# LabVIEW™

## Version 6.1

These upgrade notes describe the process of upgrading LabVIEW for Windows, Macintosh, and UNIX to version 6.1.

Refer to the *LabVIEW Release Notes* for installation instructions and other important information.

# About These Upgrade Notes

This document includes information about issues you might encounter when you upgrade to LabVIEW 6.1 and information about new features.

**For more information…**

Refer to the *LabVIEW User Manual* and the *LabVIEW Help* for more information about LabVIEW 6.1 features. Access the *LabVIEW Help* by selecting **Help»VI, Function, and How-To-Help**. Access a PDF version of the *LabVIEW User Manual* and all other LabVIEW manuals by selecting **Help»Search the LabVIEW Bookshelf**. The printed LabVIEW manuals were not updated for LabVIEW 6.1. Refer to the *LabVIEW Bookshelf* for updated PDFs of the LabVIEW manuals. You must have Adobe Acrobat Reader 4.0 or later installed to view the PDFs. Refer to the Adobe Systems Incorporated Web site at `www.adobe.com` to download Acrobat Reader.

# Contents





*321780D-01*  *November 2001*

# Upgrade Issues

If you are upgrading from LabVIEW 5.*x*, refer to the *Converting VIs, Upgrading Applibs and Toolsets,* and *Upgrading from LabVIEW 5.x* sections.

If you are upgrading a version of LabVIEW 4.*x* or earlier version, refer to the *Converting VIs, Upgrading Applibs and Toolsets,* and *Upgrading from LabVIEW 4.x* sections.

## Converting VIs

Upgrading LabVIEW applications is an automated process. When you open a VI last saved in LabVIEW 4.0 or later, LabVIEW 6.1 automatically converts and compiles the VI. You must save the VI in LabVIEW 6.1 or the

conversion process, which uses your system's memory resources, occurs every time you access the VI.

**Note** VIs you save in LabVIEW 6.1 will *not* load in older versions of LabVIEW. Select **File»Save with Options** and use the **Save for Previous** option to save VIs so they can run in older versions of LabVIEW.

You can estimate the amount of memory required to convert VIs by totalling the amount of memory that the VIs and all their subVIs occupy on disk. If the VIs are in VI libraries, add approximately 30 percent of the VI library size because the VIs are compressed. The conversion process might require at least that much memory and an additional 3 MB of memory to run LabVIEW.

If your computer does not have enough memory to convert all your VIs at once, convert the VIs in stages. Examine the hierarchy of VIs you want to convert and begin by loading and saving subVIs in the lower levels of the hierarchy. Then progress gradually to the higher levels of the hierarchy. You also can select **Tools»Advanced»Mass Compile** to convert a directory of VIs. However, this option converts VIs in a directory or VI library in alphabetical order. If the conversion process encounters a high-level VI first, **Mass Compile** requires approximately the same amount of memory as if you opened the high-level VI first.

You can monitor your memory usage by selecting **Help»About LabVIEW** to view a summary of the amount of memory you have used.

## Upgrading Applibs and Toolsets

Most existing toolsets work with LabVIEW 6.1 without problems. However, you must mass compile the VIs for use in LabVIEW 6.1. Refer to the *Converting VIs* section earlier in this document for more information about mass compiling VIs. LabVIEW 6.1 is compatible with toolsets designed for LabVIEW 4.0 and later versions, with the following exceptions:

- **(Full Development System) LabVIEW Application Builder**—You must upgrade to LabVIEW Application Builder 6.1. The Professional Development System version 6.1 includes the updated Application Builder libraries.
- **(Full Development System) LabVIEW Professional G Developers Toolkit**—If you have the Professional G Developers Toolkit 5.0 or later, you must upgrade to the LabVIEW Professional Development System version 6.1. This upgrade is free to existing users of the Professional G Developers Toolkit 5.1. The Professional Development System version 6.1 includes the new version of the Professional G Developers Toolkit.

- **LabVIEW Test Executive**—If you have LabVIEW Test Executive 5.1 or earlier, you must mass compile these VIs for use in LabVIEW 6.1. Refer to the *Converting VIs* section earlier in this document for more information about mass compiling VIs.

# Upgrading Previous Versions of LabVIEW

The following sections describe upgrade and compatibility issues specific to different versions of LabVIEW.

## Upgrading from LabVIEW 6.0

This section describes the issues you might encounter when you upgrade to LabVIEW 6.1 from LabVIEW 6.0.

