UPGRADE NOTES

LabVIEW™ 2020 Upgrade Notes

These upgrade notes describe the process of upgrading LabVIEW for Windows, macOS, and Linux to LabVIEW 2020. 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 2020

Contents

Upgrading to LabVIEW 2020

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 2020, 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 2020 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 2020, copy files from the `labview\user.lib` directory in the previous version to the `labview\user.lib` directory in LabVIEW 2020.

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 2020 as well.

4. Convert Your VIs and Address Behavior Changes

Mass compiling your VIs in LabVIEW 2020 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 2020. 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 2020 can occasionally cause some VIs to break if they use modified features. To quickly identify and fix broken VIs in LabVIEW 2020, 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 2020 Known Issues list—Lists bugs discovered before and throughout the release of LabVIEW 2020. Refer to the NI website at `ni.com/info` and enter the Info Code *lv2020ki* 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 view the readme for your Driver, go to *ni.com/manuals*.

  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 2020.
- Run your own tests—Perform the same tests on the VIs in LabVIEW 2020 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 2020 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 2020.

Upgrading from LabVIEW 2016

You might encounter the following compatibility issues when you upgrade to LabVIEW 2020 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 2020 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.

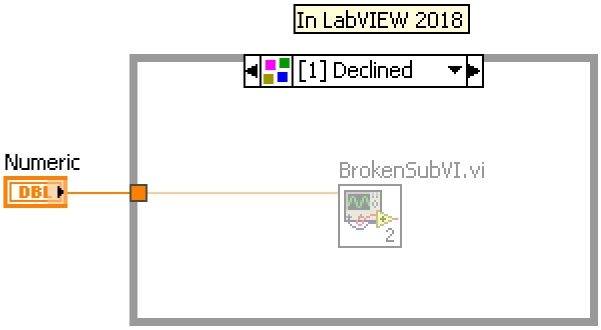
Upgrading from LabVIEW 2018

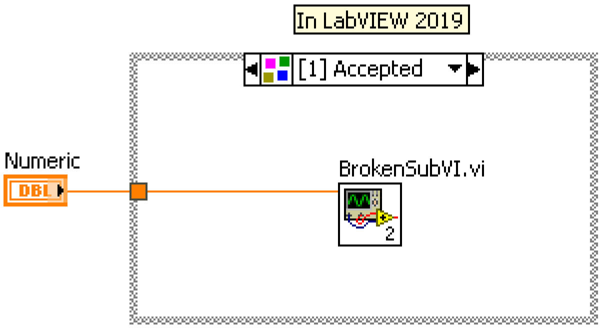
You might encounter the following compatibility issue when you upgrade to LabVIEW 2020 from LabVIEW 2018.

- The **Package Attributes** page of the Package Properties dialog box is renamed to **Package** page.
- The Array, Matrix & Cluster palette is renamed Data Containers.
- 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.

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 and earlier, the Type Specialization structure considers errors within the structure, such as broken wires, and errors from broken subVIs or other dependencies as syntax errors. In LabVIEW 2019 and later, the Type Specialization structure considers only errors within the structure as syntax errors.

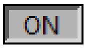
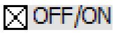
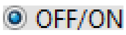
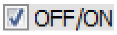
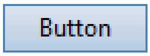
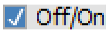




Refer to the **VI and Function Reference»Programming VIs and Functions»Structures» Type Specialization Structure** topic in the *LabVIEW Help* for more information about the Type Specialization structure.

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 and later.

Style	Type	LabVIEW 2018 and earlier	LabVIEW 2019 and later
Classic	Control		
System	Control/Indicator		
NXG	Control		

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 and later, 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 and later, 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 by a text fade out effect with an arrow. To enable text overflow in LabVIEW 2018 and earlier, right-click a control or indicator or a constant inside a cluster or cluster array 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.

Upgrading from LabVIEW 2019

You might encounter the following compatibility issue when you upgrade to LabVIEW 2020 from LabVIEW 2019.

MathScript Functionality Is No Longer Recommended

LabVIEW MathScript is no longer recommended for new designs. Visit *ni.com/migratemathscript* for more information and recommended alternatives.

SSL Encryption Is Deprecated For Remote Front Panels

In LabVIEW 2020, Secure Sockets Layer (SSL) encryption is no longer supported when you place a checkmark in the **SSL** checkbox in the **Remote Panel Server** section on the **Web Server** page of the **Options** dialog box. LabVIEW returns an error when you attempt to make a secure connection to a remote front panel server.

