LabVIEW™ 2018 Upgrade Notes

These upgrade notes describe the process of upgrading LabVIEW for Windows, macOS, and Linux to LabVIEW 2018. 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 2018

Contents

Upgrading to LabVIEW 2018....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....6

Upgrade and Compatibility Issues....6
- Upgrading from LabVIEW 2013 or Earlier....6
- Upgrading from LabVIEW 2014....6
- Upgrading from LabVIEW 2015....7
- Upgrading from LabVIEW 2016....7
- Upgrading from LabVIEW 2017....7

LabVIEW 2018 Features and Changes....8
- Customizing a Malleable VI for Different Data Types....8
- Running Operations Using the Command Line Interface for LabVIEW....8
- Calling Python Code from LabVIEW....9
- Application Builder Enhancements....9
- Environment Enhancements....10
- Block Diagram Enhancements....10
- Front Panel Enhancements....11
- New VIs and Functions....11
- New and Changed Properties and Methods....12

Features and Changes in Previous Versions of LabVIEW....13

Upgrading to LabVIEW 2018

Although you can upgrade small applications to a new version of LabVIEW by installing the new version and then loading your VIs, NI 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 NI website at `ni.com/info` and enter the Info Code `lifecycle` for information about which versions of LabVIEW still receive mainstream support.

NATIONAL INSTRUMENTS™

Overview of the Recommended Upgrade Process

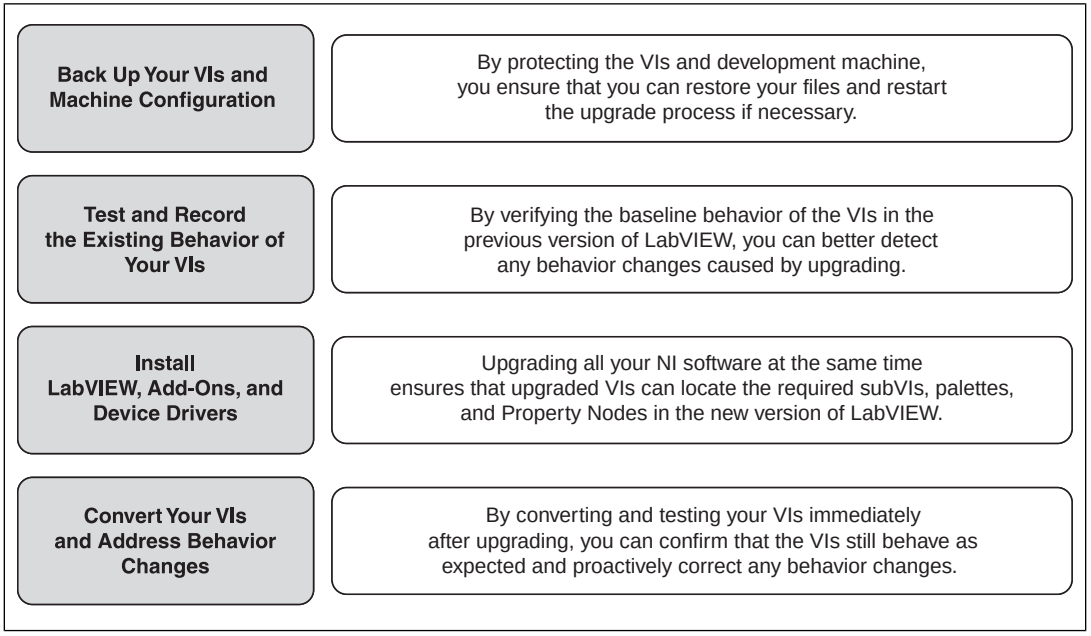

Figure 1.

Note To upgrade from LabVIEW 5.1 or earlier, you must first upgrade to an intermediate version of LabVIEW. Refer to the NI 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 2018, 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, NI 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, differences between the previous version of LabVIEW and LabVIEW 2018 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.

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 2018, copy files from the `labview\user.lib` directory in the previous version to the `labview\user.lib` directory in LabVIEW 2018.

4. Convert Your VIs and Address Behavior Changes

Mass compiling your VIs in LabVIEW 2018 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.

Note If you mass compiled your VIs when you upgraded to LabVIEW 2017, you can skip this step because LabVIEW 2017 and later supports backward compatibility for the LabVIEW Run-Time Engine.

