UPGRADE NOTES

LabVIEW™ 2019 Upgrade Notes

These upgrade notes describe the process of upgrading LabVIEW for Windows, macOS, and Linux to LabVIEW 2019. 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 2019

Contents

Upgrading to LabVIEW 2019

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 downtime; 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.

Overview of the Recommended Upgrade Process

Back Up Your VIs and Machine Configuration	By protecting the VIs and development machine, you ensure that you can restore your files and restart the upgrade process if necessary.
Test and Record the Existing Behavior of Your VIs	By verifying the baseline behavior of the VIs in the previous version of LabVIEW, you can better detect any behavior changes caused by upgrading.
Install LabVIEW, Add-Ons, and Device Drivers	Upgrading all your NI software at the same time ensures that upgraded VIs can locate the required subVIs, palettes, and Property Nodes in the new version of LabVIEW.
Convert Your VIs and Address Behavior Changes	By converting and testing your VIs immediately after upgrading, you can confirm that the VIs still behave as expected and proactively correct any behavior changes.

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 2019, 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 backup copy. Without the backup 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 by submitting 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 in the *LabVIEW Help*.

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 downtime for your applications would be costly:

- Create a backup 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 backup disk image.
- Test the upgrade process on a test machine—Use a test machine, commonly a virtual machine, to test the upgrade process. Although upgrading on a test machine requires more time than creating a backup image, NI strongly recommends this approach if you need to prevent or minimize downtime 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 2019 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 that identify the existence of broken VIs

Mass compiling your VIs before you upgrade 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 in 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 the VIs.
- Stress tests that verify whether the VIs handle unexpected data correctly

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

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 in the *LabVIEW Help*.

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 2019, copy files from the `labview\user.lib` directory in the previous version to the `labview\user.lib` directory in LabVIEW 2019.

c. Reinstall VI Packages

If you used the JKI VI Package Manager (VIPM) to install VI packages in the previous version of LabVIEW, go to the VIPM software and reinstall all packages to LabVIEW 2019 as well.

4. Convert Your VIs and Address Behavior Changes

Mass compiling your VIs in LabVIEW 2019 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 *Upgrade and Compatibility Issues* to identify and correct behavior changes associated with the new version of LabVIEW.

Note NI recommends that you use source control in LabVIEW to back up VIs and track changes. This allows you to revert to a previous version of the VIs if you cannot address behavior changes caused by upgrading the VIs.

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

Mass compiling VIs simultaneously converts and saves the VIs in LabVIEW 2019. 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 in 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 2019 can occasionally cause some VIs to break if they use modified features. To quickly identify and fix broken VIs in LabVIEW 2019, 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 *Upgrade and Compatibility Issues*.

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 *Upgrade and Compatibility Issues* for the version of LabVIEW that corresponds to the tests.
- Read the upgrade documentation
    - *Upgrade and Compatibility Issues*—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* topic, open your upgraded VIs and select **Edit»Find and Replace**.
    - LabVIEW 2019 Known Issues list—Lists bugs discovered before and throughout the release of LabVIEW 2019. Refer to the NI website at `ni.com/info` and enter the Info Code *lv2019ki* 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 2019.
- Run your own tests—Perform the same tests on the VIs in LabVIEW 2019 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 `troubleshooting_guide.html` document installed in the `labview\manuals` directory 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 2019 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 2019.

Upgrading from LabVIEW 2015

You might encounter the following compatibility issues when you upgrade to LabVIEW 2019 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.

Upgrading from LabVIEW 2016

You might encounter the following compatibility issues when you upgrade to LabVIEW 2019 from LabVIEW 2016.

