LabVIEW™ Upgrade Notes

These upgrade notes describe the process of upgrading LabVIEW for Windows, OS X, and Linux to LabVIEW 2015. 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 2015

Contents

Upgrading to LabVIEW 2015

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

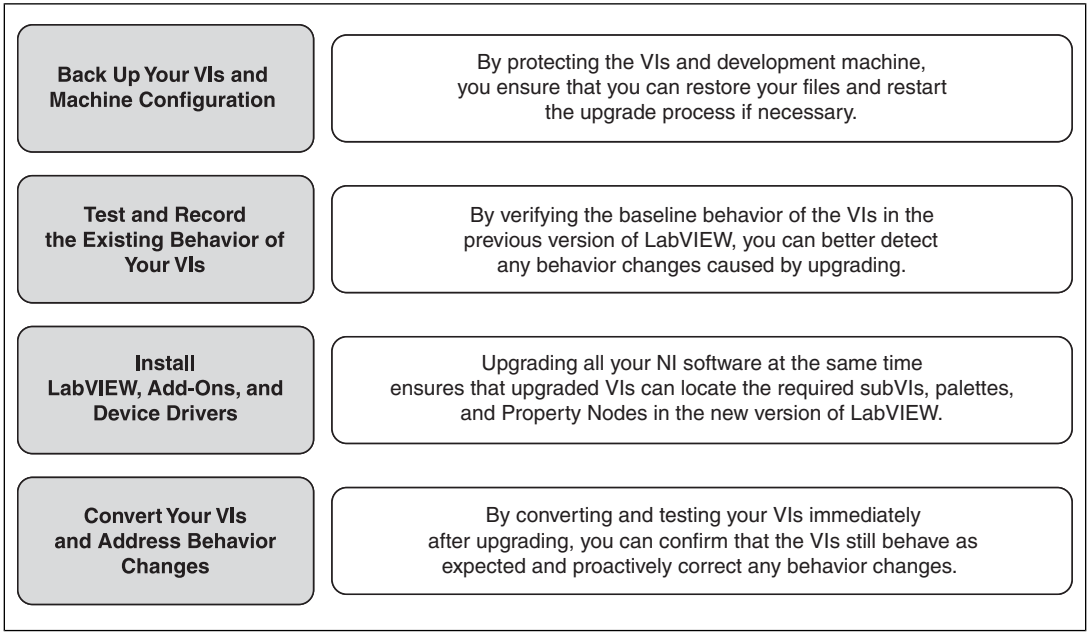

Figure 1.

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 2015, 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 2015 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:

- 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 2015, copy files from the `labview\user.lib` directory in the previous version to the `labview\user.lib` directory in LabVIEW 2015.

4. Convert Your VIs and Address Behavior Changes

Mass compiling your VIs in LabVIEW 2015 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 2015. 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.

Note When you mass compile VIs that contain FPGA or real-time resources, the **Mass Compile** dialog box may report the VIs as non-executable VIs. To check for errors, you must open the VIs under the FPGA or RT target in a LabVIEW project with the required FPGA or real-time resources.

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 2015 can occasionally cause some VIs to break if they use outdated features. To quickly identify and fix broken VIs in LabVIEW 2015, 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 2015 Known Issues list—Lists bugs discovered before and throughout the release of LabVIEW 2015. Refer to the National Instruments website at `ni.com/info` and enter the Info Code `lv2015ki` to access this list. Refer to the *Upgrade - Behavior Change* and *Upgrade - Migration* sections, if available, to identify workarounds for any bugs that may affect the behavior of upgraded VIs.
  - Module and toolkit documentation—Lists upgrade issues specific to some modules and toolkits, such as the LabVIEW FPGA Module and the LabVIEW Real-Time Module.
  - 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 2015.

- **Your own tests**—Perform the same tests on the VIs in LabVIEW 2015 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 2015»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 2015 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 2015.