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

Mass compiling VIs simultaneously converts and saves the VIs in LabVIEW 2018. 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

Differences between your previous version of LabVIEW and LabVIEW 2018 can occasionally cause some VIs to break if they use modified features. To quickly identify and fix broken VIs in LabVIEW 2018, 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 NI 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:

- **Run the VI Analyzer Upgrade 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 VI Analyzer Upgrade Tests for all versions of LabVIEW later than your previous version. Refer to the NI 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 2015 VI Analyzer Upgrade tests locate a potential behavior change, refer to the *Upgrading from LabVIEW 2014* section of that topic.
- **Read the 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 2018 Known Issues list—Lists bugs discovered before and throughout the release of LabVIEW 2018. Refer to the NI website at `ni.com/info` and enter the Info Code `lv2018ki` 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 2018.

- **Run your own tests**—Perform the same tests on the VIs in LabVIEW 2018 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 2018»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 2018 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 2018.

Upgrading from LabVIEW 2013 or Earlier

Refer to the NI website at `ni.com/info` and enter the Info Code `upnote14` to access upgrade and compatibility issues you might encounter when you upgrade to LabVIEW 2018 from LabVIEW 2013 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 2014

You might encounter the following compatibility issues when you upgrade to LabVIEW 2018 from LabVIEW 2014. Refer to the *Upgrading from LabVIEW 2015*, *Upgrading from LabVIEW 2016*, and *Upgrading from LabVIEW 2017* sections 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 2018 from LabVIEW 2015. Refer to the *Upgrading from LabVIEW 2016* and *Upgrading from LabVIEW 2017* sections of this document for information about other upgrade issues you might encounter.

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.

Upgrading from LabVIEW 2016

You might encounter the following compatibility issue when you upgrade to LabVIEW 2018 from LabVIEW 2016. Refer to the *Upgrading from LabVIEW 2017* section of this document for information about other upgrade issues you might encounter.

Behavior Change in the Actor Framework VIs

In LabVIEW 2016 and earlier, when a nested actor fails to launch because of an error in the Pre-Launch Init method, the nested actor returns an error and sends a Last Ack message that contains the error to its caller actor. In LabVIEW 2017, the nested actor returns an error without sending a Last Ack message to its caller actor.

Upgrading from LabVIEW 2017

You might encounter the following compatibility issue when you upgrade to LabVIEW 2018 from LabVIEW 2017.

Backward Compatibility of the LabVIEW Run-Time Engine

LabVIEW 2017 and later supports backward compatibility for the LabVIEW Run-Time Engine. You can load and run binaries and VIs built in older versions of LabVIEW without mass recompiling VIs or rebuilding binaries in the current version of LabVIEW. For example, versions of LabVIEW later than 2017 can load binaries and VIs built with LabVIEW 2017 without recompiling. This improvement applies to stand-alone applications (EXEs), shared libraries (DLLs), and packed project libraries.

To enable binaries to be backward compatible, place a checkmark in the following checkbox on the **Advanced** page of the specific dialog box depending on your build specification:

Build Specification	Dialog Box	Checkbox
Stand-alone application (EXE)	**Application Properties**	**Allow future versions of the LabVIEW Runtime to run this application**
Packed project library	**Packed Library Properties**	**Allow future versions of LabVIEW to load this packed library**
Shared library (DLL)	**Shared Library Properties**	**Allow future versions of LabVIEW to load this shared library**

LabVIEW enables these options by default for build specifications you create in LabVIEW 2017 and later. You can disable these options to bind a build specification to a specific version of LabVIEW. Disabling these options prevents any changes to the performance profiles and helps you avoid unexpected problems resulting from compiler upgrades. For real-time applications, these options do not appear in the dialog boxes but the functionality is enabled by default.

Behavior Changes to the Report Generation VIs

In LabVIEW 2018, the Report Generation VIs no longer support generating reports in the Standard Report format. You can generate reports in HTML, Word, or Excel formats only. Because of the behavior change, the following VIs are deprecated:

- **Easy Print VI Panel or Documentation**—This VI is deprecated. Use the **Print VI Panel or Documentation** VI instead.
- **Easy Text Report**—This VI is deprecated. Use the **Create Easy Text Report** VI instead.
- **Get Report Type**—This VI is deprecated. Use the **Report Type** VI instead.
- **New Report**—This VI is deprecated. Use the **Create Report** VI instead.
- **Set Report Tab Width**—This VI is deprecated.