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 and later, 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 issues when you upgrade to LabVIEW 2019 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

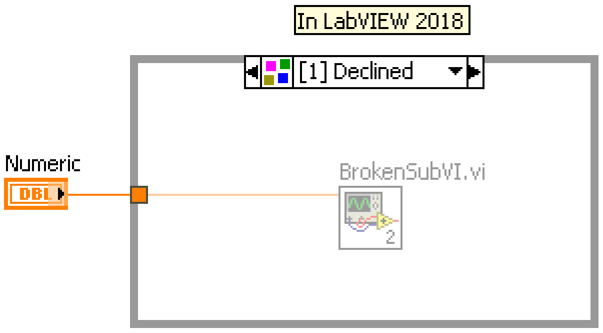
LabVIEW 2018 and later do not support the Number To Enum VI. Use the Coerce To Type function instead.

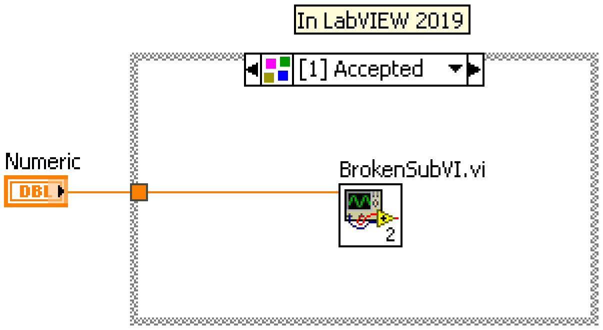
Upgrading from LabVIEW 2018

You might encounter the following compatibility issues when you upgrade to LabVIEW 2019 from LabVIEW 2018.

Behavior Changes to the Type Specialization Structure

In LabVIEW 2019, the Type Specialization structure changes the syntax error checking behavior when deciding whether to accept or decline a subdiagram. In LabVIEW 2018, the Type Specialization structure considers errors within the structure, such as broken wires, and errors from broken subVIs or other dependencies as reasons to decline a subdiagram. In LabVIEW 2019, the Type Specialization structure considers only errors within the structure as reasons to decline a subdiagram.

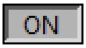
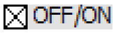
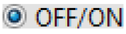
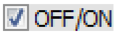
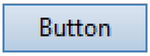
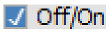




Refer to the **VI and Function Reference»Programming VIs and Functions»Structures» Type Specialization Structure** topic in the *LabVIEW Help* for more information about the Type Specialization structure.

Behavior Changes to Creating Boolean Controls and Indicators with Classic, System, or NXG Style

LabVIEW 2019 changes the behavior of creating controls and indicators from Boolean terminals when a VI is configured to use classic, system, or NXG style for creating controls and indicators. The following table compares the appearances of Boolean controls and indicators created in LabVIEW 2018 and earlier versus in LabVIEW 2019.

Style	Type	LabVIEW 2018 and earlier	LabVIEW 2019
Classic	Control	 ON	 OFF/ON
System	Control/Indicator	 OFF/ON	 OFF/ON
NXG	Control	 Button	 Off/On

Note To configure the style for creating controls and indicators, select **File»VI Properties**, select **Editor Options** from the **Category** pull-down menu, and select an appropriate style in the **Control Style for Create Control/Indicator** list.

The behavior change applies to Boolean controls and indicators that you create using the following methods:

- Use the Create Control or Create Indicator method.
- Right-click a Boolean terminal and select **Create»Control** or **Create»Indicator** from the shortcut menu.

Behavior Changes to Automatic Error Handling of the Data Value Reference Read / Write Element Border Node

When you place a pair of Data Value Reference Read / Write Element border nodes on an In Place Element structure, both the left and right border nodes have the **error out** terminal. In LabVIEW 2018, if an error occurs and the VI has automatic error handling enabled, LabVIEW displays an error dialog box for each unwired **error out** terminal. In LabVIEW 2019, LabVIEW displays only one error dialog box regardless of the number of unwired **error out** terminals.

Refer to the **VI and Function Reference»Programming VIs and Functions»Structures»In Place Element Structure»Data Value Reference Read / Write Element Border Node** topic in the *LabVIEW Help* for more information about the Data Value Reference Read / Write Element border node.

