miscellaneous VI and function changes:

- **Get Class Hierarchy from Class Name**—Returns an array of class names in descending order using a specified Class Name that inherits from Generic. For example, if Class Name is WhileLoop, this VI returns the following array:

```
[Generic, GObject, Node, Structure, Loop, WhileLoop]
```

- **High Resolution Relative Seconds**—Returns the value of the timer. This VI is similar to the Tick Count (ms) function, except that this VI provides a time stamp with a much higher resolution. You can use this VI to benchmark code with high precision.

- **Is Path and Not Empty?**—If **Path** is a value other than an empty path or <Not A Path>, this VI returns TRUE. Otherwise, this VI returns FALSE.
- **Variant Constant**—Use this constant to pass an empty variant to the block diagram. LabVIEW always drops an empty variant when you use this VI. LabVIEW does not allow you to edit the value of the variant.
- **Report Error Msg:Send Error Report**—Use this VI to send an error to an actor. The error will be handled by the Handle Error VI of the actor. If the error is not handled there, it will cause the actor to stop running.
- **TDMS Set Properties**—This function includes improvements to the `NI_MinimumBufferSize` property by enabling you to set the minimum buffer size at the group or file level for a `.tdms` file.
- **Clear Errors**—This VI includes the new **specific error code to clear** input, which ignores only the specific error code wired to this input. This VI also includes the new **specific error cleared?** output, which indicates whether the error referenced by **specific error code to clear** has been cleared.

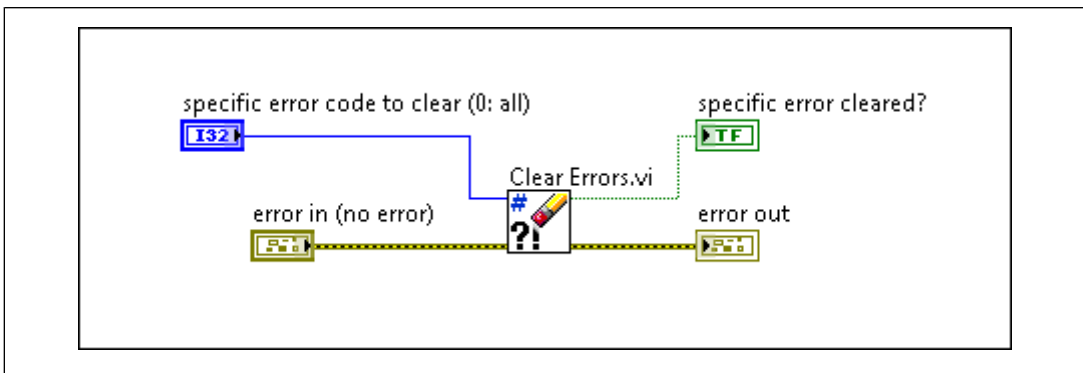


Figure 6.

Application Builder Enhancements

LabVIEW 2014 includes the following enhancements to the LabVIEW Application Builder and build specifications.

Deploying Installers to Windows Embedded Standard Targets

You can deploy a built installer to a target that runs the Windows Embedded Standard operating system. In the Project Explorer, under the Windows Embedded Standard target, right-click the build specification and select **Deploy** or **Install**.

Improvements to Loading Packed Project Libraries and Shared Libraries with the Same Library Version

To simplify the task of loading shared libraries from VIs or packed project libraries that share the same version, LabVIEW 2014 suppresses the **Load Warning Summary** dialog box that earlier versions of LabVIEW launch when you cross-link shared libraries or packed project libraries. Cross-linking occurs when you load a shared library from a VI or packed project library and then load another shared library with the same name from another VI or packed project library in a different location while the first library is still in memory. When loading the second shared library, LabVIEW links to the location of the first shared library. Shared libraries or packed project libraries must have the same version number to be the same version.