Deprecated VIs, Functions, and Nodes

LabVIEW 2020 and later do not support the MD5Checksum File VI. Use the File Checksum VI instead.

Name Change to the Call Parent Method Node

The Call Parent Method node is renamed Call Parent Class Method.

LabVIEW 2020 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`.

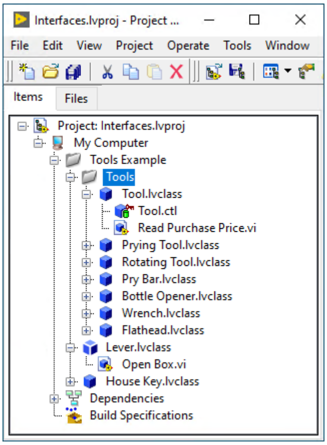
Refer to *Upgrade and Compatibility Issues* for information about upgrade and compatibility issues specific to different versions of LabVIEW. Refer to the `readme.html` file in the `labview\readme` directory for known issues, a partial list of bugs fixed, additional compatibility issues, and information about late-addition features in LabVIEW 2020.

Improving Code Flexibility Using LabVIEW Interfaces

LabVIEW 2020 introduces interfaces. An *interface* can be thought of as a class without a private data control, but that small difference enables an interface to serve entirely different purposes in software architectures than classes. Specifically, interfaces enable a form of multiple inheritance.

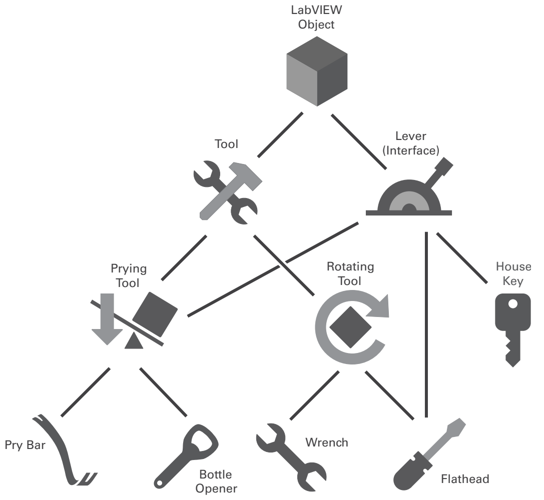
An interface declares a role that an object can play without defining how to perform that role. By inheriting from an interface, a class declares that its objects fulfill that role, and the class becomes responsible for specifying how the behaviors are performed. When a class inherits from multiple interfaces, its objects can be passed into multiple software modules requiring different roles.

The following project includes the Tool class with several child classes of different tools. The project also includes the Lever interface. You can see that the Tools class has a control that defines the data in the class, whereas the Lever interface does not have a control because interfaces do not have private data. A class is represented by a solid cube (■). An interface is represented by the faces of a cube (□). Interfaces and classes use the same file extension `.lvclass`.



```
Project: Interfaces.lvproj
  My Computer
    Tools Example
      Tools
        Tool.lvclass
          Tool.ctl
          Read Purchase Price.vi
        Prying Tool.lvclass
        Rotating Tool.lvclass
        Pry Bar.lvclass
        Bottle Opener.lvclass
        Wrench.lvclass
        Flathead.lvclass
      Lever.lvclass
        Open Box.vi
      House Key.lvclass
    Dependencies
    Build Specifications
```

The following image illustrates the inheritance relationship among the Tool classes, the House Key class, and the Lever interface. The Tool class and the Lever interface both inherit from LabVIEW Object. In addition to having their own methods and methods of their respective parent classes, the Prying Tool class and the Flathead class also inherit from the Lever interface because they both can be used as a lever. They inherit the methods of the Lever interface. Since Lever is an interface rather than a class, this multiple inheritance is legal. The three unrelated classes (Prying Tool, Flathead, House Key) have a common ancestor (Lever) other than LabVIEW Object.



You can create an interface in the following ways:

- Right-click **My Computer** in the **Project Explorer** window and select **New »Interface** from the shortcut menu.
- Select **File »New** and select **Other Files »Interface** from the **Create New** list.
- Use the LabVIEW Class:Create Interface method.

Refer to the *New and Changed Properties, Methods, and Events* section for more information about the new properties and methods for interfaces.