Behavior Changes to Calling Community Members in a Password-Protected Library

In LabVIEW 2018, if a friend VI calls a community member in a library that is password-protected, you must provide the password of the library the first time you edit or run the friend VI. In LabVIEW 2019, you can edit or run the friend VI without providing the password of the library.

Indicating Text Overflow in Constants, Controls, and Indicators

By default, LabVIEW 2019 indicates that visible text is cut off in strings, numerics, timestamps, text rings and enums, and combo boxes. The indication is a text fade-out effect with an arrow. To disable text overflow, right-click a control, indicator, or constant and select **Visible Items»Text Overflow**.

Terminal Name Changes to the Data Type Parsing VIs

The top-level terminal names of the Data Type Parsing VIs changed from uppercase to lowercase. Sub-level terminal names, such as cluster elements, are unchanged.

Refer to the **VI and Function Reference»Programming VIs and Functions»Cluster, Class, & Variant VIs and Functions»Variant VIs and Functions»Data Type Parsing VIs** book in the *LabVIEW Help* for more information about the Data Type Parsing VIs.

LabVIEW 2019 Features and Changes

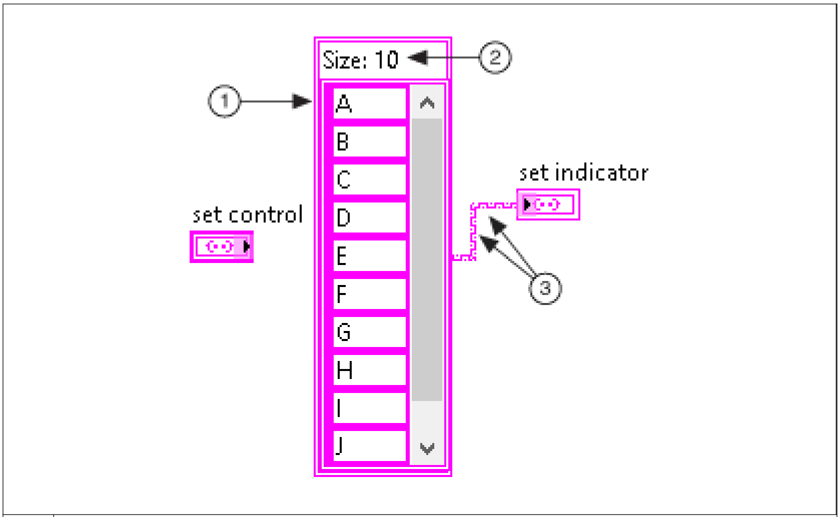
The Idea Exchange icon denotes a new feature idea that originates from a product feedback suggestion on the *NI Idea Exchange* discussion forums at `ni.com`.

Refer to *Upgrade and Compatibility Issues* for information about upgrade and compatibility issues specific to different versions of LabVIEW. Refer to the `readme.html` file in the `labview\readme` directory for known issues, a partial list of bugs fixed, additional compatibility issues, and information about late-addition features in LabVIEW 2019.

New Fundamental Collection Types: Set and Map

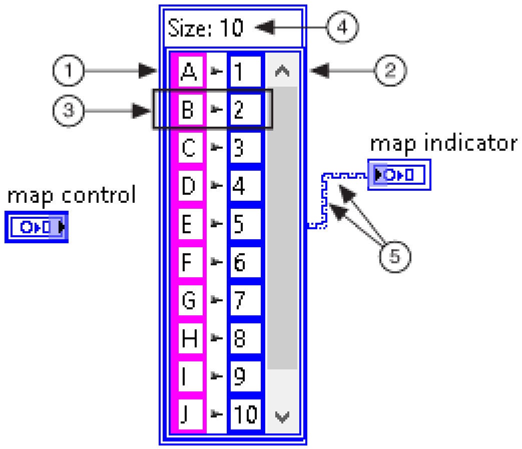
LabVIEW 2019 introduces the following *collection* data types for aggregating collections of homogeneous data: *set* and *map*. Both types maintain unique elements or entries in sorted order, which allows faster search, insertion, modification, and removal operations on the data than unordered data structures, such as arrays, even when the data size is large.