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 2015 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 2015 from LabVIEW 2010. Refer to the *Upgrading from LabVIEW 2011*, *Upgrading from LabVIEW 2012*, *Upgrading from LabVIEW 2013*, and *Upgrading from LabVIEW 2014* 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 2015 from LabVIEW 2011. Refer to the *Upgrading from LabVIEW 2012*, *Upgrading from LabVIEW 2013*, and *Upgrading from LabVIEW 2014* 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 2015 from LabVIEW 2012. Refer to the *Upgrading from LabVIEW 2013* and *Upgrading from LabVIEW 2014* sections 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 *Web method VI* refers to the VIs that receive HTTP requests from clients and return data to clients.	The concept of Web method VIs is renamed *HTTP method VIs*.
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 *startup VIs*. 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.

Changes to the Queued Message Handler Template

The error handling scheme of the Queued Message Handler template changed in LabVIEW 2013. In the new error handling scheme, each loop handles errors using a loop-specific error handler subVI. If an error occurs in the Message Handling Loop, LabVIEW displays an error message.

Changes to the Continuous Measurement and Logging Sample Project

The error handling scheme of the Continuous Measurement and Logging sample project changed in LabVIEW 2013. In the new error handling scheme, each loop handles errors using a loop-specific error handler subVI. If an error occurs in the Message Handling Loop, LabVIEW displays an error message.

In LabVIEW 2013 and later, the Acquisition Message Loop and Logging Message Loop include state checking to handle cases where the loop receives a Start message when it has already started, or a Stop message when it has already stopped.

Upgrading from LabVIEW 2013

You might encounter the following compatibility issues when you upgrade to LabVIEW 2015 from LabVIEW 2013. Refer to the *Upgrading from LabVIEW 2014* section of this document for information about other upgrade issues you might encounter.

Behavior Change in the String to Path Function

In LabVIEW 2014 and later, 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 and later, 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 and later do not support the following VIs, functions, and nodes.

Apple Event VIs

(OS X) LabVIEW 2014 and later no longer support 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 and later do 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 and later do not support the In Port and Out Port VIs.

Deprecated Properties, Methods, and Events

LabVIEW 2014 and later do 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.

Upgrading from LabVIEW 2014

You might encounter the following compatibility issues when you upgrade to LabVIEW 2015 from LabVIEW 2014.

Identifying Buffer Allocations in LabVIEW Applications

LabVIEW 2014 Service Pack 1 and later include the **Profile Buffer Allocations** window to identify and analyze buffer allocations in a LabVIEW application. Select **Tools»Profile»Profile Buffer Allocations** to display this window.

Hyperlinks in Free Labels

LabVIEW 2015 detects URLs in free labels and converts them to hyperlinks underlined in blue text. LabVIEW does not automatically convert URLs in free labels to hyperlinks when you upgrade from LabVIEW 2014 or earlier. To enable hyperlinks in front panel labels, right-click the free label and select **Enable Hyperlinks** in the shortcut menu. You cannot enable or disable block diagram labels.

Deprecated VIs, Functions, and Nodes

LabVIEW 2015 does not support the following VIs, functions, and nodes.

- **Read From Spreadsheet File**—Use the Read Delimited Spreadsheet VI instead.
- **Write To Spreadsheet File**—Use the Write Delimited Spreadsheet VI instead.

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 2015.

Adding Custom Items to Shortcut Menus

You can add custom items to the shortcut menu of front panel and block diagram objects by creating shortcut menu plug-ins. You can create shortcut menu plug-ins that appear when you right-click edit-time front panel and block diagram objects or when you right-click run-time block diagram objects.