Refer to the following projects for examples of using interfaces:

- `labview\examples\Object-Oriented Programming\Basic Interfaces\Basic Interfaces.lvproj`
- `labview\examples\Object-Oriented Programming\Actors and Interfaces\Actors and Interfaces.lvproj`

LabVIEW Web Services Enhancements

LabVIEW 2020 provides functionalities for developing LabVIEW Web services and publishing Web services to the NI Web Server, in addition to the Application Web Server that continues to be supported.

The NI Web Server is a production-grade web server that can host user-authored services, such as LabVIEW Web services, and SystemLink services created by NI. The NI Web Server protects web applications against common web security threats, provides high scalability to many enterprise-grade data services, and allows device management.

Note The NI Web Server supports Windows (64-bit) only. The Application Web Server supports both Windows (32-bit and 64-bit) and RT targets.

You can publish a Web service to the NI Web Server through a stand-alone application, package, or package installer. You can establish secure communication between Web clients and LabVIEW Web services by enabling Secure Sockets Layer (SSL) encryption on the NI Web Server or assigning different privileges to each user role.

The default response format to a Web client changes from an XML to a JSON string. If you want to create user interfaces to visualize data and interact with Web services through a web browser, integrate WebVIs into Web services.

For more information about WebVIs, refer to the *LabVIEW NXG Web Module Manual*.

Refer to the **Fundamentals »Transferring Data »Transferring Data over a Network » Communicating with a LabVIEW Application from Web Clients »Overview: Web-based Communication with a LabVIEW Application** topic in the *LabVIEW Help* for more information about developing, hosting, and publishing a LabVIEW Web service.

Application Builder Enhancements

LabVIEW 2020 includes the following enhancements to the LabVIEW Application Builder and build specifications:

(Windows) Repairing Packages

If an installation does not run correctly, you can repair the package in NI Package Manager. In LabVIEW, you can configure a custom repair action to execute when NI Package Manager repairs the package. To create a custom repair action, on the **Advanced** page of the **Package Properties** dialog box, right-click in the **Custom actions** table and select **Add Repair Action**. You can choose from the following options:

- **Pre-repair**—Specifies the action to execute before repairing the current package.
- **Post-repair**—Specifies the action to execute after repairing the current package.
- **Post-repair all**—Specifies the action to execute after repairing all packages.

Miscellaneous Application Builder Enhancements

The **Feed** page of the **Package Properties** dialog box includes the following new options:

- **Include all dependencies in feed**—Specifies whether to add all dependencies of a package or only the package to the feed.
- **Append package version to feed location**—Specifies whether to include the package version in the feed. LabVIEW appends a subfolder with the package version to the feed location.
- **Include all dependencies in SystemLink feed**—Specifies whether to publish all dependencies of a package or only the package to the SystemLink feed.

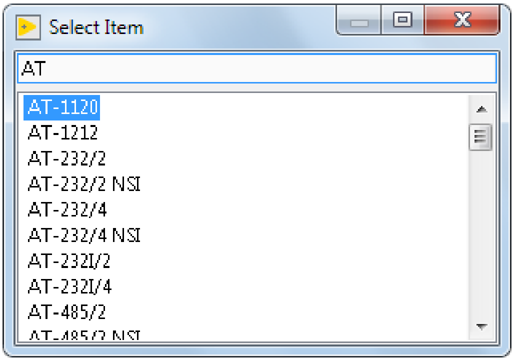
These options are available for Windows only.

Environment Enhancements

LabVIEW 2020 includes the following enhancements to the LabVIEW environment:

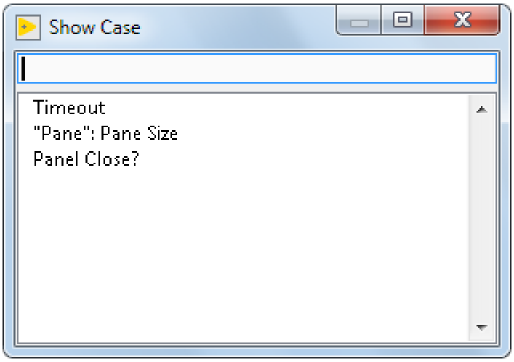
Improvements to Selecting a List Item

In LabVIEW 2020, you can change the value of a list item in a ring control, enum, or I/O control or constant by using the **Select Item** dialog box. Right-click the control or constant on the front panel or constant on the block diagram and select **Select Item**. This dialog box replaces the **Select Item** shortcut menu item.