Set	A collection of unique elements of the same data type.



①	Element
②	Size—Number of elements in the set
③	Wires of set—Wire color matches the data type color of the element

Map A collection of entries consisting of key-value pairs where all the keys are unique. The key and value can each be any data type. Maps are also known as dictionaries because the key is used to look up a value.



1	Key
2	Value
3	Entry—A pair of associated key and value
4	Size—Number of entries in the map
5	Wires of map—Wire color matches the data type color of the value

Use the set and map VIs and functions, located on the **Programming»Collection** palette, and the set and map controls and indicators, located on the **Data Containers** palette, to create or manipulate sets and maps.

Note You cannot modify the embedded data in a set or map interactively or programmatically. You can update set or map data only as a whole through operations such as writing to the set or map front panel terminal or right-clicking the set or map and selecting **Data Operations»Copy Data** or **Data Operations»Paste Data**.

Refer to the **Fundamentals»Grouping Data Using Strings, Arrays, Clusters, and Collections»Concepts»Grouping Data with Collections** topic in the *LabVIEW Help* for more information about sets and maps.

Refer to the **VI and Function Reference»Programming VIs and Functions »Collection VIs and Functions** book in the *LabVIEW Help* for more information about the set and map VIs and functions.

Refer to the following VIs and project for examples of manipulating data using sets or maps:

- `labview\examples\Collections\Set Collection - Word Counting.vi`
- `labview\examples\Collections\Map Collection - Comparing Test Results.vi`
- `labview\examples\Collections\Map Collection - Word Counting.vi`
- `labview\examples\Design Patterns\Registration Map\Registration Map Usage.lvproj`

New Shortcut Menu Items

LabVIEW 2019 includes the following new shortcut menu items.

Create Constant, Create Control, and Create Indicator	Creates a constant, control, or indicator from a terminal. These shortcut menu items have long been available and are now duplicated at the top of the shortcut menu.
Replace Conversion	Converts from one numeric conversion function to another numeric conversion function when you right-click a numeric conversion function and select **Replace Conversion**. *[Idea submitted by NI Discussion Forums member EthanStern.]*
Create Color Box	Creates a color box constant, control, or indicator from a color terminal when you right-click a color terminal and select **Create Constant**, **Create Control**, or **Create Indicator**. *[Idea submitted by NI Discussion Forums member altenbach.]*
Create Cluster from Selection	Creates a cluster from a selection of constants, controls, or indicators when you right-click a selection of constants, controls, and indicators and select **Create»Cluster from Selection**. *[Idea submitted by NI Discussion Forums member okubik.]*
Create Scalar Constant or Control	Creates a scalar constant or control from an array terminal when you right-click an array terminal and select **Create»Scalar»Constant** or **Create»Scalar»Control**. *[Idea submitted by NI Discussion Forums member altenbach.]*
Replace Comparison	Converts from one comparison function to another comparison function when you right-click a comparison function and select **Replace Comparison**.

Application Builder Enhancements

LabVIEW 2019 includes the following enhancements to the LabVIEW Application Builder and build specifications:

(Windows) Distributing Your Application as Package Installers

You can distribute your applications to clients through NI Package Manager by creating a package installer (`.exe`). The *package installer* includes all the package dependencies so that your clients can install the package without network access.

To create a package installer, right-click **Build Specifications**, select **New»Package**, and select **Create a Package Installer** on the **Package Installer** page of the **Package Properties** dialog box.

Publishing NI Packages to Feeds

You can publish NI packages to a feed, so that your clients can subscribe to the feed to receive update notifications and install the NI packages via network access.

A *feed* is a collection of package files that includes a manifest containing information on the included packages. NI recommends using feeds when distributing multiple NI packages that have dependencies on one another. With feeds, you can create a single location where you host and maintain all NI packages intended for end user access.

