

LabVIEW™ Upgrade Notes

These upgrade notes describe the process of upgrading LabVIEW for Windows, OS X, and Linux to LabVIEW 2016. 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 2016

Contents

Upgrading to LabVIEW 2016.....	1
1. Back Up Your VIs and Machine Configuration.....	2
2. Test and Record the Existing Behavior of Your VIs.....	3
3. Install LabVIEW, Add-Ons, and Device Drivers.....	4
4. Convert Your VIs and Address Behavior Changes.....	4
Troubleshooting Common Upgrade Issues.....	5
Upgrade and Compatibility Issues.....	6
Upgrading from LabVIEW 2011 or Earlier.....	6
Upgrading from LabVIEW 2012.....	6
Upgrading from LabVIEW 2013.....	9
Upgrading from LabVIEW 2014.....	10
Upgrading from LabVIEW 2015.....	11
LabVIEW 2016 Features and Changes.....	11
Improvements to Selecting, Moving, and Resizing Objects.....	11
Asynchronously Communicating Data between Parallel Sections of Code.....	11
Environment Enhancements.....	12
New and Changed VIs and Functions.....	13
New and Changed Classes, Properties, Methods, and Events.....	13
Add-On Consolidation for LabVIEW (64-bit).....	14
Changes to LabVIEW for OS X.....	14
Changes to LabVIEW for Linux.....	14
Features and Changes in Previous Versions of LabVIEW.....	14

Upgrading to LabVIEW 2016

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

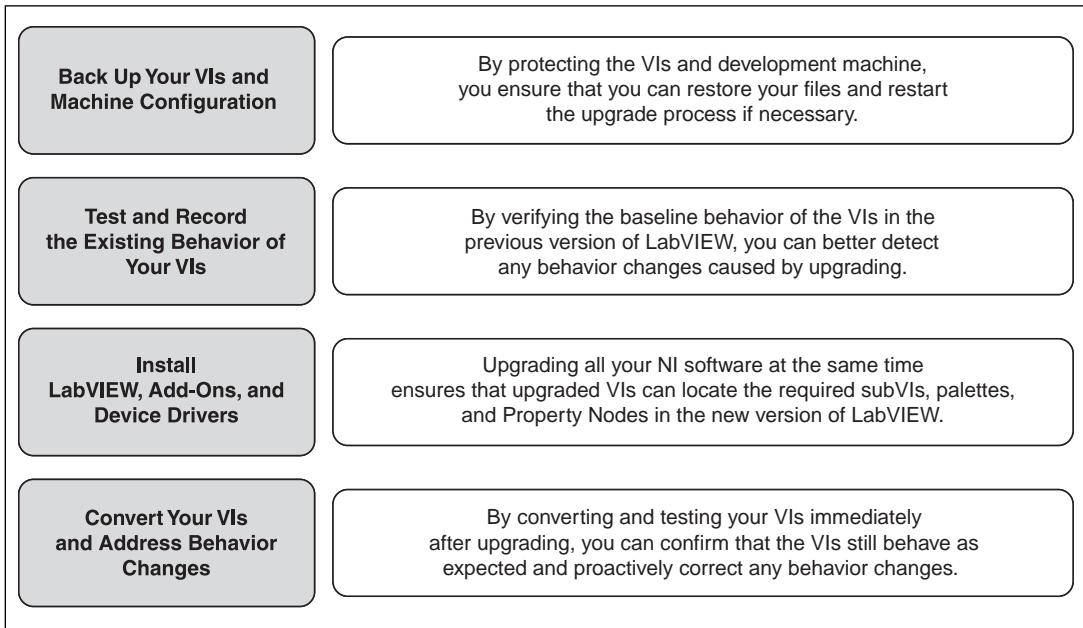


Figure 1.



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 2016, 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 2016 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 2016, copy files from the `labview\user.lib` directory in the previous version to the `labview\user.lib` directory in LabVIEW 2016.