Improvements to Moving between Subdiagrams

In LabVIEW 2020, you can change the visible frame of a Case, Event, or Conditional Disable structure by using the **Show Case** dialog box. Right-click on the structure border and select **Show Case**. This dialog box replaces the **Show Case** shortcut menu item.

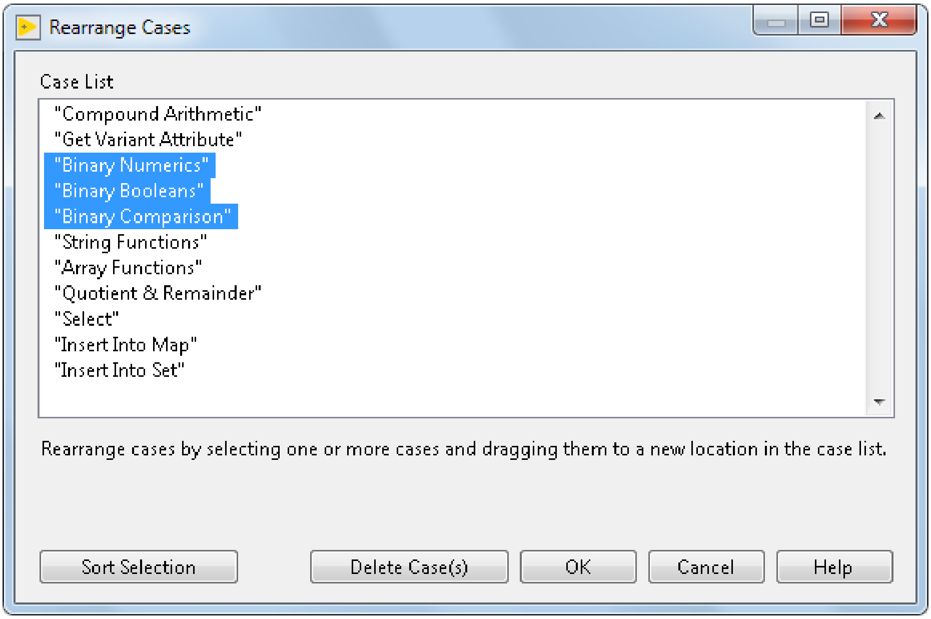


Improvements to Rearranging Cases or Subdiagrams

The **Rearrange Cases** dialog box has been redesigned and includes the following new functionalities:

- Rearrange multiple cases
- Delete one or more cases

- Sort only selected cases within the case list
- Resize the dialog box



"Compound Arithmetic"
"Get Variant Attribute"
"Binary Numerics"
"Binary Booleans"
"Binary Comparison"
"String Functions"
"Array Functions"
"Quotient & Remainder"
"Select"
"Insert Into Map"
"Insert Into Set"

Rearrange cases by selecting one or more cases and dragging them to a new location in the case list.

[Idea submitted by NI Discussion Forums member Intaris.]

Clearing Data for Variant Data Types

To clear data for a variant data type, right-click the variant control or indicator and select **Data Operations »Clear Data** .

[Idea submitted by NI Discussion Forums member altenbach.]

Improvements to Displaying Errors

LabVIEW 2020 changes the way a VI or library reports errors when the VI or library is error-free but has a broken dependency. In LabVIEW 2019 and earlier, the **Error list** window displays an error about the immediate dependency of the VI or library. Double-click the error to open the immediate dependency. In LabVIEW 2020, the **Error list** window displays the error: **`ROOT CAUSE: Dependency is broken`**. Double-click the error to open the broken dependency that causes the error.

Launching the Cluster Size Dialog Box

In addition to right-clicking, you can double-click the Array to Cluster function to open the **Cluster Size** dialog box.

[Idea submitted by NI Discussion Forums member PalanivelThiruvenkadam.]

Displaying the Last Item in a Ring Control

You can hide the last item in a ring control so that users cannot select the item while the VI runs. To show or hide the last item, right-click the ring control and select **Show/Hide Last Item When Running**. You can also use the **Last Value Hidden** property to show or hide the last item in a ring control programmatically.

Dialog Box Enhancements

LabVIEW 2020 includes the following new and updated dialog boxes for configuring settings for LabVIEW Web services.