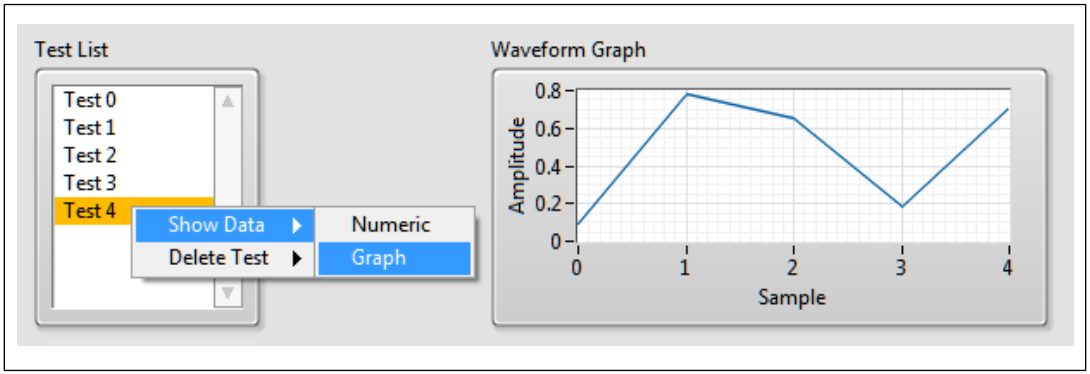

Figure 2.

To create a shortcut menu plug-in, open the following utility VI and follow the instructions on the VI front panel: `labview\resource\plugins\PopupMenus\Create Shortcut Menu Plug-in From Template.vi`

The utility VI generates the files you need for your plug-in. Customize the files to build the shortcut menu and execute the plug-in.

LabVIEW 2015 includes the following plug-ins.

- **Change To Array Or Element**—Converts a scalar value to an array of that type, or converts an array to a scalar of its element type. This plug-in affects controls, indicators, control and indicator terminals, and constants. This plug-in supports multi-object selection. *[Idea submitted by NI Discussion Forums member David_L.]*

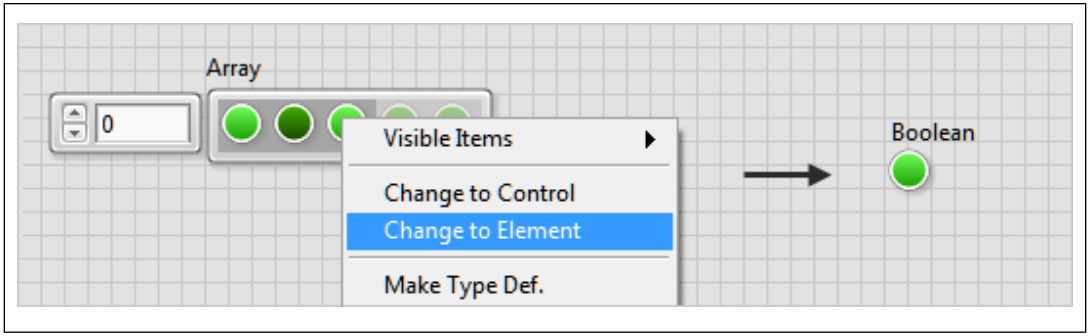

Figure 3.

- **Empty Listboxes**—Removes all rows from a listbox or multicolumn listbox. This plug-in affects listboxes and multicolumn listboxes on the front panel. This plug-in supports multi-object selection.
- **Explore**—Navigates to the file location on disk of a subVI, class, or typedef. This plug-in affects subVIs, class controls and indicators, class control and indicator terminals, class constants, typedef controls and indicators, typedef control and indicator terminals, and typedef constants.
- **Remove and Rewire Objects**—Removes the selected block diagram object and any wires and constants connected to the selected object, and connects wires of identical data types that were wired to the inputs and outputs of the deleted object. This plug-in affects any block diagram object that you can delete. This plug-in supports multi-object selection.
- **Size Array Constants To Contents**—Resizes the width of an array constant to match the width of the widest element in the array. This plug-in affects array constants. *[Idea submitted by NI Discussion Forums member blawson.]*
- **Transpose 2D Array**—Transposes the contents of a 2D array. This plug-in affects 2D array controls, indicators, and constants. This plug-in supports multi-object selection. *[Idea submitted by NI Discussion Forums member moderator1983.]*
- **Wire All Unwired Terminals**—Creates controls, indicators, or constants for all unwired inputs and outputs of the selected block diagram object. This plug-in affects any block diagram object you can wire. This plug-in supports multi-object selection.