Deprecated VIs, Functions, and Nodes

LabVIEW 2018 and later do not support the **Number To Enum** VI. Use the Coerce To Type function instead.

LabVIEW 2018 Features and Changes

The Idea Exchange icon denotes a new feature idea that originates from a product feedback suggestion on the NI Idea Exchange discussion forums. Refer to the NI 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 2018.

Customizing a Malleable VI for Different Data Types

The Comparison palette includes the new Assert Type subpalette. Use the Assert Type VIs and function to force a malleable VI (`.vim`) to accept only data types that meet certain requirements. Use the Type Specialization structure to customize sections of code in a malleable VI for specific data types.

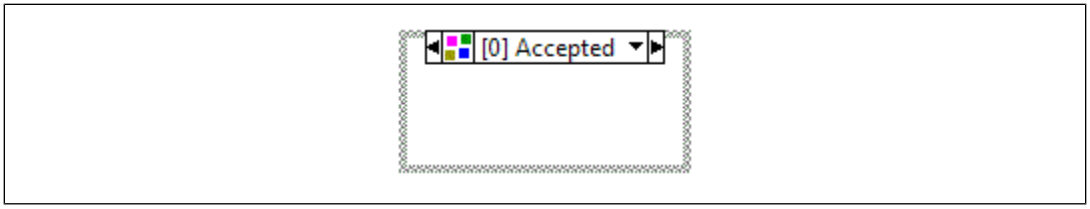

Figure 2.

Refer to the `labview\examples\Malleable VIs\Type Specialization Structure\Malleable VIs - Type Specialization Structure.lvproj` for an example of using the Type Specialization structure to customize sections of code in a malleable VI.

Running Operations Using the Command Line Interface for LabVIEW

LabVIEW 2018 allows you to run operations in LabVIEW by executing commands using the command line interface (CLI) for LabVIEW. For example, use the CLI for LabVIEW to automate the build process of LabVIEW applications. The CLI for LabVIEW supports the following operations:

- MassCompile—Mass compiles the files in the specified directory.
- ExecuteBuildSpec—Builds an application, a library, or a bitfile using the settings in the specified build specification(s) and returns the path of the output files.

- RunVI—Runs a VI with the predefined connector pane interface and returns the output or error information.
- CloseLabVIEW—Closes LabVIEW without any prompts.
- **(VI Analyzer Toolkit)** RunVIAnalyzer—Runs the specified VI analyzer task in the LabVIEW VI Analyzer Toolkit and saves the test report to the specified location.
- **(Unit Test Framework Toolkit)** RunUnitTests—Runs tests on the specified files in the LabVIEW Unit Test Framework Toolkit and saves the JUnit file to the specified location.

Note To run this operation in LabVIEW, you must install the UTF Junit Report library using the JKI VI Package Manager (VIPM) software. Refer to the **Toolkits»Accessing LabVIEW Add-ons with the VI Package Manager (VIPM) Software** topic on the **Contents** tab in the *LabVIEW Help* for more information about using the VIPM software.

You can also create custom operations to run in LabVIEW.

Refer to the **Fundamentals»Running Operations Using the Command Line Interface for LabVIEW** book on the **Contents** tab in the *LabVIEW Help* for more information about using the CLI for LabVIEW.

Calling Python Code from LabVIEW

The Connectivity palette includes the new Python subpalette, which you can use to call Python code from LabVIEW code. The Python palette includes the following functions:

- **Open Python Session**—Opens a Python session with a specific version of Python.
- **Python Node**—Calls a Python function directly.
- **Close Python Session**—Closes a Python session.

Note You must install Python 2.7 or 3.6 to use the LabVIEW Python functions. Although unsupported versions might work with the LabVIEW Python functions, NI recommends using supported versions of Python only. Visit `ni.com/info` and enter the Info Code `python` to learn more about installing Python.

Application Builder Enhancements

LabVIEW 2018 includes the following enhancements to the LabVIEW Application Builder and build specifications.

Creating Packages on Windows and Linux Real-Time Targets

You can create packages in LabVIEW and deploy them to clients through NI Package Manager or SystemLink. You can use packages with Package Manager and SystemLink to distribute all types of files, including source distributions, packed project libraries, shared libraries, .NET assemblies, and executables.