In LabVIEW 2019, you can use the following options on the **Feed** page of the **Package Properties** dialog box to add your NI package to a local feed or publish the NI package to a SystemLink feed.

- **Add to feed**—Adds the NI package to a local feed. Your clients use NI Package Manager to subscribe to the feed to install the NI package.
- **Publish to SystemLink feed**—Publishes the NI package you created in LabVIEW to a feed on the SystemLink server. Your client can find and install the NI package in SystemLink.

Refer to the **Fundamentals»Building and Distributing Applications»Creating Build Specifications»Types of Distributions** topic in the *LabVIEW Help* for conceptual information about packages, package installers, and feeds.

Refer to the **Fundamentals»Building and Distributing Applications»Creating Build Specifications»Creating Packages for Distribution** topic in the *LabVIEW Help* for more information about creating packages or package installers and distributing them to clients.

Miscellaneous Application Builder Enhancements

The **Package Attribute** page of the **Package Properties** dialog box is renamed to **Package**. The **Package** page includes the following new options:

- **Include the NI Certificates package**—Specifies to include the NI Certificates package, ni-certificates, as a recommended dependency when clients install your application in NI Package Manager.
- **Display in the runtime deployment packages list**—Specifies to display the package in the **Related packages** list on the **Dependencies** page when you select **Runtime deployment packages only** for **Filter options**.
- **Display in NI Package Manager when filtered by products**—Specifies to display this package when you filter packages by selecting **Products only** in NI Package Manager. Enable this option if the package you create is a product that you want to include in the **Browse Products** section of the NI Package Manager.

Front Panel Enhancements

LabVIEW 2019 includes the following enhancements to the front panel:

Set and Map Controls and Indicators

In LabVIEW 2019, the **Array, Matrix & Cluster** palette is renamed **Data Containers**. The palette now includes the new map and set controls and indicators.

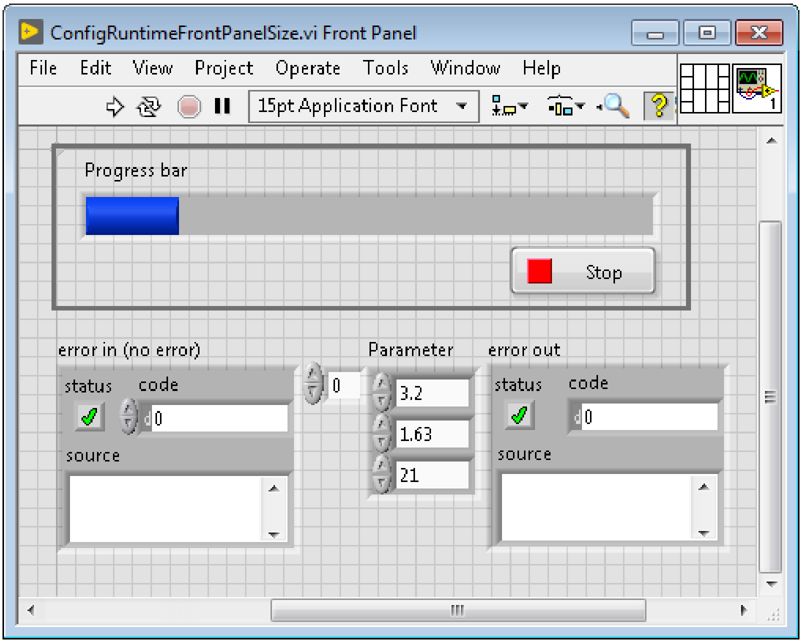
Refer to the **Fundamentals»Building the Front Panel»Concepts»Front Panel Controls and Indicators »Data Containers Controls and Indicators** topic in the *LabVIEW Help* for more information about the data container controls and indicators.

More Controls and Indicators in NXG Style

LabVIEW 2019 includes additional controls and indicators on the NXG Style palette for you to easily create numeric, Boolean, container, and I/O name controls and indicators with the same visual style as LabVIEW NXG.