4. Convert Your VIs and Address Behavior Changes

Mass compiling your VIs in LabVIEW 2016 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 2016. 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 2016 can occasionally cause some VIs to break if they use outdated features. To quickly identify and fix broken VIs in LabVIEW 2016, 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 2013 Upgrade VI Analyzer tests locate a potential behavior change, refer to the *Upgrading from LabVIEW 2012* 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 2016 Known Issues list—Lists bugs discovered before and throughout the release of LabVIEW 2016. Refer to the National Instruments website at ni.com/info and enter the Info Code `lv2016ki` 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 2016.
- **Your own tests**—Perform the same tests on the VIs in LabVIEW 2016 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 2016»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 2016 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 2016.

Upgrading from LabVIEW 2011 or Earlier

Refer to the National Instruments website at ni.com/info and enter the Info Code `upnote12` to access upgrade and compatibility issues you might encounter when you upgrade to LabVIEW 2016 from LabVIEW 2011 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 2012

You might encounter the following compatibility issues when you upgrade to LabVIEW 2016 from LabVIEW 2012. Refer to the *Upgrading from LabVIEW 2013*, *Upgrading from LabVIEW 2014*, and *Upgrading from LabVIEW 2015* 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 <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.

LabVIEW 2012 and Earlier	LabVIEW 2013
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.

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 2016 from LabVIEW 2013. Refer to the *Upgrading from LabVIEW 2014* and *Upgrading from LabVIEW 2015* sections 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?
- AECreat Comp Descriptor
- AECreat Descriptor List
- AECreat Logical Descriptor
- AECreat Object Specifier
- AECreat Range Descriptor
- AECreat 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 2016 from LabVIEW 2014. Refer to the *Upgrading from LabVIEW 2015* section of this document for information about other upgrade issues you might encounter.

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 and later detect 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 disable hyperlinks in block diagram labels.

Deprecated VIs, Functions, and Nodes

LabVIEW 2015 and later do 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.

Upgrading from LabVIEW 2015

You might encounter the following compatibility issue when you upgrade to LabVIEW 2016 from LabVIEW 2015.

In LabVIEW 2016 and later, the **Quick Drop Configuration** dialog box contains a default list of shortcuts for front panel and block diagram objects. Shortcuts you created in LabVIEW 2015 or earlier do not automatically migrate to the list of shortcuts in LabVIEW 2016 and later.

LabVIEW 2016 Features and Changes

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 2016.

Improvements to Selecting, Moving, and Resizing Objects

LabVIEW 2016 includes usability improvements to selecting objects, moving objects, and resizing structures on the front panel or block diagram.

- **Selecting Objects**
 - When you select objects, the area covered by the selection rectangle displays in gray and a marquee highlights the selected objects. Selected structures appear with a darker background to indicate your selection includes them.
 - By default, when you create a selection rectangle around objects, you must enclose the entire structure or the midpoint of a wire segment to select it. If you press the spacebar when you create the selection rectangle, you select any objects that the selection rectangle touches. To restore the default selection behavior, press the spacebar again.
- **Moving Objects**—When you move selected objects, the entire selection moves in real time. Certain objects, such as structures, rearrange or resize to accommodate the movement of the selected objects.
- **Resizing Structures**—When you resize a structure by dragging the resizing handles, the structure grows or shrinks in real time instead of displaying dashed borders.

Asynchronously Communicating Data between Parallel Sections of Code

In LabVIEW 2016, you can use channel wires to communicate data between parallel sections of code. Channel wires are asynchronous wires that connect two parallel sections of code without forcing an execution order, thus creating no data dependency between sections of code.

LabVIEW provides several channel templates, each of which expresses a different communications protocol. You can choose a template based on the communication requirements of your applications.