Refer to the **Fundamentals»LabVIEW Environment»How-To»Understanding Shortcut Menu Plug-Ins** topic on the **Contents** tab in the *LabVIEW Help* for information about how to add custom items to shortcut menus.

Improvements to Adding or Reducing Space on the Front Panel or Block Diagram

LabVIEW 2015 includes usability improvements that make adding space easier and allow you to reduce space from the front panel or block diagram.

To increase the space between tightly grouped objects, press <Ctrl> and drag in the direction you want to add space. **(OS X)** Press <Option>. To reduce space between scattered objects, press <Ctrl-Alt> and drag in the direction you want to decrease the space. **(OS X)** Press <Option-Ctrl>. The objects move in real time as you drag the mouse. If the direction in which you drag is primarily vertical or horizontal, the operation snaps to the dominant direction.

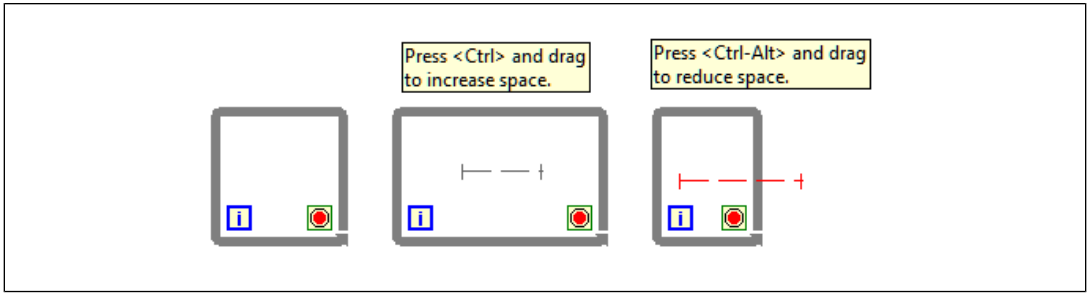

Figure 4.

Probe Enhancements

LabVIEW 2015 includes the following enhancements to probes.

- Most probe displays scale to fit the **Probe Display** subpanel of the **Probe Watch Window**.
- Generic probes for array data display multiple elements. Scrollbars display if elements do not fit the **Probe Display** subpanel.
- The default probe for string data is a custom probe. Right-click a wire and select **Custom Probe»Default String Probe** from the shortcut menu to use the default string probe. You can choose the string display types by clicking the gray bar on the left of the **Probe Display** subpanel.

Refer to the **Fundamentals»Grouping Data Using Strings, Arrays, and Clusters»Concepts»Grouping Data with Strings** topic on the **Contents** tab in the *LabVIEW Help* for information about string display types.

Hyperlinks in Free Labels

In LabVIEW 2015, LabVIEW detects URLs in free labels and converts them to hyperlinks underlined in blue text. You can click and open a hyperlink in the default web browser. LabVIEW 2015 enables hyperlinks by default. To disable hyperlinks in front panel labels, right-click the free label and deselect **Enable Hyperlinks** in the shortcut menu. You cannot disable hyperlinks in block diagram labels.

[Idea submitted by NI Discussion Forums member Rick L.]

Creating Actor Framework Actor and Message Classes

To create Actor Framework actor and message classes, you no longer need to load a project that uses the Actor Framework. You can use the new shortcut menu options in the **Project Explorer** window to create Actor Framework actor and message classes. The shortcut menu options in the **Project Explorer** window replace the **Actor Framework Message Maker** dialog box.