### Coercion Dots and Type Definitions

Wires include information about type definitions, so you might find more coercion dots on your block diagrams. If you wire a type definition control to a VI or function terminal that is not a type definition terminal, a coercion dot appears. A coercion dot also appears if you wire an output terminal that is a type definition to an indicator that is not a type definition. These coercion dots indicate where you are not using type definitions consistently in your VIs.

In this case, coercion dots do not affect run-time performance.

**Note** Refer to the *LabVIEW Help* for information about using the Flatten To String function to flatten type definitions.

### Control Online Help Function

The **Path to the help file** input of the Control Online Help function is required. You can wire a compiled help filename (`.chm` or `.hlp`) or the full path to a compiled help file to the input. If you wire only a compiled help filename, LabVIEW searches the `labview\help` directory for that file.

### Technical Support Form

The LabVIEW installer does not install `techsup.llb`. Refer to the Technical Support section of `ni.com` to solve installation, configuration, and application problems and questions.

# Upgrading from LabVIEW 5.x

This section describes the issues you might encounter when you upgrade to LabVIEW 6.1 from LabVIEW 5.*x*.

## Converting Datalog Files

When you open a datalog file created in an earlier version of LabVIEW, LabVIEW 6.1 prompts you to convert the file to the LabVIEW 6.1 format. If you choose to convert it, LabVIEW replaces the datalog file with data converted to the new format. If you choose not to convert the file, LabVIEW 6.1 returns an error and does not open the file.

To automatically convert datalog files when you open them, add the following line to the LabVIEW preference file:

```
silentDatalogConvert=True
```

**(Macintosh)** Add the following line:

```
silentDatalogConvert:True
```

**(UNIX)** Add the following line:

```
labview.silentDatalogConvert:True
```

Set the preference to `False` if you do not want to automatically convert datalog files when you open them.

## Compatibility Issues between LabVIEW 5.x Server and LabVIEW 6.1 Client

Attempting to make a connection to the VI Server of a LabVIEW 5.*x* application from a LabVIEW 6.1 client fails because the LabVIEW 5.*x* application does not recognize some new aspects of the LabVIEW 6 VI Server protocol.

You can connect to the VI Server of a LabVIEW 6.1 application from a LabVIEW 5.*x* client.

## UDP Functions

Use the built-in UDP functions located on the **Functions»Communication»UDP** palette. The UDP VIs exist as compatibility VIs in the `vi.lib\_oldvers\_oldvers.llb`.

# Upgrading from LabVIEW 4.x

This section describes the issues you might encounter when you upgrade to LabVIEW 6.1 from LabVIEW 4.*x*.

## Converting Boolean Data to and from LabVIEW 4.x

The format in which Boolean data is stored changed between LabVIEW 4.*x* and LabVIEW 5.*x*. LabVIEW 4.*x* stores Boolean data in two bytes unless the data is in an array, in which case LabVIEW 4.*x* stores each Boolean element in a single bit. LabVIEW 6.1 stores a Boolean value in a single byte, regardless of whether it is in an array. This change enables more block diagram functions to support arrays of Boolean values and makes the behavior of these arrays more consistent with the behavior of arrays of numbers. The new Boolean data format affects data manipulation in Code Interface Nodes (CINs), but LabVIEW 6.1 provides compatibility for existing CINs.

If you write binary data that includes one or more Boolean values to a file in LabVIEW 4.*x*, its format is different than if you write the same data in LabVIEW 6.1. LabVIEW 6.1 provides a mechanism for reading binary data written in LabVIEW 4.*x* and writing binary data that LabVIEW 4.*x* can read. Five functions—Write File, Read File, Type Cast, Flatten To String, and Unflatten From String—have a **Convert 4.x Data** shortcut menu item. If you right-click the function and select this menu item, the function treats binary data as if it were written for LabVIEW 4.*x*. To produce data formatted for LabVIEW 4.*x*, use the Write File, Flatten to String, or Type Cast function. To read data formatted for LabVIEW 4.*x*, use the Read File, Unflatten From String, or Type Cast function. When you select the **Convert 4.x Data** shortcut menu item, LabVIEW 6.1 draws a red `4.x` on the function to indicate that it is converting data to or from LabVIEW 4.*x* format. To avoid the conversion of data, select the **Convert 4.x Data** shortcut menu item again to remove the checkmark next to it.

If you have several data files with Boolean values, you can create a VI that opens these files and writes the data to a new data file that LabVIEW 6.1 recognizes.