Viewing Run-Time Front Panel Bounds in Edit Mode

In LabVIEW 2019, you can configure a VI to remember the run-time front panel size and scroll position. When the VI is in edit mode, LabVIEW reflects the settings as a rectangular border on the front panel. When the VI is running, the rectangular border vanishes. In the following figure, the rectangular border indicates the position and size of the visible portion of the front panel when the VI runs.



Refer to the **Fundamentals»Building the Front Panel»How-To»Designing User Interface»Viewing Run-Time Front Panel Bounds in Edit Mode** topic in the *LabVIEW Help* for instructions about how to view run-time front panel bounds.

Note LabVIEW shows run-time front panel bounds for single-pane front panels only.

[Idea submitted by NI Discussion Forums member altenbach.]

Replacing Project Items

In LabVIEW 2019, the **Replace with a packed library** shortcut menu is renamed **Replace with**. You can replace a VI, class, or library in a LabVIEW project with another item of the same type. Right-click a project item in the **Project Explorer** window and select **Replace with** to select a file to replace the original item.

Refer to the **Fundamentals»Working with Projects and Targets»Concepts»Managing a Project in LabVIEW»Replacing Project Items** topic in the *LabVIEW Help* for more information about replacing project items.

Highlighting the Execution of a Section of Code

LabVIEW 2019 allows you to highlight the execution of a section of code instead of the whole block diagram of a VI. You can set a point where you want to toggle execution highlighting by right-clicking a wire and selecting **Custom Probe»Toggle Execution**

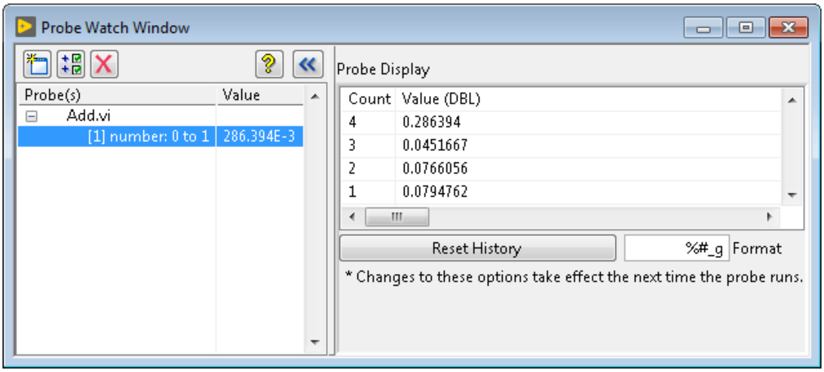
Highlighting. When the VI is running, this probe toggles the execution highlighting at the specified point.

Note The **Toggle Execution Highlighting** custom probe is not supported on RT targets.

[Idea submitted by NI Discussion Forums member therealkilkenny.]

Monitoring the Flow of Data Using History Probes

In LabVIEW 2019, you can display a history of the data that flowed through a wire by placing a history probe on the wire.



Probe(s)	Value
Add.vi	
[1] number: 0 to 1	286.394E-3

Count	Value (DBL)
4	0.286394
3	0.0451667
2	0.0766056
1	0.0794762

%#_g Format

* Changes to these options take effect the next time the probe runs.

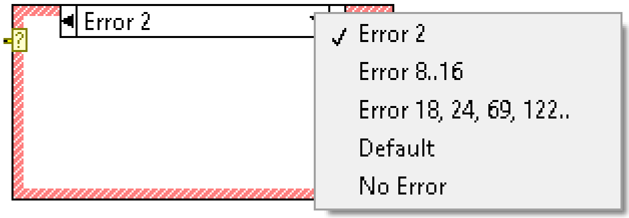
You can use history probes with the following data types:

- Boolean
- String
- Path
- I8
- I16
- I32
- I64
- U8
- U16
- U32
- U64
- SGL
- DBL
- EXT
- CSG
- CDB
- CXT

To place a history probe on a wire, right-click the wire and select **Custom Probe** and select the history probe option for your data type.