- To create an actor class, right-click a target in the **Project Explorer** window and select **New»Actor** from the shortcut menu.
- To create a message class, right-click a public method VI of an actor class and select **Actor Framework»Create Message** from the shortcut menu. You also can right-click multiple public method VIs and select **Actor Framework»Create Messages** from the shortcut menu to create a message class for each public method VI that you select.
- To create a message class for each public method VI of an actor class, right-click an actor class and select **Actor Framework»Create Messages for Actor** from the shortcut menu. You also can right-click multiple actor classes and select **Create Messages for Actors** from the shortcut menu to create a message class for each public method VI of the actor classes that you select.

- To create an abstract message class for an actor class, right-click an actor class and select **Actor Framework»Create Abstract Message for Caller** from the shortcut menu. The abstract message class defines only the message data but not the actor class to receive the message class.
  - You must create an abstract message class before you can create a child message class of the abstract message class. To create a child message class, right-click a public method VI of the actor class that will receive the abstract message class and select **Actor Framework»Create Child of Abstract Message**. The actor class that receives the abstract message class can use the new child message class to communicate with the actor class that sends the abstract message class. The actor class that sends the abstract message class does not have to know which actor class will receive the abstract message class and how the actor class will receive the message class.
- To rebuild an existing message class after the connector pane of the corresponding method VI changes, right-click a message class and select **Actor Framework»Rescript Message** from the shortcut menu.

Front Panel Enhancements

LabVIEW 2015 includes the following enhancements to the front panel.

Skipping Error In Clusters When Tabbing

In LabVIEW 2015, the new **error in** clusters by default include a checkmark in the **Skip this control when tabbing** option in the **Key Navigation** page of the **Properties** dialog box. When you press <Tab> on a running VI, LabVIEW skips the **error in** cluster controls. To include the **error in** clusters in the tabbing order, remove the checkmark from this option.

Note For **error in** clusters upgraded from previous versions of LabVIEW, you must manually enable this option to skip the controls when tabbing.

Environment Enhancements

LabVIEW 2015 includes the following enhancements to the LabVIEW environment.

Improvements to Compiler Optimizations

In LabVIEW 2015, compiler optimizations improve the execution performance of large VIs above the VI code complexity threshold. These improvements may slow the compile time. You can adjust the complexity threshold in the **Compiler** section on the **Environment** page of the **Options** dialog box. Changing the complexity threshold will continue to influence the compiler optimizations profile used when compiling VIs based on the VI code complexity relative to the threshold.

Refer to the **Fundamentals»Managing Performance and Memory»How-To»Choosing between Editor Responsiveness and VI Execution Speed** topic on the **Contents** tab in the *LabVIEW Help* for information about adjusting the complexity threshold.

Listing Missing Components After Loading VIs

When loading VIs, LabVIEW no longer prompts you to locate VIs from missing components, such as LabVIEW modules, toolkits, and drivers, and third-party add-ons. After LabVIEW loads the VIs, you can click **Show Details** in the **Load Warning Summary** dialog box or the **Save for Previous Warning Summary** dialog box or select **View»Load and Save Warning List** to display the **Load and Save Warning List** dialog box. The **Load and Save Warning List** dialog box includes the new **Missing Components** section, which lists the missing components that LabVIEW needs when loading the VIs.

Miscellaneous Environment Enhancements

LabVIEW 2015 includes the following miscellaneous environment enhancements:

- You can use the Error Ring in a subVI that is inlined into its calling VIs.
- LabVIEW 2015 includes an upgraded version of Math Kernel Library (MKL) 11.1.3 software for Windows and Linux. MKL is third-party software that LabVIEW uses to improve performance of linear algebra VIs. For more information regarding MKL, refer to the Intel Developer Zone website at `software.intel.com/en-us/intel-mkl`.

Miscellaneous Dialog Box Enhancements

LabVIEW 2015 includes the following miscellaneous dialog box enhancements:

- The **Additional Installers** page of the **Installer Properties** dialog box includes the new **Only display runtime installers** checkbox, which filters the run-time installers to display. Place a checkmark in this checkbox to view run-time installers only. This option is enabled by default.
- The **Find Project Items** dialog box includes the new **Export** button. Click this button to export search results to a text file.