In LabVIEW 6.1, when you load a VI last saved in LabVIEW 4.*x* or previous versions, LabVIEW 6.1 automatically sets the **Convert 4.x Data** attribute on the Write File, Read File, Type Cast, Flatten To String, and Unflatten From String functions. These functions continue to function as before. When you decide that your VIs need to use the LabVIEW 6.1 Boolean data format, select the **Convert 4.x Data** shortcut menu item on each of these functions. Typically, if VIs do not need to manipulate files that contain Boolean data written in a previous version of LabVIEW, or the VIs send or receive data that contain Boolean data to or from VIs running in a previous version of LabVIEW, use the LabVIEW 6.1 Boolean data format. Support for the previous Boolean data format might be discontinued in future versions of LabVIEW.

### Converting Datalog Files

Refer to the *Converting Datalog Files* section earlier in this document for more information about converting datalog files when upgrading.

### VI Control VIs

The VI Control VIs are not in the default palette view, but exist as compatibility VIs in the `vi.lib\utility\victl.llb`. Use the VI Server functions Open VI Reference, Call By Reference, Property Node, and Invoke Node located on the **Functions»Application Control** palette instead of the VI Control VIs.

Some of the error codes returned by the VI Control VIs changed in LabVIEW 6.1. In previous versions of LabVIEW, the VI Control VIs returned the error codes 7 and 1000. The VI Control VIs in LabVIEW 6.1 return the codes 1004 and 1003. If a VI built in LabVIEW 4.*x* checks for error codes 7 and 1000, you need to make modifications for the VI to work in LabVIEW 6.1.

### DDE VIs

**(Windows)** The DDE VIs are not in the default palette view, but exist as compatibility VIs in the `vi.lib\platform\dde.llb`.

## Upgrading from LabVIEW 3.x or Earlier Versions

Refer to the National Instruments Web site at `ni.com` for information about upgrading from LabVIEW 3.*x* or earlier.

# LabVIEW 6.1 Features

Refer to the *LabVIEW User Manual* and the *LabVIEW Help* for more information about LabVIEW 6.1 features.

## Viewing and Controlling Front Panels Remotely

You can view and control a front panel remotely, either from within LabVIEW or from within a Web browser, by connecting to the LabVIEW built-in Web Server. When you open a front panel remotely from a client, the Web Server sends the front panel to the client, but the block diagram and all the subVIs remain on the server computer. You can interact with the front panel in the same way as if the VI were running on the client, except the block diagram executes on the server. Use this feature to publish entire front panels or to control your remote applications safely, easily, and quickly.

## Event-Driven Programming

Use the Event structure to handle events in an application. As with a Case structure, you can add multiple cases to the Event structure. You can then configure those cases to handle one or more events. When those events occur, LabVIEW executes the corresponding case. You can use events to detect when a user has changed a value, is closing the front panel, is exiting the application, and so on. By using an Event structure to execute code specific to an event, you reduce the need for the block diagram to poll the front panel to determine which actions the user performed. This reduces processing time and simplifies the block diagram.

## Automatic Tool Selection

When you move the cursor over objects on the front panel or block diagram, LabVIEW automatically selects the corresponding tool from the **Tools** palette. You can toggle automatic tool selection by clicking the **Automatic Tool Selection** button in the **Tools** palette or by pressing the <Shift-Tab> keys. You can disable automatic tool selection by manually selecting a tool in the **Tools** palette or by pressing the <Tab> key to move to the next tool on the palette.

The following shortcuts are new or changed:

- To enable and disable automatic tool selection, press the <Shift-Tab> keys.
- To temporarily switch to the next most useful tool when automatic tool selection is enabled, move the cursor over an object and press the <Ctrl> key. **(Macintosh)** Press the <Option> key. **(UNIX)** Press the <Meta> key.
- To temporarily switch to the Scrolling tool when automatic tool selection is enabled, move the cursor over any open space on the front panel or block diagram and press the <Shift-Ctrl> keys. **(Macintosh)** Press the <Shift-Option> keys. **(UNIX)** Press the <Meta-Shift> keys.
- To temporarily switch to the Positioning tool when automatic tool selection is enabled and another tool is selected, press the <Shift> key.
- To edit label text or block diagram constant values when automatic tool selection is enabled, double-click the label and or constant.
- To undo the last point where you set a wire, press the <Shift> key and click anywhere on the block diagram.
- To resize a node, you cannot drag the corner of the node to resize it. You must drag the top or bottom borders to add terminals to the node.
- To place a free label on the front panel or block diagram, press the <Ctrl> key and double-click any open space. **(Macintosh)** Press the <Option> key. **(UNIX)** Press the <Meta> key.