To create a channel wire, first create a writer endpoint by right-clicking a terminal or a wire and selecting **Create»Channel Writer**. Draw a channel wire from the channel terminal of the writer endpoint and create a reader endpoint by right-clicking the channel wire and selecting **Create»Channel Reader**.

The writer endpoint writes data into the channel and the reader endpoint reads data from the channel. Channel wires transfer data between sections of code the same way that refnums or variables do. However, channel wires use fewer nodes than refnums or variables, and represent the data transfer graphically using a visible wire.

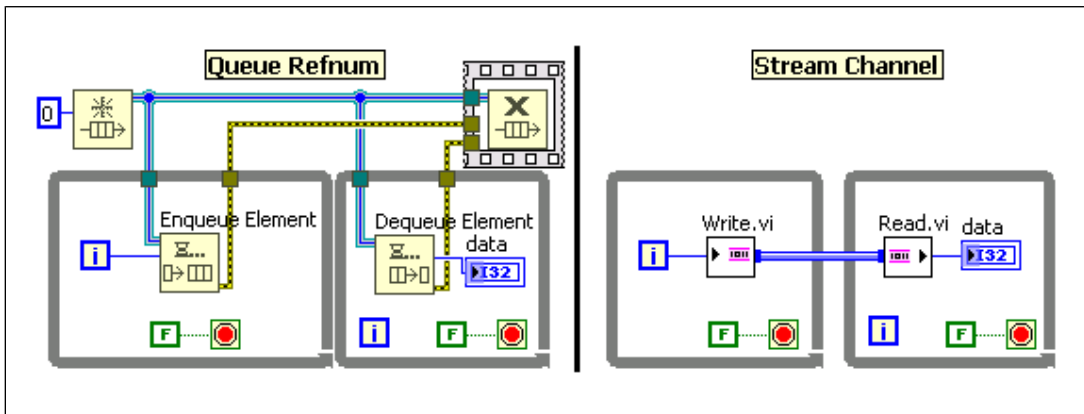


Figure 2.

Environment Enhancements

LabVIEW 2016 includes the following enhancements to the LabVIEW environment:

Dialog Box Enhancements

The **Quick Drop Configuration** dialog box contains a default list of shortcuts for front panel and block diagram objects. You can use the default shortcuts in the **Quick Drop** dialog box instead of configuring shortcuts manually.

The **Quick Drop Configuration** dialog box also adds the following options on the **Front Panel** page and the **Block Diagram** page:

- **Restore Default Panel Shortcuts/Restore Default Diagram Shortcuts**—Replaces the existing list of shortcuts with the default list of shortcuts.
- **Remove All Panel Shortcuts/Remove All Diagram Shortcuts**—Removes all shortcuts from the list.



Note After you click **Restore Default Panel Shortcuts/Restore Default Diagram Shortcuts** or **Remove All Panel Shortcuts/Remove All Diagram Shortcuts**, you must click **OK** to apply the changes. To revert your changes, click **Cancel**.

(Linux) Fonts and Encoding Support Enhancements

LabVIEW 2016 for Linux adds support for True Type fonts and UTF-8 character encoding.

- LabVIEW uses True Type fonts by default. You no longer need to install specific font packages that are required for LabVIEW bitmap fonts. To revert to bitmap fonts, add the

UseXftFonts=False configuration token to the LabVIEW configuration file under the following directory: /home/<username>/natinst/.config/<labview>/labview.conf.

- LabVIEW uses UTF-8 encoding if the locale environment of the Linux system that LabVIEW runs on is configured to use UTF-8 encoding. To disable UTF-8 encoding in LabVIEW, add the DisableUTF8=True configuration token to the LabVIEW configuration file under the following directory: /home/<username>/natinst/.config/<labview>/labview.conf.