New and Changed VIs and Functions

LabVIEW 2015 includes the following new and changed VIs and functions.

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

New VIs and Functions

LabVIEW 2015 includes the following new VIs and functions.

Advanced TDMS VIs and Functions

The Advanced TDMS palette includes the new TDMS In Memory subpalette, which you can use to open, close, read from, and write to `.tdms` files in memory. This subpalette includes the following functions:

- TDMS In Memory Close
- TDMS In Memory Open
- TDMS In Memory Read Bytes

The Advanced TDMS palette also includes the new TDMS Delete Data function. Use this function to delete data from a channel or multiple channels in a group.

Data Type Parsing VIs

The Variant palette includes the new Data Type Parsing subpalette, which includes the following VIs:

- Check for Contained Data Type
- Disconnect Type Definitions
- Get Array Information
- Get Cluster Information
- Get Fixed-Point Information
- Get LabVIEW Class Information
- Get Numeric Information
- Get Polymorphic VI Information
- Get Refnum Information
- Get Tag Information
- Get Type Definition Path
- Get Type Information

- Get User-Defined Refnum Information
- Get User-Defined Tag Information
- Get VI Information
- Get Waveform Information
- Is or Contains Type Definition

Use the Data Type Parsing VIs to retrieve the data type of a variant and information about the data type. You also can check whether the data type of a variant matches a specific data type.

Reading from and Writing to Delimited Spreadsheets

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

- **Read Delimited Spreadsheet**—Reads from delimited text files. This VI replaces the Read From Spreadsheet File VI.
- **Write Delimited Spreadsheet**—Converts data to a delimited text string and writes the string to a file. This VI replaces the Write To Spreadsheet File VI.

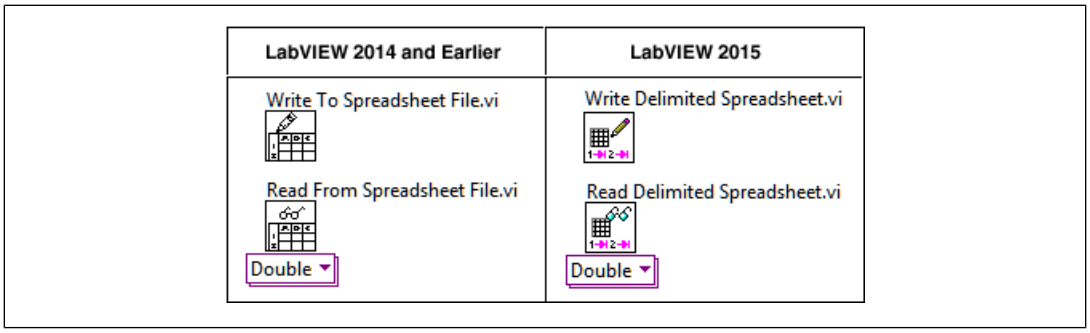

Figure 5.

Miscellaneous New VIs and Functions

LabVIEW 2015 includes the following miscellaneous new VIs and functions:

- The Discrete Math palette includes the new Check Prime VI. Use this VI to check whether a number is a prime number.
- The Advanced File palette includes the new Create File with Incrementing Suffix VI. Use this VI to create a file and append an incrementing number suffix to the filename if the file already exists at a specified path.
- The Dialog & User Interface palette includes the new Write to System Log VI. Use this VI to write error log messages to the `nierrlog system` log. You can view the messages using the system log viewer for your operating system. **(NI Linux Real-Time)** Open the **System Log Viewer** page of NI Web-based Configuration & Monitoring.

Miscellaneous VI and Function Changes

LabVIEW 2015 includes the following miscellaneous VI and function changes:

- **Get File Extension**—This VI includes the new **unmodified file extension** output, which returns the extension of the file you specify in **file**, without the period (**.**).