(Windows 64-bit) Create NI packages (`.nipkg`) by right-clicking **Build Specifications** in the **Project Explorer** window and selecting **New»Package**. Your clients can use Package Manager or SystemLink to subscribe to a feed to find and install your packages.

(NI Linux Real-Time) You can also create opkg packages (`.ipk`) on NI Linux Real-Time targets if you install the LabVIEW Real-Time Module. Your clients can install packages through SystemLink or from the command line on the NI Linux Real-Time target. Package Manager does not support `.ipk` files.

Refer to the **Fundamentals»Building and Distributing Applications»Creating Build Specifications»Creating Packages for Distribution** topic on the **Contents** tab in the *LabVIEW Help* for more information about creating packages.

Backward Compatibility Support for LabVIEW-Built .NET Assemblies

With support for backward compatibility, .NET interop assemblies can load either in the LabVIEW version that they are built with or in the latest version of the LabVIEW Run-Time Engine installed on the machine. For example, you can load and run .NET interop assemblies, which are built with LabVIEW 2018, in versions of the LabVIEW Run-Time Engine later than 2018 without recompiling.

To enable backward compatibility support for .NET assemblies, place a checkmark in the **Allow future versions of LabVIEW to load this .NET assembly** checkbox on the **Advanced** page of the **.NET Interop Assembly Properties** dialog box.

LabVIEW enables this option by default for build specifications you create in LabVIEW 2018 and later. You can disable this option to bind a build specification to a specific version of LabVIEW. Disabling this option prevents any changes to the performance profiles and helps you avoid unexpected problems resulting from compiler upgrades. For real-time applications, this option does not appear in the dialog box but the functionality is enabled by default.

Environment Enhancements

LabVIEW 2018 includes the following enhancements to the LabVIEW environment:

Improvements to Creating Type Definitions

In LabVIEW 2018, you have more ways of creating a type definition, which links all the instances of a custom control or indicator to a saved custom control or indicator file. You can create a new type definition in one of the following ways:

- Select **File»New** and select **Type Definition** under **Other Files**.
- Right-click **My Computer** in the **Project Explorer** window and select **New»Type Definition** from the shortcut menu.

[Idea submitted by NI Discussion Forums member Mathis_B.]

Keyboard Shortcuts for Formatting Text

Use the following keyboard shortcuts to format the font style when editing text in the LabVIEW environment:

- <Ctrl-B>—Bolds text.
- <Ctrl-I>—Italicizes text.
- <Ctrl-U>—Underlines text.

[Idea submitted by NI Discussion Forums member vt92.]

Block Diagram Enhancements

LabVIEW 2018 includes the following enhancements to the block diagram and related functionality:

Improvements to Error Handling on Parallel For Loops

LabVIEW 2018 introduces error registers to simplify error handling on a For Loop with parallel iterations enabled. Error registers take the place of shift registers for error clusters on a parallel For Loop, as shown in the following block diagram.

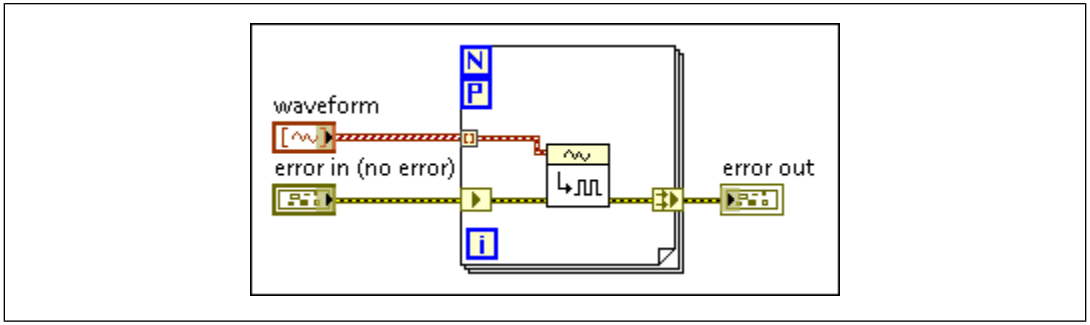

Figure 3.

Error registers automatically merge errors from parallel iterations. LabVIEW preserves the best practice of flowing errors through a shift register by automatically converting shift registers to error registers when you configure parallel iterations on a For Loop.