## Finding Example VIs

The *Search Examples Help* was replaced with the Find Examples tool, which you can access by selecting **Help»Find Examples** or by clicking the **Find Examples** button on the **LabVIEW** dialog box. Use this tool to browse installed example VIs by functionality or by directory structure and preview the VI descriptions. You also can search the examples by keyword.

## Converting LabVIEW Data Types to XML

Use the Flatten to XML function located on the **Functions»Advanced» Data Manipulation** palette to convert any LabVIEW data type to the XML format according to the LabVIEW XML schema. Use the Unflatten from XML function located on the same palette to convert a data type in the XML format into LabVIEW data.

**Note** For these conversions, LabVIEW uses a predefined XML schema that is described in `labview\help\LVXMLSchema.xsd`.

## Color Picker Enhancements

The color picker contains a set of user-defined and system colors and a history of the most recently used colors so you can maintain a consistent color scheme across the VIs in an application. Use system colors to adapt the appearance of a front panel to the system colors of any computer that runs the VI. Because of the enhancements to the color picker, the **Customize Window Appearance** dialog box no longer contains the **Use System Color for Panel** option. Instead, use the system colors on the color picker to color front panel objects.

You also can move the cursor over colors in the color picker to display the RGB value of the color. The transparent (**T**) box is located in the upper right corner of the color picker.

## Linking VIs to HTML Files or Compiled Help Files

You can link from a VI to an HTML or compiled help file. To do so, select **File»VI Properties** and select **Documentation** from the pull-down menu. In **Help Path** you can enter the filename for an `.htm`, `.html`, HTML Help (`.chm`), or WinHelp (`.hlp`) file. If **Help Path** contains a `.chm` filename, **Help Tag** can be an index keyword or the filename of an individual HTML file in the HTML Help project.

You also can use the Control Online Help function to link a VI to an `.htm` or `.html` file, or to an HTML Help file using an index keyword or an HTML filename.

## Case Structure Enhancements

You do not have to wire the Case structure output tunnels for every case. You can right-click the output tunnel and select **Use Default If Unwired** from the shortcut menu to use the default value for the tunnel data type for all unwired tunnels.

## Application Builder Enhancements

Refer to the *Application Builder Release Notes* for information about the Application Builder enhancements made for this release of LabVIEW.

## Tab Control Enhancements

Right-click the tabs of a tab control and select from the following options on the **Advanced** shortcut menu:

- **Customize**—Select this option to use the Control Editor to create a custom tab control. When you select this option, you can customize only the tab control and not the objects on the tab control pages.
- **Allow Multiple Colors**—Select this option to use different colors on the individual pages of a tab control.
- **Tab Layout**—Select this option to use text and/or images on the tabs of each tab control page. Use this shortcut menu to add and remove images from the tabs.
- **Tab Location**—Select this option to change the location of the tabs on the tab control.
- **(Windows)** You can place an ActiveX container on a tab control page.

Use the following properties to configure a tab control programmatically:

- **Page Label**—Use this property to read the label of a tab control page.
- **Tab Caption—**Use this property to get or set the text of a caption on a tab control page.
- **Independent Label**—Use this property to make the tab control page caption independent of the page label. After you set this property to TRUE, you can use the Tab Caption property to change the page caption.
- **Allow Multiple Colors** and **Colors**—Use these properties to configure the colors of a tab control and its individual pages.
- **Decorations on Page** and **Objects on Page**—Use these properties to obtain references to all decorations on a tab control page and to all controls, indicators, and decorations on a page, respectively.
- **Description** and **Tip Strip**—Use these properties to change the description and tip information for each individual page.

## Using Queues and Notifiers

The Notification and Queue VIs were replaced with new Notifier Operations and Queue Operations functions on the **Advanced» Synchronization** palette. The old VIs could accept only string data. The new functions are polymorphic and can accept any type of data. The new functions also are easier to use, perform slightly faster, and use less memory in built applications. Refer to the `examples\general\notifier.llb` and `examples\general\queue.llb` for examples of the Notifier Operations and Queue Operations functions.

## Point-By-Point VIs

Point By Point VIs are included in the LabVIEW Full Development and Professional Development Systems. These VIs are located on the **Functions»Analyze»Point By Point** palette. Use these VIs to conveniently and efficiently process data a point at a time. These VIs operate like a filter while producing results as each data sample is available. The Point-By-Point VIs perform the actual data analysis during the time between acquiring consecutive data points. Refer to the *Getting Started with Point-By-Point VIs* PDF located on the *LabVIEW Bookshelf* for more information about these VIs.