Application Builder Enhancements

LabVIEW 2015 includes the following enhancements to the LabVIEW Application Builder and build specifications.

(Windows) Building Shared Libraries with Type Libraries

LabVIEW 2015 provides an updated Application Builder that allows you to explicitly specify whether to embed a type library when you build shared libraries (DLLs). If you use TestStand or the LabVIEW Call Library Function Node, you must manually enable this option by placing a checkmark in the **Include a type library for TestStand or Call Library Nodes** checkbox on the **Advanced** page of the **Shared Library Properties** dialog box. The TestStand C/C++ DLL Adapter, LabWindows/CVI Adapter, and the LabVIEW Call Library Function Node use the type library to display a list of functions in the shared library, including the parameters and data types for the functions. You must install additional tools to embed a type library. Visit `ni.com/info` and enter the Info Code `DownloadMSDTBuildTools` to access the additional tools.

Improving Load Time for LabVIEW-Built Applications and Shared Libraries

You can build stand-alone applications (EXE) and DLLs that load faster by using the fast file format in LabVIEW.

To use the fast file format, place a checkmark in the **Use fast file format** checkbox on the **Advanced** page of the **Shared Library Properties** dialog box or the **Application Properties** dialog box. This option is disabled by default. When you enable the fast file format, LabVIEW does not use the **Application Builder** object cache. Therefore, stand-alone applications and shared libraries may take longer to build.

Note To use the fast file format, ensure that the **Enable debugging** checkbox on the **Advanced** page does not contain a checkmark.

New and Changed Classes, Properties, Methods, and Events

LabVIEW 2015 includes the following new and changed classes, properties, methods, and events.

VI Server Properties and Methods

LabVIEW 2015 includes the following new VI Server properties and methods:

- Enable Hyperlinks property (class: Text)—Reads or writes the setting that controls whether Text detects URLs in free labels and converts them to a hyperlink underlined in blue text.
- Disconnect Terminal method (class: Wire)—Disconnects a terminal from the wire without removing the loose end.
- Found Dependency Names property (class: GObject)—Reads an array of the qualified names of all external file dependencies that are loaded into memory of an object. For example, a control may have a dependency upon a `.ctl` or a `.xctl` file. If that dependency is in memory, its qualified name is included in the array.
- Found Dependency Paths property (class: GObject)—Reads an array of the paths of all the external file dependencies that are loaded into memory of an object. For example, a control may have a dependency upon a `.ctl` or a `.xctl` file. If that dependency is in memory, its path is included in the array.
- Missing Dependency Names property (class: GObject)—Reads an array of the qualified names of all the external file dependencies that are missing of an object. For example, a control may have a dependency upon a `.ctl` or a `.xctl` file. If that dependency is missing, its qualified name is included in the array.
- Missing Dependency Paths property (class: GObject)—Reads an array of the paths of all the external file dependencies that are missing of an object. For example, a control may have a dependency upon a `.ctl` or a `.xctl` file. If that dependency is missing, its path is included in the array.
- Missing VI Name property (class: SubVI)—Reads the qualified name of the VI that a subVI node calls if and only if that VI is missing. Otherwise, returns an empty string.

- Missing VI Path property (class: SubVI)—Reads the path of the VI that a subVI node calls if and only if that VI is missing. Otherwise, returns an empty string.
- Value (Undoable) property (class: Control)—Has the same effect as writing the Value property of a control except the scripting transaction system registers the write, which allows you to undo the value change. This property is write only.
- Default Value (Undoable) property (class: Control)—Has the same effect as writing the Default Value property of a control except the scripting transaction system registers the write, which allows you to undo the value change. This property is write only.

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 2011 Upgrade Notes*—`upnote11`
- *LabVIEW 2012 Upgrade Notes*—`upnote12`
- *LabVIEW 2013 Upgrade Notes*—`upnote13`
- *LabVIEW 2014 Upgrade Notes*—`upnote14`

371780L-01 Jun15