- The **Web Service Properties** dialog box includes the following changes:
  - **NI Web Server Privileges**—Use this new page to create and define privileges for a Web service.
  - **HTTP Method VI Settings**—Use the new **NI Web Server** tab to configure privileges required to invoke the HTTP Method VI. The **Security** tab is renamed **Application Web Server**, on which you configure permissions associated with the Application Web Server.
- The **Package Properties** dialog box includes the new **Web Services** page, which you can use to configure the available Web services to include with a package.
- The **Web Services** page of the **Application Properties** dialog box includes the new **NI Web Server** option. You can specify NI Web Server as the web server for hosting Web services included in a stand-alone application by selecting this option.
- LabVIEW 2020 reorganizes the options on the **Web Server** page of the **Options** dialog box to allow you to configure access to Web services depending on whether you use the NI Web Server or Application Web Server.

The **Inheritance** page of the **Class Properties** dialog box includes the following changes:

- New **Parent Interfaces** section—Use this section to define the interfaces that the class inherits from.
- The **Inheritance Hierarchy** section is renamed **Parent Class Hierarchy**.
- The **Change Inheritance** option is renamed **Change Parent Class**.

Enhancements to the Import Shared Library Wizard

On the **Configure VIs and Controls** page of the Import Shared Library wizard, use the following new options to apply settings to multiple parameters and functions:

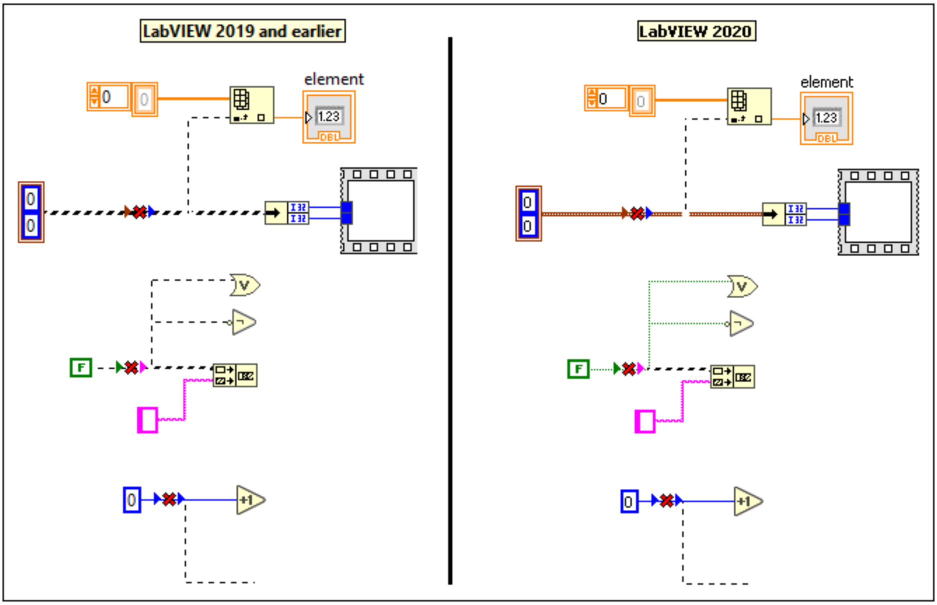
- **Apply to All Matching Parameters**—Applies the type settings of the current parameter to all function parameters that match the declaration in the header file.
- **Apply to All**—Applies the calling convention of the current function to all functions in the shared library.

Block Diagram Enhancements

LabVIEW 2020 includes the following enhancements to the block diagram and related functionality:

Cleaning Up Broken Wire Branches

In previous versions of LabVIEW, a broken wire branch causes the entire wire to display as broken and removing the wire branch removes the entire wire. In LabVIEW 2020, only broken wire branches display as broken. When you remove broken wires by selecting **Edit »Remove Broken Wire** or pressing <Ctrl-B>, LabVIEW removes only broken wire branches. LabVIEW removes the entire wire if all branches are broken.



[Idea submitted by NI Discussion Forums member altenbach.]

Hiding the Event Data Node

In an Event Structure, you can show or hide the Event Data Node for a case when all data items are unused. Right-click the structure border and select **Visible Items »Event Data Node for This Case** to show or hide the Event Data Node.

[Idea submitted by NI Discussion Forums member Broken_Arrow.]