Excluding Dependent Packed Project and Shared Libraries from a Build Specification

In LabVIEW 2014, you can reduce the number of files LabVIEW copies when you create multiple build outputs that include the same libraries. You can specify to exclude dependent packed libraries and shared libraries from a build specification, and LabVIEW retains relative links to the source location of the excluded files. Otherwise, LabVIEW duplicates the packed project libraries and shared libraries in each subsequent output. To exclude these libraries, enable **Exclude dependent packed libraries** and **Exclude dependent shared libraries** on the **Additional Exclusions** page of the **Properties** dialog box for the build specification.

Setting the Destinations of Packed Project and Shared Libraries You Include with a Build Specification

In LabVIEW 2014, you can specify the destinations of dependent packed project libraries and shared libraries included as part of a build specification configuration. To specify the destinations of dependent files, select **Dependencies** on the **Source Files Settings** page of a build specification. Enable **Set destination for all contained items** and **Set destination for packed and shared libraries**, and select destinations from the drop-down menu.

Managing Compiled Code for a Source Distribution

In LabVIEW 2013 and earlier, you can use the **Remove compiled code** checkbox on the **Additional Exclusions** page of the **Properties** dialog box to reduce the size of a source distribution. LabVIEW 2014 provides the following additional options for managing compiled code:

- **Preserve compiled code**—Keeps the compiled code for all files.
- **Preserve file settings of each VI or library**—Maintains the stored settings for each file.

Refer to the **Facilitating Source Control by Separating Compiled Code from VIs and Other File Types** topic in the **Fundamentals»Working with Projects and Targets»Concepts»Using Source Control** book of the *LabVIEW Help* for more information about when to separate or keep compiled code for files.

Additional Functionality through Add-On Consolidation

The following add-on consolidations provide additional functionality for LabVIEW 2014:

- The LabVIEW 2014 Full and Professional Development Systems include all of the functionality of the LabVIEW PID and Fuzzy Logic Toolkit except the PID (FPGA) Express VI, which is part of the LabVIEW 2014 FPGA Module.
- LabVIEW 2014 Professional Development System now includes the following toolkits:
 - LabVIEW Database Connectivity Toolkit
 - LabVIEW Desktop Execution Trace Toolkit
 - LabVIEW Report Generation Toolkit
 - LabVIEW Unit Test Framework Toolkit
 - LabVIEW VI Analyzer Toolkit

The following toolkit consolidations also provide additional functionality:

- LabVIEW 2014 Digital Filter Design Toolkit includes the LabVIEW Adaptive Filter Toolkit.
- LabVIEW 2014 Control Design and Simulation Module and LabVIEW 2014 Advanced Signal Processing Toolkit include LabVIEW System Identification Toolkit.
- The LabVIEW 2014 FPGA Module includes the FPGA Compile Farm Toolkit, which is now known as the FPGA Compile Farm Server, and the FPGA IP Builder.
- The LabVIEW 2014 Real-Time Module includes the Real-Time Trace Viewer.

New LabVIEW Suites

To enable you to meet visualization, analysis, distribution, and software engineering needs and build systems with confidence, LabVIEW Suites contain LabVIEW Professional Edition in addition to our most popular application software and add-ons in one package. You can select from the following three LabVIEW suites built around different application areas:

- LabVIEW 2014 Automated Test Suite
- LabVIEW 2014 Embedded Control and Monitoring Suite
- LabVIEW 2014 HIL and Real-Time Test Suite

Visit ni.com/labview/suites for more information about the new LabVIEW Suites.

LabVIEW Web Services Enhancements

LabVIEW 2014 includes the following enhancements to LabVIEW Web services and related functionality.

Enhanced Security for LabVIEW Web Services

LabVIEW 2014 updates Web services to use the 1.0.1g release of the OpenSSL security protocol.

Providing Custom Documentation for VIs and Applications

In LabVIEW 2014, you can link a VI to web-based or local documentation from the **Context Help** window. **(Windows)** Use any file type for custom documentation except text files. **(OS X and Linux)** Use only HTML (.html and .htm) and PDF files for custom documentation.



Note (Linux) To use a PDF file for custom documentation, you must have one of the following PDF viewers installed:

- Acroread
- Xpdf
- KPDF
- GGv

In LabVIEW 2014, you also can provide custom help from the **Help** menu in a VI or application.