New and Changed VIs and Functions

LabVIEW 2016 includes the following new and changed VIs and functions:

New VIs and Functions

LabVIEW 2016 includes the following new VIs and functions:

- The Advanced File palette includes the following new VIs:
 - **Is Name Multiplatform**—Use this VI to check the validity of a filename on operating systems supported by LabVIEW.
 - **Show in File System**—Use this VI to open a path to a file or directory in (**Windows**) Windows Explorer, (**OS X**) the Finder, or (**Linux**) a file system browser depending on the current platform.
- The Data Type Parsing palette includes the new Get Channel Information VI. Use this VI to retrieve the information about a channel and the transmission data type.
- The Actor Framework palette includes the new Read Autostop Nested Actor Count VI. Use this VI to return the number of nested actors that have not sent a Last Ack message to the calling actor. This VI counts only the nested actors that stop when the calling actor stops.
- The In Place Element structure includes the new Variant Attribute Get / Replace border node. Use this border node to access and modify one or more attributes of a variant without copying the attributes out of the variant. Refer to the Variant Attribute Lookup Table VI in the NI Example Finder or in the labview\examples\Performance\Variant Attribute Lookup Table directory for an example of using the Variant Attribute Get / Replace border node to create a high-performance lookup table.

Changed VIs and Functions

LabVIEW 2016 includes the following changed VIs and functions:

- (**Windows**) Similar to LabVIEW for OS X and LabVIEW for Linux, LabVIEW for Windows now has a finite limit of 1024 network sockets available to a single LabVIEW instance for all user-created and internal connections. This change impacts the Protocols VIs and functions for the TCP, UDP, Bluetooth, and IrDA protocols. Other protocols, such as Network Streams, Network Published Shared Variables, and Web Services, are not affected.
- There are new values for certain Math & Scientific Constants and Express Math & Scientific Constants. The values for Avogadro Constant, Elementary Charge, Gravitational Constant, Molar Gas Constant, Planck's Constant, and Rydberg Constant are updated to match values provided by CODATA 2014.

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 and Changed Classes, Properties, Methods, and Events

LabVIEW 2016 includes the following new and changed classes, properties, methods, and events:

VI Server Properties and Methods

LabVIEW 2016 includes the new Execution highlighting? (class: VI) property. Use this property to read or write the execution highlighting setting of a VI. You must enable VI Scripting to use this property. Unlike the Highlight Execution? (class: TopLevelDiagram) property, you can set the Execution highlighting? property for a VI that is a clone of a reentrant VI.

Add-On Consolidation for LabVIEW (64-bit)

LabVIEW 2016 (64-bit) now includes the LabVIEW Report Generation Toolkit for Microsoft Office.

New 64-bit Modules and Toolkit for Windows, OS X, and Linux

LabVIEW 2016 (64-bit) for all operating systems supports the new 64-bit versions of the following modules and toolkits:

- Control Design and Simulation Module—excludes System Identification VIs, System Identification Assistant, and Control Design Assistant
- MathScript RT Module
- VI Analyzer Toolkit
- **(Windows)** Desktop Execution Trace Toolkit
- **(Windows)** Unit Test Framework Toolkit

Refer to the readme of each product for more information, including system requirements, installation instructions, and activation.

Changes to LabVIEW for OS X

LabVIEW 2016 for OS X is available only in 64-bit. NI no longer provides the 32-bit version of LabVIEW for OS X.

Changes to LabVIEW for Linux

LabVIEW 2016 (32-bit and 64-bit) for Linux supports only 64-bit versions of Linux operating systems. LabVIEW for Linux no longer supports 32-bit versions of Linux operating systems.

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 2012 Upgrade Notes*—[upnote12](#)
- *LabVIEW 2013 Upgrade Notes*—[upnote13](#)
- *LabVIEW 2014 Upgrade Notes*—[upnote14](#)
- *LabVIEW 2015 Upgrade Notes*—[upnote15](#)

Refer to the *NI Trademarks and Logo Guidelines* at ni.com/trademarks for more information on NI trademarks. Other product and company names mentioned herein are trademarks or trade names of their respective companies. For patents covering NI 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 NI 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.