## Waveform Measurements VIs

The Amplitude and Levels, Cycle Average and RMS, Pulse Measurements, and Transition Measurements VIs located on the **Functions»Analyze» Waveform Measurements** palette are new. Use these VIs to characterize the time-domain shape of waveforms. Refer to the *LabVIEW Help* for more information about these VIs.

## Monitoring Front Panel Activity

Use the Wait For Front Panel Activity function located on the **Functions» Time & Dialog** palette if you want a block diagram to execute only after a user changes the value of a front panel object, such as when the user clicks a button, turns a knob, or enters data. This function activates that block diagram if the function detects user activity on the front panel. This function is similar to the Occurrence functions.

Use this function to eliminate the need for continually polling the front panel to determine if the value of a front panel object changes.

**Note** You *cannot* use this function to handle front panel events such as mouse clicks or keystrokes programmatically. Use the Event structure to handle front panel events programmatically.

## Using LabVIEW with Wireless Devices

**(Windows)** Use the IrDA functions located on the **Functions»Communications»IrDA** palette to establish a wireless communication link between VIs running on separate computers. You can write VIs that send data to and read data from remote computers using IrDA technology, which is a protocol to send data through infrared beams. To use this technology, you must have IrDA devices connected to the client and server computers.

**Note** IrDA technology is a popular technology for hand-held computers. However, currently LabVIEW is *not* available on any of the operating systems hand-held computers use, such as PalmOS and Windows CE. Refer to the *Using LabVIEW with Wireless Devices* Application Note for more information about IrDA technology and how you can use it with LabVIEW.

## Support for Custom ActiveX Automation Interfaces

If you are writing an ActiveX client that accesses properties and methods from an ActiveX server using LabVIEW, you can access custom interfaces exposed by the server. The ActiveX server writer must make sure the parameters of the properties and methods in these custom interfaces have Automation(IDispatch) data types. Refer to your server development programming environment documentation for more information about accessing custom interfaces.

## Inserting and Deleting Rows and Columns in Tables and Multicolumn Listboxes

To insert a row or column in a table, right-click where you want to insert a new row or column and select **Data Operations»Insert Row Before** or **Insert Column Before** from the shortcut menu. To delete a row or column, right-click the row or column and select **Data Operations»Delete Row** or **Delete Column** from the shortcut menu.

To insert a row or column in a multicolumn listbox, right-click where you want to insert a new row or column and select **Insert Row Before** or **Insert Column Before** from the shortcut menu. To delete a row or column, right-click the row or column and select **Delete Row** or **Delete Column** from the shortcut menu.

## Updated Graphic Support

LabVIEW supports animated GIF files on the front panel and block diagram and in Boolean and picture ring controls. You import animated GIF files in LabVIEW using the same methods you use for any graphic format your system supports. LabVIEW also supports the MNG, animated MNG, PNG, and animated PNG graphic formats, including transparency.

## Run VI Method

The Run VI method has a new parameter called **Auto Dispose Ref**. The default is FALSE. If **Auto Dispose Ref** is TRUE, the target VI detaches the reference from the main VI. When the target VI finishes executing, LabVIEW automatically disposes the reference, along with the parallel data space. Refer to the `examples\viserver\runvi.llb` for an example of using the Run VI method.

## Open VI Reference Function

The **Options** parameter in the Open VI Reference function has a new options bit **0x08 - Prepare for Reentrant Run**. This new option puts a target VI in reserve mode and allocates a parallel data space only if the target VI is reentrant. Refer to the `examples\viserver\runvi.llb` for an example of using this new option.

## Saving For a Previous Version

In LabVIEW 6.1, you can save your VIs for LabVIEW 6.0 by selecting the **Save with Options**. If you want to save your VIs for LabVIEW 5.*x*, you must open the VI in LabVIEW 6.0 and save it by selecting **File»Save With Options**.

## Source Code Control Tool Enhancements

The Source Code Control pull-down menu has been improved to include the more common source code control operations. LabVIEW performs each operation on the VI from which you launch the menu item. The old dialog boxes are consolidated into a single interface from which you can check in files, edit projects, and change local configurations.

LabVIEW supports two third-party source code control systems—Perforce and Microsoft Visual SourceSafe. ClearCase for Solaris is no longer available.