Linking VIs to Web-Based Documentation

To link a VI to web-based documentation from the **Context Help** window, select the **Web-based Help** option from the drop-down menu on the **Documentation** page of the **VI Properties** dialog box. Use the **Help URL** text box to specify the URL of the topic you want to link to a VI.

Linking VIs to Local Documentation

To link a VI to local documentation from the **Context Help** window, select the **Local Help File** option from the drop-down menu on the **Documentation** page of the **VI Properties** dialog box. Use the **Help path** text box to browse to custom documentation on the local computer. Optionally, use the **Help tag** text box to link a VI to an HTML topic within a compiled help file or to link a VI to a bookmark within an HTML help file. Otherwise, leave the **Help tag** text box empty.

For example, if you specify a path to a compiled help file named `My Custom Help.chm` in the **Help path** text box, you can enter `GettingStarted.html#Intro` in the **Help tag** text box so the **Detailed help** link and the **Detailed help** button in the **Context Help** window link directly to the introductory information you provide in the custom compiled help file.

Programmatically Linking a VI to Web-Based or Local Documentation

Use the Help:Use Web UR property in the Properties class to indicate you want to link a VI to a web-based help file. To specify the URL for the help file, use the Help:Document Web URL property in the Properties class.

You can link a VI to a local help file by wiring a VI Reference to the Document Path property and specifying the location of the help file on the local computer. Optionally, you can use the Help:Document Tag property to link to an individual HTML topic within a compiled help file or to a bookmark within an HTML file.

Providing Custom Help from the Help Menu

In LabVIEW 2014, you can provide custom help from the **Help** menu. (**Windows**) Use any file type for custom help except text files. (**OS X and Linux**) You can use only HTML (.html and .htm) and PDF files for custom help.



Note (Linux) To use a PDF file for custom documentation, you must have one of the following PDF viewers installed:

- Acroread
- Xpdf
- KPDF
- GGV

Complete the following steps to display custom help in the **Help** menu:

1. Place the custom help file in the `labview\help` directory.
2. Place a text file that has the same name as the custom help file in the `labview\help` directory. For example, if you name the help file `My Custom Help.html`, you must name the text file that contains the menu title `My Custom Help.txt`. Within the text file, include the help title you want to display in the **Help** menu.



Note The only information you must provide in the text file is the custom help title you want to display in the **Help** menu.

LabVIEW 2014 (64-bit) for Mac OS X

LabVIEW 2014 introduces a 64-bit version of the LabVIEW Development System for OS X. LabVIEW 2014 (64-bit) for Mac OS X includes virtually all of the development environment features of LabVIEW 2014 (32-bit) for Mac OS X. The following table describes the differences between LabVIEW 2014 (32-bit) and LabVIEW 2014 (64-bit) for Mac OS X.

To test the following compatibility issues in existing VIs, visit the National Instruments website at ni.com/info and enter the Info Code `analyzevi` to download automated tests.

Difference	LabVIEW 2014 (32-bit)	LabVIEW 2014 (64-bit)
Application Location	32-bit /Applications/National Instruments/LabVIEW 2014/LabVIEW.app	64-bit/Applications/National Instruments/LabVIEW 2014 64-bit/LabVIEW.app
Path String Format	Mac Classic (Macintosh HD:Users:johnd:Desktop)	POSIX (/Users/johnd/Desktop)
End of Line Constant value	CR (carriage return)	LF (linefeed)
CIN	Supported	Not Supported

Difference	LabVIEW 2014 (32-bit)	LabVIEW 2014 (64-bit)
Text Encoding	Mac Classic System Encoding	UTF-8
Print Preview	Non-native Print Preview functionality	Print Preview functionality from the Print dialog box



Note LabVIEW 2014 (64-bit) for Mac OS X supports a limited number of National Instruments drivers, modules, and toolkits. Refer to *National Instruments Product Compatibility for LabVIEW (64-bit) for Mac OS X* in the KnowledgeBase at ni.com/info for more information.