Enhancements to Handling Errors in Case Structures

In LabVIEW 2019, you can configure a case structure to execute a specific subdiagram for specific errors or list of errors by entering error value(s) or ranges in the selector labels using the same syntax as entering numeric values or ranges.

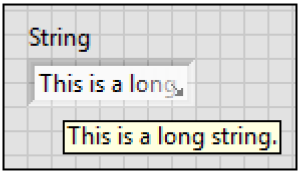


[Idea submitted by NI Discussion Forums member Hueter.]

Refer to the **Fundamentals»Running and Debugging VIs»How-To»Error Checking and Error Handling»Handling Errors Using Case Structures** topic in the *LabVIEW Help* for more information about handling errors in Case structures.

Indicating Text Overflow in Constants, Controls, and Indicators

LabVIEW 2019 indicates that visible text is cut off in strings, numerics, timestamps, text rings and enums, and combo boxes by a text fade-out effect with an arrow. Hover over the arrow to display the entire text in a tip strip.



LabVIEW 2019 enables text overflow by default. To disable text overflow, right-click a control, indicator, or constant and deselect **Visible Items»Text Overflow**.

[Idea submitted by NI Discussion Forums member Lavezza.]

Environment Enhancements

LabVIEW 2019 includes the following enhancements to the LabVIEW environment:

- When you double-click a dynamic dispatch subVI node, the **Choose Implementation** dialog box includes the new **Diagram Preview** section that displays a preview of the block diagram of the selected VI. *[Idea submitted by NI Discussion Forums member PrimaryKey.]*

- (Windows) When you build an installer in LabVIEW, the default name of the executable is changed from `setup.exe` to `install.exe`. This behavior change may have potential impact to post-build processes that rely on the `setup.exe` name. To specify a different name for the executable, use the **Installer name** option on the **Product Information** page of the **Installer Properties** dialog box.
- When you select the Error Ring drop-down arrow, the **Select Error** dialog box includes the new **Filter** option that allows you to search by keyword within the specified error code range. Use this option to narrow down the list of error codes to display in the **Error Listing** table.
- You can set the position of the left and top coordinates of an object on the front panel or block diagram on the **Appearance** page of the **Properties** dialog box. *[Idea submitted by NI Discussion Forums member smmarlow.]*
- Arrange the objects and placement of the front panel and block diagram by using the **Quick Drop** keyboard shortcut <Ctrl-F>. When you press <Ctrl-Space> and then <Ctrl-F>, LabVIEW performs the following clean-up actions:

  Note You can also arrange objects on the front panel and block diagram by pressing <Ctrl-U>.

  – On the front panel, arranges controls and indicators to be consistent with the connector pane arrangement, and resizes and moves the front panel to a consistent top-left location on the primary monitor.
  – On the block diagram, scrolls the block diagram to a reasonable location relative to the top-most and/or left-most block diagram object, and resizes and moves the block diagram to a consistent top-left location on the primary monitor.

New and Changed VIs and Functions

LabVIEW 2019 includes the following new VIs and functions. Refer to the **VI and Function Reference** book in the *LabVIEW Help* for more information about VIs, functions, and nodes.

New VIs and Functions

Map and Set VIs and Functions

The Programming palette includes the new Collection subpalette, which includes the following map and set VIs and functions:

- Map VIs and Functions—Creates and manipulates maps.
  – Build Map
  – Collection Size
  – Convert Map To Array
  – Empty Collection?
  – In Place Map Access
  – Insert Into Map
  – Look In Map

    - Map Constant
    - Read Map Max & Min Keys
    - Remove From Map
    - Registration Map VIs
        - Registration Map: Confirm Registration
        - Registration Map: Register
        - Registration Map: Unregister
- Set VIs and Functions—Creates and manipulates sets.
    - Build Set
    - Collection Size
    - Convert Set To Array
    - Empty Collection?
    - Insert Into Set
    - Element of Set?
    - Read Set Max & Min
    - Remove From Set
    - Set Cartesian Product
    - Set Constant
    - Set Difference
    - Set Intersection
    - Set Symmetric Difference
    - Set Union