Hiding the Iteration Terminal in Loops

In For Loops and While Loops, you can choose to show or hide the iteration terminal. Right-click the loop border and select **Iteration Terminal** to show or hide the terminal. You cannot hide the iteration terminal if the terminal is wired.

[Idea submitted by NI Discussion Forums member somebody_that_i_used_to_k.]

Swapping Wire Positions When Only One Input is Wired

In previous versions of LabVIEW, you can swap wire positions on a function that has two inputs and both inputs are wired. In LabVIEW 2020, you can swap wire positions on the function when only one input is wired. Press the <Ctrl> key while clicking the wired input to swap wire positions.

[Idea submitted by NI Discussion Forums member altenbach and tst.]

Setting Text on Icons

In LabVIEW 2020, you can quickly create an icon that contains text by using the **Quick Drop** keyboard shortcut <Ctrl-K>. When you press <Ctrl-Space> and then <Ctrl-K>, LabVIEW adds the filename of the VI as text in the icon. LabVIEW truncates the text if it is too long.

You also can right-click the VI icon and select **Set Icon to VI Name**.

You can customize the text by pressing <Ctrl-Space>, typing the text you want to appear in the icon, and then pressing <Ctrl-K>.

[Idea submitted by NI Discussion Forums member tst.]

Displaying Context Help for Objects in Quick Drop

The **Context Help** window displays information about the object you select in the **Quick Drop** dialog box.

[Idea submitted by NI Discussion Forums member elset191.]

Improvements to Wiring Terminals for Objects Inserted from Quick Drop

Objects you add on the selected wire using **Quick Drop** have improved default wiring. When you use the **Quick Drop** keyboard shortcut <Ctrl-I>, LabVIEW wires the object to the most appropriate terminals. The following functions have improved wiring:

- Binary numeric functions, such as Add, Subtract, Multiple, and Divide
- Binary Boolean functions, such as And, Or, Exclusive Or, Not And, Not Or, Not Exclusive Or, and Implies
- Binary Comparison functions, such as Equal?, Not Equal?, Greater?, Less?, Greater Or Equal?, Less Or Equal?
- Compound Arithmetic
- Get Variant Attribute
- String functions, such as Match Pattern, Search and Replace String, and Search/Split String
- Array functions, such as Delete From Array, Insert Into Array
- Quotient & Remainder
- Select

- Insert Into Map
- Insert Into Set

[Idea submitted by NI Discussion Forums member D.]*

New and Changed VIs and Functions

LabVIEW 2020 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

Web Services VIs

The Web Services palette is reorganized and includes the following new subpalettes:

- Application Web Server—Contains VIs that specifically support Web services deployed to the Application Web Server, including VIs to configure Embedded Server Pages (ESP) scripting, encrypt and decrypt data transfers, and manage HTTP sessions on the Application Web Server.
- NI Web Server—Contains the new Get Auth Details for NI Web Server VI, which returns the authentication details for the NI Web Server.

WebSockets VIs

The Data Communication palette includes a link to install WebSockets VIs for streaming data to or from web user interfaces. On the Data Communication palette, click **Install WebSockets Add-on** to install the WebSockets Toolkit from the JKI VI Package Manager (VIPM). The WebSockets VIs appear on the **Data Communication** »**WebSockets** palette.

[Special thanks to Sam Sharp, author of the WebSockets add-on.]

Multiple Errors VIs

The Dialog & User Interface palette includes the new Multiple Errors subpalette. Use the Multiple Errors VIs to convert an error cluster into different formats or to manipulate the attributes of an error cluster.

Transport Layer Security Functions

The TCP palette includes the new Transport Layer Security (TLS) palette. The TLS protocol presents an interface similar to the Transmission Control Protocol (TCP) and is encrypted and authenticated on top of TCP. LabVIEW TLS support reuses the TCP functions. LabVIEW supports TLS version 1.2. TLS functions do not support FPGA targets.

Miscellaneous New VIs and Functions

LabVIEW 2020 includes the following miscellaneous new VIs and functions:

- The following VIs can compute message digest algorithms:
  - The new File Checksum VI on the Advanced File palette computes the message digest on the contents of a file.
  - The the new Byte Array Checksum VI on the Data Manipulation palette computes the message digest on a byte array.