LabVIEW 2014 (64-bit) for Linux

LabVIEW 2014 introduces a 64-bit version of the LabVIEW Development System for Linux. LabVIEW 2014 (64-bit) for Linux supports all of the functionality of LabVIEW 2014 (32-bit) for Linux except the following features and add-ons:

- NI Instrument Driver Finder
- LabVIEW Control Design and Simulation Module
- LabVIEW VI Analyzer Toolkit

Changes to Touch Panel Functionality

Protecting Data from Modifications with Write Filter VIs

LabVIEW 2014 includes the Write Filter VIs on the Advanced File VIs and Functions palette. Write filters protect data from unwanted modifications by redirecting write operations to another location or overlay. Use enhanced write filters (EWF) to protect volumes and redirect writes to a disk location on another volume or to the RAM. Use file-based write filters (FBWF) to protect files and folders in a volume and redirect writes to the memory cache.

Previously, the Write Filter VIs required the LabVIEW Touch Panel Module. In LabVIEW 2014, the Write Filter VIs are available when you develop a touch panel application in the **Project Explorer** window. Support for touch panel applications requires the LabVIEW Application Builder, which is available with the LabVIEW Professional Development System.

Using the Touch Panel Project Template and Touch Panel VI Template

The Touch Panel Project template helps you create projects that target a touch panel device running the Windows Embedded Standard 7 operating system. The project template includes a VI template that you can adapt for your specific touch panel application. Select **File»Create Project** and browse to the Touch Panel Project template. Use the **Create Project** dialog box to configure project settings, including the touch panel target and VI template. Refer to the `Project Documentation` folder in the **Project Explorer** window for more information about how to modify the project.

You also can add a touch panel VI template to an existing touch panel target. LabVIEW includes portrait and landscape templates whose user interfaces are preset to the touch panel device. These templates include controls and indicators that you commonly use in touch panel applications. Right-click a touch panel target and select **New VI Template** to add a touch panel VI template to the target. The **Select a template** dialog box appears. Select the VI template to use with the target.

Working with Touch Panel Targets

LabVIEW 2014 includes support for developing, debugging, and deploying LabVIEW applications to a touch panel target that runs the Windows Embedded Standard 7 operating system. Previously, developing, debugging, and deploying touch panel applications required the LabVIEW Touch Panel

Module. With LabVIEW 2014, you develop and debug touch panel applications on the host computer, from which you can deploy the touch panel application to the touch panel target. Support for touch panel targets requires the LabVIEW Application Builder. The LabVIEW Professional Development System includes the Application Builder.

New and Changed Toolkits

LabVIEW Third Party Licensing and Activation Toolkit

With LabVIEW Third Party Licensing and Activation Toolkit 2014, you can provide the option for users to deactivate an add-on if the add-on is licensed with the toolkit. To deactivate an add-on, select **Help»Activate Add-ons** to open the Third Party Add-on Activation Wizard. Select the add-on, click **Deactivate**, and follow the steps in the wizard.

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 and enter the Info Code for the appropriate LabVIEW version from the following list:

- *LabVIEW 2010 Upgrade Notes*—[upnote10](#)
- *LabVIEW 2011 Upgrade Notes*—[upnote11](#)
- *LabVIEW 2012 Upgrade Notes*—[upnote12](#)
- *LabVIEW 2013 Upgrade Notes*—[upnote13](#)

Refer to the *NI Trademarks and Logo Guidelines* at 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. You can find information about end-user license agreements (EULAs) and third-party legal notices in the readme file for your NI product. Refer to the *Export Compliance Information* at 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. NI MAKES NO EXPRESS OR IMPLIED WARRANTIES AS TO THE ACCURACY OF THE INFORMATION CONTAINED HEREIN AND SHALL NOT BE LIABLE FOR ANY ERRORS. U.S. Government Customers: The data contained in this manual was developed at private expense and is subject to the applicable limited rights and restricted data rights as set forth in FAR 52.227-14, DFAR 252.227-7014, and DFAR 252.227-7015.