JSONtext VIs

The Flatten/Unflatten String palette includes a link to install JSONtext VIs for programming JSON in LabVIEW. On the Flatten/Unflatten String palette, click **Install JSONtext Add-on** to install the JSONtext add-on from the JKI VI Package Manager (VIPM). The JSONtext VIs appear on the **Addons»JSONtext** palette.

[Special thanks to Dr. James David Powell, author of the JSONtext add-on.]

Data Type Parsing VIs

The Data Type Parsing palette includes the following new VIs:

- Get Map Collection Information—Retrieves map information from the data type stored in the input variant data.
- Get Set Collection Information—Retrieves set information from the data type stored in the input variant data.
- Is Error Cluster—Indicates whether the data type stored in the input variant is an error cluster.

Advanced File VIs

The Advanced File palette includes the following new VIs:

- Create Directory Recursive—Creates a directory and any parent directories it requires to exist.
- Create File and Containing Folders—Creates a file at a specified path and any folders within that path that do not exist.

Numeric VIs and Functions

The Numeric palette includes the following new constant and VI:

- Not A Number—Returns the value `NaN` to the block diagram. *[Idea submitted by NI Discussion Forums member altenbach.]*
- Random Number (Range)—Generates a random value from a specified range. You can use this VI with the following data types: U64, I64, and DBL.

Additional String VIs and Functions

The Additional String palette includes the following new VIs:

- 1D String Array to Delimited String—Converts the elements of a one-dimensional string array to a single string with input array elements separated by a delimiting character.
- Delimited String to 1D String Array— Converts substrings in a delimited string to the elements of a one-dimensional string array.

Miscellaneous New VIs and Functions

LabVIEW 2019 includes the following miscellaneous new VIs and functions:

- The Array palette includes the new Remove Duplicates From 1D Array VI. This VI removes duplicate elements from a 1D array. This VI preserves the original order of the elements.
- The Assert Type palette includes the new Assert Structural Type Mismatch function. This function breaks the calling VI if the input type is the same data type as any of the specified mismatch inputs, ignoring type definitions and type names. Use the Assert Structural Type Mismatch function in conjunction with the Type Specialization structure to customize sections of code in a malleable VI (`.vim`) for specific data types or to force a malleable VI to decline specific data types.
- The Protocols palette includes the new Wait for Configured Network VI. Use this VI to wait until the system can contact the remote host.
- The Synchronization palette includes the new Synchronize Data Flow VI. This VI passes through the values of the input wires after the upstream code executes. Use this VI to synchronize multiple parallel code paths at a single point of data flow to ensure a certain execution order.
- The Application Control palette includes the new Get Command Line Arguments VI. This VI returns the user-defined arguments passed from the command line when LabVIEW or a LabVIEW-built application launched. User-defined arguments start after two hyphens (--) surrounded by spaces in the command line.

Changed VIs and Functions

LabVIEW 2019 includes the following changed function:

- The Python Node, located on the **Connectivity»Python** palette, supports a new data type —Boolean. You also can use the Python Node to marshal numeric arrays to NumPy arrays.

New Properties

LabVIEW 2019 includes the following new properties:

- The VI class includes the new **Block Diagram Window:Alignment Grid Size** and **Front Panel Window:Alignment Grid Size** properties. Use these properties to read or write the alignment grid size on the block diagram or front panel of a VI. You must enable VI Scripting to use these properties.

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 2015 Upgrade Notes—*upnote15*
- LabVIEW 2016 Upgrade Notes—*upnote16*
- LabVIEW 2017 Upgrade Notes—*upnote17*
- LabVIEW 2018 Upgrade Notes—*upnote18*

371780R-01 March 1, 2019