  LabVIEW can compute the following message digest algorithms:
  - SHA-256
  - SHA-224
  - SHA-512
  - SHA-384
  - SHA-512/256
  - SHA-512/224
  - SHA3-224
  - SHA3-256
  - SHA3-384
  - SHA3-512
- The new Get Memory Status VI on the Memory Control palette returns the amount of physical memory that your LabVIEW process uses. This VI also returns the status of system-wide memory usage.
- The new Create NI GUID VI on the Additional String palette generates a globally unique identifier (GUID) string.
- The new Range Limits for Type VI on the Numeric palette returns the maximum and minimum values of the input data type.
- The new Enum to Arrays of Enum VI on the Conversion palette creates an array of enums with the same number of elements as values in the enum.
- The new Get LabVIEW Class Parent And Member VI Information VI on the Data Type Parsing palette retrieves parent and member VI information of the LabVIEW class or interface stored in a variant.

Changed VIs and Functions

LabVIEW 2020 includes the following changed VIs and functions:

- The default data type of the **name or relative path** input of the Build Path function is changed from string to path. When you create a control or constant from this input, the data type is a path. This input still also accepts a string.
- The To More Specific Class and To More Generic Class functions add support for interfaces. Use these functions to cast a wire of one class or interface type to another class or interface type.

- The Assert Same or Descendant Type VI adds support for interfaces. This VI breaks the calling VI unless the input data is the same as or is a descendant of the input class or interface.
- The Call Parent Method node is renamed Call Parent Class Method. Use this node to call the nearest ancestor implementation of a class method.
- The TDMS File Viewer VI and **TDMS File Viewer** dialog box are redesigned to be more intuitive. You can display data from a `.tdms` file and change visualization settings all in a single page instead of switching between multiple tabs.
- In previous versions of LabVIEW, the Merge Errors function always returns the first error or warning in the input error clusters. In LabVIEW 2020, you can configure the function to return an error cluster that combines all of the input errors and warnings by right-clicking the function and selecting **Retain All Errors** .

New and Changed Properties, Methods, and Events

LabVIEW 2020 includes the following new and changed properties, methods, and events.

New Properties, Methods, and Events

LabVIEW 2020 includes the following new properties, methods, and events:

- The LVClassLibrary class includes the following new properties and methods:
  - Parent Interfaces
  - Parent Libraries
  - Is Interface
  - Write Parent Interfaces
  - Write Parent Library Paths
  - Add Parent Interface
  - Remove Parent Interface
- The Application class includes the following new properties and method:
  - Palettes:Controls Names
  - Palettes:Functions Names
  - LabVIEW Class:Create Interface
- The VI class includes the following new property and method:
  - Execution:Retain Wire Values
  - Control VI Apply Changes
- The GObject class includes the following new properties:
  - Grouped
  - Group Member Refs[]
  - Locked
- The Array class and Array Constant class include the following new property:
  - Index Rect
- The Path class includes the following new property:
  - Browse Button Rect

- The Ring class includes the following new properties:
  - Last Value Hidden
  - Flavor
- The TreeControl class includes the following new property:
  - Focus Item
- The MulticolumnListbox class includes the following new properties:
  - Selected Cell
  - Bulk Cells:Font Colors
  - Bulk Cells:Background Colors
- The Project class includes the following new method:
  - Item From Item ID
- The Numeric class, NumericConstant class, and OverridableParameterTerminal class include the new properties for configuring fixed-point behavior for numeric controls and constants and fixed-point output configuration on math primitives.
- The Ring class, Enum class, and ComboBox class include the following new events:
  - Operate Menu Activation
  - Operate Menu Dismissed
- The Control class includes the following new event:
  - Shortcut Menu Dismissed

Changed Methods

LabVIEW 2020 includes the following changed methods:

- The **Consider Parent Classes** parameter of the LVClassLibrary:Has Implementation? method is renamed **Consider Parents**. This method returns an error if the class inherits multiple methods of the same name from parent interfaces and does not provide its own override.
- The Library:Source Scope:Set and Propagate method includes the new **skip prompt?** parameter. Use this parameter to specify whether to ask before applying the access scope setting to all override VIs when the item is a dynamic dispatch VI.

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 2016 Upgrade Notes—*upnote16*
- LabVIEW 2017 Upgrade Notes—*upnote17*

- LabVIEW 2018 Upgrade Notes—*upnote18*
- LabVIEW 2019 Upgrade Notes—*upnote19*

371780T-01 March 5, 2020