When using a third-party SCC provider such as Perforce or Microsoft Visual SourceSafe, you can leave VIs in LLBs. However, all source code control operations are performed on the LLB that contains the VI, not on the VI itself.

## Defining Custom Error Codes in Text Files

National Instruments recommends that you use the General Error Handler VI to define custom error codes in the range of 5000 through 9999. However, you also can define custom error codes in the same range by creating an XML-based text file in the `labview\user.lib\errors` directory and adding the error codes and messages to the text file. Use this method if you want to use the same custom error codes with several VIs or if you want to distribute custom error codes with an application or shared library. If you want to distribute the custom error codes with an application or shared library, you must distribute the error code text files.

## Building Shared Libraries (DLLs) That Include the Waveform Data Type

You can build shared libraries (DLLs) that include the waveform data type. You also can access the waveform data type in shared libraries with the Call Library Function Node.

## Support for Windows 2000/Me/98 Animated Menus

**(Windows)** LabVIEW supports the animated menu features of Windows 2000/Me/98, such as fading menus and pop-up windows. You must have this feature enabled on your computer to disable or enable this feature in LabVIEW. By default, this feature is disabled in Windows 98. In Windows 2000/Me, this feature is enabled. To disable this feature in LabVIEW, select **Tools»Options**, select **Miscellaneous** from the top pull-down menu, and place a checkmark in the **Display menu animation** checkbox. Refer to the Windows 2000/Me/98 documentation for information about enabling window animation on your computer.

## Front Panel Terminal Appearance

Black arrows appear on front panel terminals to indicate whether the terminal is a control or an indicator. An arrow appears on the right if the terminal is a control, and an arrow appears on the left if the terminal is an indicator. If you monitor block diagrams for coercion dots to optimize memory usage and the black arrows conflict with the default gray color of coercion dots, you can select **Tools»Options**, select the **Colors** page, and change the default color of coercion dots.

## Other LabVIEW 6.1 Features

- LabVIEW 6.1 supports Windows XP. Refer to the National Instruments Web site at `ni.com/info` and enter the info code `winxp` for more information about LabVIEW support of Windows XP.
- **(UNIX)** LabVIEW provides support for the mouse wheel on systems that support it.

- The Radar Plot and Draw Legend VIs located on the **Functions» Graphics & Sound»Picture Plots** palette are new.
- The Color to RGB and RGB to Color VIs located on the **Functions» Graphics & Sound»Picture Functions** palette are new.
- The FFT Spectrum (Mag-Phase), FFT Spectrum (Real-Im), FFT Power Spectrum, FFT Power Spectral Density, Averaged DC-RMS, Basic Averaged DC-RMS, Extract Single Tone Information, Harmonic Distortion Analyzer, and SINAD Analyzer VIs located on the **Functions»Analyze»Waveform Measurements** palette are polymorphic and can accept single- or multiple-channel data. The Limit Testing VI located on the **Functions»Analyze»Waveform Monitoring** palette is polymorphic and can accept time or frequency data.
- Use the Run-Time Menu Path property in the VI class to specify the path for a run-time menu (`.rtm`) file programmatically. This is useful if you are developing multilingual applications and you want to switch menus for each language programmatically.
- The First Call function indicates that a subVI or section of block diagram is running for the first time.
- All VIs using the FFT algorithm execute much faster due to algorithm improvements.
- The Write JPEG File, Read JPEG File, Write PNG File, and Read PNG File VIs no longer call external shared libraries. Instead they call a MNG library in LabVIEW.
- The Enum Registry Values Simple VI replaces the Enum Registry Values VI, which uses the output **data types**. The new VI uses the output **simple data types** to be compatible with other registry VIs. LabVIEW still includes the original VI in `vi.lib`, but the original VI is not located on the palette.
- When a graph or chart scale resizes, other elements on the graph or chart move and resize. To disable this behavior so the plot area size is fixed, right-click the graph or chart and select **Advanced»Auto Adjust Scales** from the shortcut menu. If you disable this behavior, the scales might clip or overlap each other.
- If you drag a graph cursor past the edge of the graph, the graph scrolls in the direction of the cursor. To disable this behavior, right-click the graph and select **Advanced»Cursors Scroll Graph** from the shortcut menu. If you disable this behavior, the scales do not update when you drag the cursor past the edge of the graph.
- You can hide a DataSocket status indicator by right-clicking the front panel object and selecting **Visible Items»DataSocket LED** from the shortcut menu. You also can use the Datasocket:LEDVisible property to hide the indicator programmatically.