Error registers and shift registers differ in their run-time behavior. The left side terminal of the error register behaves like a non-indexing input tunnel and produces the same value on every iteration. The right side terminal of the error register merges the values of each iteration such that the error or warning value from the earliest iteration, by index, is the output value of the error register. If the For Loop iterates zero times, the value you wire into the left side tunnel carries forward to the output on the right side tunnel.

Improvements to Removing and Rewiring Objects

When you remove and rewire a selection of block diagram objects, LabVIEW also removes any decorations, including free labels, that are in the selection rectangle. Remove and rewire objects by dragging a selection rectangle around block diagram objects, right-clicking the selection, and selecting **Remove and Rewire**. You can also use the **Quick Drop** keyboard shortcuts <Ctrl-Space> and <Ctrl-R> keys after selecting objects to remove and rewire objects.

Front Panel Enhancements

NXG Style Controls and Indicators

The Controls palette includes the new **NXG Style** of front panel controls and indicators. Use the NXG style controls and indicators to create front panels with the same style as LabVIEW NXG. The appearance of these controls and indicators changes depending on the platform on which end users run the VI. Using these controls and indicators minimizes distortion of your front panels if you migrate the VIs to LabVIEW NXG.

New VIs and Functions

LabVIEW 2018 includes the following new VIs and functions:

- The Comparison palette includes the new Assert Type subpalette, which includes the following VIs and function:
  - Assert Array Dimension Count
  - Assert Array Dimension Sizes
  - Assert Complex Numeric Type
  - Assert Error Cluster Type
  - Assert Fixed-Point Numeric Type
  - Assert Floating-Point Numeric Type
  - Assert Fractional Numeric Type

  - Assert Integer Type
  - Assert Real Floating-Point Numeric Type
  - Assert Real Numeric or Waveform Type
  - Assert Real Numeric Type
  - Assert Same or Descendant Type
  - Assert Scalar Numeric or Waveform Type
  - Assert Scalar Numeric Type
  - Assert Signed Integer Type
  - Assert Structural Type Match
  - Assert Unsigned Integer Type
  - Type Specialization Structure
- The Connectivity palette includes the new Python subpalette, which includes the following functions:
  - Open Python Session
  - Python Node
  - Close Python Session
- The Conversion palette includes the new Coerce To Type function. Use this function to convert the input data to a compatible data type while preserving the data value. Unlike the Type Cast function, this function does not reinterpret the input data. Use this function in the following cases:
  - To eliminate a coercion dot
  - To convert data without a type definition to a compatible type definition or vice versa
  - To rename data on the wire, such as a user event refnum

  [Idea submitted by NI Discussion Forums member JackDunaway.]
- The Timing palette includes the new High Resolution Polling Wait VI. Use this VI to wait the specified number of seconds with higher resolution than you can obtain with the Wait (ms) function.

New and Changed Properties and Methods

LabVIEW 2018 includes the following new and changed properties and methods:

- The LeftShiftRegister class includes the new Is An Error Register property. Use this property to read whether a shift register is an error register. An error register is a special form of shift register that exists on a For Loop with parallel iterations enabled and when the data type of the shift register is an error cluster.
- The VI class includes the new Configure Panel As Top-Level Hidden method. Use this method to hide the front panel of a VI and optionally to hide the VI from the taskbar when the VI runs as a top-level VI. For example, use this method to hide the front panel of startup VIs of stand-alone applications that you build in LabVIEW.
- The DisableStructure class includes the new Disable Style property. Use this property to read whether the structure is a Diagram Disable structure, a Conditional Disable structure, or a Type Specialization structure.
- The **Disable Style** parameter of the Change Disable Style (class: DisableStructure) method includes the new **Type Specialization Style** option. Use this option to change a Diagram Disable structure or a Conditional Disable structure to a Type Specialization structure.

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 NI website at `ni.com/info` and enter the Info Code for the appropriate LabVIEW version from the following list:

- *LabVIEW 2014 Upgrade Notes*—`upnote14`
- *LabVIEW 2015 Upgrade Notes*—`upnote15`
- *LabVIEW 2016 Upgrade Notes*—`upnote16`
- *LabVIEW 2017 Upgrade Notes*—`upnote17`

371780P-01 Mar18